

Synopsis

Version: 0.4

Stefan Seefeld, Stephen Davies

Copyright © 2001 Stefan Seefeld, Stephen Davies

This is Version: 0.4 of the Synopsis Reference Manual

Permission is granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies.

Synopsis Reference Manual Copyright Notice

Copyright (C) 2001 Stefan Seefeld, Stephen Davies

Permission is granted to use at your own risk and distribute this software in source and binary forms provided the above copyright notice and this paragraph are preserved on all copies. This software is provided "as is" with no express or implied warranty.

Acknowledgments

1 Overview

To be written...

2 The Core Module

The backbone of synopsis is an Abstract Syntax Tree, together with a Type dictionary. Both are designed to be sufficiently abstract to support a variety of languages, without giving up the necessary details to retain language specific information. The AST module contains the type that are used to generate all the nodes of a syntax tree, such as 'Class', 'Module', or 'Variable', while the Type module contains the associated meta types, i.e. for example 'Parametrized', 'BaseType', 'Declared'. Both modules define a Visitor which you can use for introspection of the AST/Type dictionary, to manipulate the data and or to generate some output from it.

2.1 The AST Module

Synopsis

[package]

Config

[module]

Configuration script module. There are a large number of command line options to control Synopsis, but there are many more options than it is feasible to implement in this fashion. Rather, Synopsis opts for the config file approach if you want full control over the process.

The problem Synopsis is very modular, and it is desirable to separate the options from different modules - something achieved by the -W flag. The modules form a hierarchical structure however, with up to three levels of depth. Some modules are used rarely, and the user may want different sets of settings depending on what they are doing - eg: generating HTML or Docbook, parsing different sections of the code, etc. We tossed about various ideas of representing options in an related way, and came up with the idea of a Python script that builds an object hierarchy that mirrors that of the modules.

The Config Script A config script is specified by passing the -c option to 'synopsis'. Options may be passed to the script via the -Wc option, as key=value pairs. For example:

```
<example>synopsis -c config.py -Wc,formatter=C++</example>
```

The script is interpreted as a Python source file, and the only requirement is that once loaded it have a global object called Config. This Config object is usually a class which is constructed with the options dictionary to retrieve an object with nested configuration objects. Every config object has at least one attribute 'name', which identifies the module the object configures.

If no config script is specified on the command line, the default class Base is used, defined in this Config module. It is recommended that the Config class in your config script derive from this Base class to extend it as you wish.

Modules In many places modules or lists of modules can be specified to perform tasks. These may be selected in one of four ways:

1. From a list of built-in defaults for that task as a simple string (depends on attribute),
2. As a member of a module or package as a simple string (depends on attribute),
3. From anywhere via a tuple of two strings: (module, class-name); for example, to use the provided DOxygen Scope Sorter, you specify ('Synopsis.Formatter.HTML.doxygen', 'DOScopeSorter') or to use your own something like ('mymodules.py', 'ScopeSorter') - Note that ending the first string in '.py' forces it to be loaded from a file, something which cannot be achieved using the next method:
4. From anywhere via an absolute dotted reference, eg: 'Synopsis.Formatter.HTML.doxygen.DOScopeSorter'.

Of these methods, 1 or 2 are preferable for brevity and 4 for absolute references, unless a filename is needed in which case you need 3.

Base [class]

The base class for configuration objects. If no config script is specified on the command line, then this class is instantiated directly.

Parser [class]

Contains nested classes for the different Parser modules.

IDL [class]

Config object for the IDL parser.

`--init--(self, argv)` [operation on IDL]

CXX [class]

Config object for the C++ parser.

`--init--(self, argv)` [operation on CXX]

Python [class]

Config object for the Python parser.

`--init--(self, argv)` [operation on Python]

Linker [class]

Contains nested classes for the Linker modules. Currently there is just Linker.

Linker [class]

Config for the main linker module. The linker performs various options on an AST, including some which are essential when merging multiple AST's.

XRefCompiler [class]

This is the config object for the XRefCompiler module. XRefCompiler compiles the text-based xref files generated by the C++ parser into a single pickled python datastructure. This data structure can then be used by HTML formatter for generating cross-reference info, or by external search tools.

`--init--(self, argv)` [operation on Linker]

Formatter [class]

Contains nested classes for the different Formatter modules.

HTML [class]

Config object for HTML Formatter. This is the most complicated config object, because the HTML formatter is itself very modular and flexible, allowing the user to customise output at many levels. All this flexibility comes at a price however - but if you read this (and send feedback to the Synopsis developers) then it should get easier.

FileSource [class]

This is the config object for the FileSource module. FileSource creates html pages that contain the actual source code for the program, which depending on the language may be highlighted and hyperlinked. Currently only the C++ parser provides this - other languages are displayed without formatting.

The formatting information is stored in a '.links' file for each input file, and since the location is specific to your project you must set this here, and FileSource is not in the default list of Page modules to use.

FileTree [class]

Config object for the FileTree module. FileTree creates a page with a tree of filenames, for use in the top-left frame.

ScopePages [class]

Config for ScopePages module. ScopePages is the module that creates the bulk of the documentation - the pages for modules and classes, with summary and detail sections for each type of ast node. ScopePages is very modular, and all the modules it uses are in the ASTFormatter and FormatStrategy modules, which is where it looks if you use the 'simple' module spec. (FIXME)

InheritanceGraph [class]

Config for InheritanceGraph module.

ModuleListing [class]

Config for ModuleListing module.

__init__(self, argv) [operation on HTML]

Initialise HTML config object. If there is a verbose argument, then the verbose attribute is set to true, causing various verbose information to be printed out.

HTML_Doxygen [class]

parents: parent class Doxygen-style HTML. This Config class actually derives from the HTML class but overrides a few options to provide an output that looks more like Doxygen's output. You may use this via something like: `<example>synopsis -Wcformatter=HTML_Doxygen</example>`

ScopePages [class]

Overrides the default ScopePages with doxygen modules.

DocBook [class]

`__init__(self, argv)` [operation on DocBook]

BoostBook [class]

`__init__(self, argv)` [operation on BoostBook]

TexInfo [class]

`__init__(self, argv)` [operation on TexInfo]

Dot [class]

`__init__(self, argv)` [operation on Dot]

HTML_Simple [class]

`__init__(self, argv)` [operation on HTML_Simple]

ASCII [class]

`__init__(self, argv)` [operation on ASCII]

DUMP [class]

`__init__(self, argv)` [operation on DUMP]

Dia [class]

`__init__(self, argv)` [operation on Dia]

`__init__(self, argv)` [operation on Base]

Initialise the Config object. The argv dictionary is used to fill in the attributes 'parser', 'linker' and 'formatter'. For example, if the dictionary contains a parser argument, then its value is used to select the parser to use by setting self.parser to the config object for that parser. The modules are selected from the 'modules' dictionary attribute of the Parser, Linker and Formatter nested classes.

Core [package]

Action [module]

Actions control the functionality of the stand-alone Synopsis. The functionality is divided into Actions, which are responsible for different steps such as checking source files, parsing, linking, formatting and caching.

The actual Action objects only contain the information needed to perform the actions, but are as lightweight as possible otherwise. This is because there is potentially a lot of memory associated with each stage, which should be allocated as late as possible and freed as soon as possible.

Action [class]

A Synopsis Action, ie: parsing, linking, formatting etc

`__init__(self, x, y, name)` [operation on Action]

`x(self)` [operation on Action]

`y(self)` [operation on Action]

`pos(self)` [operation on Action]

`move_to(self, x, y)` [operation on Action]

`move_by(self, dx, dy)` [operation on Action]

`name(self)` [operation on Action]

`set_name(self, name)` [operation on Action]

`inputs(self)` [operation on Action]

`outputs(self)` [operation on Action]

`accept(self, visitor)` [operation on Action]

ActionVisitor [class]

A visitor for the Action hierarchy

`visitAction(self, action)` [operation on ActionVisitor]

`visitSource(self, action)` [operation on ActionVisitor]

`visitParser(self, action)` [operation on ActionVisitor]

visitLinker(*self*, *action*) [operation on ActionVisitor]

visitCacher(*self*, *action*) [operation on ActionVisitor]

visitFormat(*self*, *action*) [operation on ActionVisitor]

SourceRule [class]

Base class for a source path

__init__(*self*, *pathtype*, *dir*, *glob*) [operation on SourceRule]

clone(*self*) [operation on SourceRule]
Returns a clone of this sourcepath.

SimpleSourceRule [class]

parents: parent class

__init__(*self*, *files*) [operation on SimpleSourceRule]
Creates a Simple source rule with a copy of the files list, which is a list of filenames to include

clone(*self*) [operation on SimpleSourceRule]

GlobSourceRule [class]

parents: parent class

__init__(*self*, *dirs*, *glob*, *recursive*) [operation on GlobSourceRule]

Creates a Glob source rule with a copy of the dirs list which is a list of base directories. The glob is a glob expression (string) for files to match in each directory. If the boolean recursive is set, then subdirectories are also searched.

clone(*self*) [operation on GlobSourceRule]

ExcludeSourceRule [class]

parents: parent class

__init__(*self*, *glob*) [operation on ExcludeSourceRule]
Creates an Exclude source rule with the given glob expression. Existing filenames which match the glob will be removed.

clone(*self*) [operation on ExcludeSourceRule]

SourceAction [class]

parents: parent class A Synopsis Action that loads source files

__init__(*self*, *x*, *y*, *name*) [operation on SourceAction]

rules(*self*) [operation on SourceAction]

Returns a list of rules. Each rule is a subclass of a SourceRule object with at least a 'type' attribute to determine the type of rule

accept(*self*, *visitor*) [operation on SourceAction]

ParserAction [class]

parents: parent class A Synopsis Action that parses source files.

Each parser object has a config object, which is passed to the Parser module. For a default config object, use Synopsis.Config.Base.xxx where xxx is the Parser module.

__init__(*self*, *x*, *y*, *name*) [operation on ParserAction]

config(*self*) [operation on ParserAction]

Returns the config object for this Parser

set_config(*self*, *config*) [operation on ParserAction]

Sets the config object for this Parser.

accept(*self*, *visitor*) [operation on ParserAction]

LinkerAction [class]

parents: parent class A Synopsis Action that links ASTs

__init__(*self*, *x*, *y*, *name*) [operation on LinkerAction]

config(*self*) [operation on LinkerAction]

Returns the config object for this Linker

set_config(*self*, *config*) [operation on LinkerAction]

Sets the config object for this Linker.

accept(*self*, *visitor*) [operation on LinkerAction]

CacherAction [class]

parents: parent class A Synopsis Action that caches ASTs to disk. It can optionally be used to load from a .syn file on disk, by setting the file attribute.

__init__(*self*, *x*, *y*, *name*) [operation on CacherAction]

accept(*self*, *visitor*) [operation on CacherAction]

FormatAction [class]

parents: parent class A Synopsis Action that formats ASTs into other media

__init__(*self*, *x*, *y*, *name*) [operation on FormatAction]

config(*self*) [operation on FormatAction]
Returns the config object for this Formatter

set_config(*self*, *config*) [operation on FormatAction]
Sets the config object for this Formatter.

accept(*self*, *visitor*) [operation on FormatAction]

AST [module]

Abstract Syntax Tree classes.

This file contains classes which encapsulate nodes in the AST. The base class is the Declaration class that encapsulates a named declaration. All names used are scoped tuples.

Also defined in module scope are the constants DEFAULT, PUBLIC, PROTECTED and PRIVATE.

ccmp(*a*, *b*) [function]
Compares classes of two objects

load(*filename*) [function]
Loads an AST object from the given filename

load_deps(*filename*) [function]
Loads a dependencies object from the given filename. The file will be an AST save file (usually *.syn), but this method only reads up to the dependencies object stored before the actual AST. The object returned is a list of (filename, timestamp) pairs.

save(*filename*, *ast*) [function]
Saves an AST object to the given filename

make_deps(*ast*) [function]
Creates the dependencies object to save in the .syn file from the given AST. The dependencies are a list of (filename, timestamp) pairs, extracted from `ast.files()`

AST [class]
Top-level Abstract Syntax Tree. The AST represents the whole AST for a file or group of files as a list of declarations and a types dictionary.

__init__(*self*, *files*, *declarations*, *typedict*) [operation on AST]
Constructor

files(*self*) [operation on AST]
The files this AST represents. Returns a dictionary mapping filename to SourceFile objects.

declarations(*self*) [operation on AST]
List of Declaration objects. These are the declarations at file scope

types(*self*) [operation on AST]
Dictionary of types. This is a Type.Dictionary object

accept(*self*, *visitor*) [operation on AST]
Accept the given visitor

merge(*self*, *other_ast*) [operation on AST]
Merges another AST. Files and declarations are appended to those in this AST, and types are merged by overwriting existing types - Unduplicator is responsible for sorting out the mess this may cause :)

Include [class]
Information about an include directive in a SourceFile. If the include directive required a macro expansion to get the filename, the `is_macro` will return true. If the include directive was actually an `include_next`, then `is_next` will return true.

__init__(*self*, *target*, *is_macro*, *is_next*) [operation on Include]

target(*self*) [operation on Include]

set_target(*self*, *target*) [operation on Include]

is_macro(*self*) [operation on Include]

is_next(*self*) [operation on Include]

SourceFile [class]

The information about a file that the AST was generated from. Contains filename, all declarations from this file (even nested ones) and includes (aka imports) from this file.

__init__(*self*, *filename*, *full_filename*, *language*) [operation on SourceFile]
 Constructor

is_main(*self*) [operation on SourceFile]
 Returns whether this was a main file. A source file is a main file if it was parsed directly or as an extra file. A source file is not a main file if it was just included. A source file that had no actual declarations but was given to the parser as either the main source file or an extra file is still a main file.

set_is_main(*self*, *value*) [operation on SourceFile]
 Sets the is_main flag. Typically only called once, and then may be by the linker if it discovers that a file is actually a main file elsewhere.

filename(*self*) [operation on SourceFile]
 Returns the filename of this file. The filename can be absolute or relative, depending on the settings for the Parser

full_filename(*self*) [operation on SourceFile]
 Returns the full_filename of this file. The filename can be absolute or relative, depending on the filename given to the Parser. This filename does not have any basename stripped from it, and should be accessible from the current directory as is whether absolute or relative.

language(*self*) [operation on SourceFile]
 Returns the language for this file as a string

declarations(*self*) [operation on SourceFile]
 Returns a list of declarations declared in this file

includes(*self*) [operation on SourceFile]
 Returns a the files included by this file. These may be absolute or relative, depending on the settings for the Parser. This may also include system files. The return value is a list of tuples, with each tuple being a pair (included-from-filename, include-filename). This allows distinction of files directly included (included-from-filename == self.filename()) but also allows dependency tracking since *all* files read while parsing this file are included in the list (even system files).

Declaration [class]
 Declaration base class. Every declaration has a name, comments, accessibility and type. The default accessibility is DEFAULT except for C++ where the Parser always sets it to one of the other three.

__init__(*self*, *file*, *line*, *language*, *strtype*, *name*) [operation on Declaration]

file(*self*) [operation on Declaration]
 The SourceFile this declaration appeared in

line(*self*) [operation on Declaration]
 The line of the file this declaration started at

language(*self*) [operation on Declaration]
 The language this declaration is in

type(*self*) [operation on Declaration]
 A string name of the type of this declaration

name(*self*) [operation on Declaration]
 The scoped tuple name of this declaration

comments(*self*) [operation on Declaration]
 A list of Comment objects

accept(*self*, *visitor*) [operation on Declaration]
 Visit the given visitor

accessibility(*self*) [operation on Declaration]
 One of the accessibility constants. This may be one of DEFAULT, PUBLIC, PROTECTED or PRIVATE, which are defined at module scope (Synopsis.AST)

set_name(*self*, *name*) [operation on Declaration]
 Change the name of the declaration. If you do want to change the name (and you probably don't!) then make sure you update your 'types' dictionary too!

set_accessibility(*self*, *axs*) [operation on Declaration]
 Change the accessibility

Macro [class]

parents: parent class A preprocessor macro. Note that macros are not strictly part of the AST, and as such are always in the global scope. A macro is "temporary" if it was `#undefined` in the same file it was `#defined` in.

__init__(*self*, *file*, *line*, *language*, *type*, *name*, *parameters*, *text*) [operation on Macro]
 Constructor

parameters(*self*) [operation on Macro]
 Returns a list of parameter names (strings) for this macro if it has any. Note that if the macro is not a function-like macro, this method will return None. If it is a function-like macro but with no parameters, an empty list will be returned.

text(*self*) [operation on Macro]
 Returns the replacement text for this macro as a string

accept(*self*, *visitor*) [operation on Macro]

Forward [class]

parents: parent class Forward declaration

__init__(*self*, *file*, *line*, *language*, *type*, *name*) [operation on Forward]

accept(*self*, *visitor*) [operation on Forward]

Group [class]

parents: parent class Base class for groups which contain declarations. This class doesn't correspond to any language construct. Rather, it may be used with comment-embedded grouping tags to regroup declarations that are to appear together in the manual.

__init__(*self*, *file*, *line*, *language*, *type*, *name*) [operation on Group]

declarations(*self*) [operation on Group]
The list of declarations in this group

accept(*self*, *visitor*) [operation on Group]

Scope [class]
parents: parent class Base class for scopes (named groups).

__init__(*self*, *file*, *line*, [operation on Scope]
language, *type*, *name*)

accept(*self*, *visitor*) [operation on Scope]

Module [class]
parents: parent class Module class

__init__(*self*, *file*, *line*, [operation on Module]
language, *type*, *name*)

accept(*self*, *visitor*) [operation on Module]

MetaModule [class]
parents: parent class Module Class that references all places where this Module occurs

__init__(*self*, *lang*, *type*, [operation on MetaModule]
name)

module_declarations(*self*) [operation on MetaModule]
The module declarations this metamodule subsumes

accept(*self*, *visitor*) [operation on MetaModule]

Inheritance [class]
Inheritance class. This class encapsulates the information about an inheritance, such as attributes like 'virtual' and 'public'

__init__(*self*, *type*, [operation on Inheritance]
parent, *attributes*)

type(*self*) [operation on Inheritance]
The type of inheritance ('implements', 'extends' etc)

parent(*self*) [operation on Inheritance]
The parent class or typedef declaration

attributes(*self*) [operation on Inheritance]
Attributes such as 'virtual', 'public' etc

accept(*self*, *visitor*) [operation on Inheritance]

set_parent(*self*, *parent*) [operation on Inheritance]

Class [class]

parents: parent class Class class.

__init__(*self*, *file*, *line*, *language*, *type*, *name*) [operation on Class]

parents(*self*) [operation on Class]
The list of Inheritance objects representing base classes

template(*self*) [operation on Class]
The Template Type if this is a template, or None

set_template(*self*, *template*) [operation on Class]

accept(*self*, *visitor*) [operation on Class]

Typedef [class]

parents: parent class Typedef class.

alias() – the type object referenced by this alias
constr() – boolean: true if the alias type was constructed within this typedef declaration.

__init__(*self*, *file*, *line*, *language*, *type*, *name*, *alias*, *constr*) [operation on Typedef]

alias(*self*) [operation on Typedef]
The Type object aliased by this typedef

constr(*self*) [operation on Typedef]
True if alias type was constructed here. For example, typedef struct _Foo {} Foo;

accept(*self*, *visitor*) [operation on Typedef]

set_alias(*self*, *type*) [operation on Typedef]

Enumerator [class]

parents: parent class Enumerator of an Enum. Enumerators represent the individual names and values in an enum.

`__init__`(*self*, *file*, *line*, [operation on Enumerator]
 language, *name*, *value*)

`value`(*self*) [operation on Enumerator]
 The string value of this enumerator

`accept`(*self*, *visitor*) [operation on Enumerator]

Enum [class]

parents: parent class Enum declaration. The actual names and values are encapsulated by Enumerator objects.

`__init__`(*self*, *file*, *line*, [operation on Enum]
 language, *name*, *enumerators*)

`enumerators`(*self*) [operation on Enum]
 List of Enumerator objects

`accept`(*self*, *visitor*) [operation on Enum]

Variable [class]

parents: parent class Variable definition

`__init__`(*self*, *file*, *line*, [operation on Variable]
 language, *type*, *name*, *vtype*, *constr*)

`vtype`(*self*) [operation on Variable]
 The Type object for this variable

`constr`(*self*) [operation on Variable]
 True if the type was constructed here. For example: struct
 Foo {} myFoo;

`accept`(*self*, *visitor*) [operation on Variable]

`set_vtype`(*self*, *vtype*) [operation on Variable]

Const [class]

parents: parent class Constant declaration. A constant is a name with a type and value.

`__init__`(*self*, *file*, *line*, [operation on Const]
 language, *type*, *ctype*, *name*, *value*)

`ctype`(*self*) [operation on Const]
 Type object for this const

value(*self*) [operation on Const]
The string value of this type

accept(*self*, *visitor*) [operation on Const]

set_ctype(*self*, *ctype*) [operation on Const]

Parameter [class]

Function Parameter

__init__(*self*, *premod*, *type*, *postmod*, *identifier*, *value*) [operation on Parameter]

premodifier(*self*) [operation on Parameter]
List of premodifiers such as 'in' or 'out'

type(*self*) [operation on Parameter]
The Type object

postmodifier(*self*) [operation on Parameter]
Post modifiers...

identifier(*self*) [operation on Parameter]
The string name of this parameter

value(*self*) [operation on Parameter]
The string value of this parameter

accept(*self*, *visitor*) [operation on Parameter]

set_type(*self*, *type*) [operation on Parameter]

__cmp__(*self*, *other*) [operation on Parameter]
Comparison operator

__str__(*self*) [operation on Parameter]

Function [class]

parents: parent class Function declaration. Note that function names are stored in mangled form to allow overriding. Formatters should use the `realname()` method to extract the unmangled name.

__init__(*self*, *file*, *line*, *language*, *type*, *premod*, *returnType*, *name*, *realname*) [operation on Function]

| | |
|---|-------------------------|
| premodifier(<i>self</i>) | [operation on Function] |
| List of premodifiers such as 'oneway' | |
| returnType(<i>self</i>) | [operation on Function] |
| Type object for the return type of this function | |
| realname(<i>self</i>) | [operation on Function] |
| The unmangled scoped name tuple of this function | |
| parameters(<i>self</i>) | [operation on Function] |
| The list of Parameter objects of this function | |
| postmodifier(<i>self</i>) | [operation on Function] |
| The list of postmodifier strings | |
| exceptions(<i>self</i>) | [operation on Function] |
| The list of exception Types | |
| template(<i>self</i>) | [operation on Function] |
| The Template Type if this is a template, or None | |
| set_template(<i>self</i>, <i>template</i>) | [operation on Function] |
| accept(<i>self</i>, <i>visitor</i>) | [operation on Function] |
| set_returnType(<i>self</i>, <i>type</i>) | [operation on Function] |
| __cmp__(<i>self</i>, <i>other</i>) | [operation on Function] |
| Recursively compares the typespec of the function | |

Operation [class]
 parents: parent class Operation class. An operation is related to a Function and is currently identical.

| | |
|--|--------------------------|
| __init__(<i>self</i>, <i>file</i>, <i>line</i>, <i>language</i>, <i>type</i>, <i>premod</i>, <i>returnType</i>, <i>name</i>, <i>realname</i>) | [operation on Operation] |
| accept(<i>self</i>, <i>visitor</i>) | [operation on Operation] |

CommentTag [class]
 Information about a tag in a comment. Tags can represent meta-information about a comment or extra attributes related to a declaration. For example, some tags can nominate a comment as belonging to another declaration, while others indicate information such as parameter and return type descriptions.

__init__(*self*, *name*, *text*) [operation on CommentTag]
 Constructor. Name is the name of tag, eg: 'class', 'param'.
 Text is the rest of the text for a tag.

name(*self*) [operation on CommentTag]
 Returns the name of this tag

text(*self*) [operation on CommentTag]
 Returns the text of this tag

Comment [class]

Information about a comment related to a declaration. The comments are extracted verbatim by the parsers, and various Linker CommentProcessors can select comments with appropriate formatting (eg: /** style comments, //. style comments, or all // style comments). The text field is text of the comment, less any tags that have been extracted. The summary field contains a summary of the comment, which may be equal to the comment text if there is no extra detail. The summary field is only set by Linker.Comments.Summarizer, which also ensures that there is only one comment for the declaration first. The list of tags in a comment can be extracted by a Linker CommentProcessor, or is an empty list if not set. C++ Comments may be suspect, which means that they were not immediately before a declaration, but the extract.tails option was set so they were kept for the Linker to deal with.

__init__(*self*, *text*, *file*, *line*, *suspect*) [operation on Comment]

text(*self*) [operation on Comment]
 The text of the comment

set_text(*self*, *text*) [operation on Comment]
 Changes the text

summary(*self*) [operation on Comment]
 The summary of the comment

set_summary(*self*, *summary*) [operation on Comment]
 Changes the summary

tags(*self*) [operation on Comment]
 The tags of the comment. Only CommentTag instances should be added to this list.

| | |
|---|------------------------|
| __str__ (<i>self</i>) | [operation on Comment] |
| Returns the text of the comment | |
| file (<i>self</i>) | [operation on Comment] |
| The file it was defined in | |
| line (<i>self</i>) | [operation on Comment] |
| The line it was defined at | |
| is_suspect (<i>self</i>) | [operation on Comment] |
| Returns true if the comment is suspect | |
| set_suspect (<i>self</i> , <i>suspect</i>) | [operation on Comment] |
| Sets whether the comment is suspect | |

Visitor [class]
 Visitor for AST nodes

| | |
|---|------------------------|
| visitAST (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitDeclaration (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitMacro (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitForward (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitGroup (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitScope (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitModule (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitMetaModule (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitClass (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitTypedef (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitEnumerator (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitEnum (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitVariable (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitConst (<i>self</i> , <i>node</i>) | [operation on Visitor] |
| visitFunction (<i>self</i> , <i>node</i>) | [operation on Visitor] |

visitOperation(*self*, *node*) [operation on Visitor]

visitParameter(*self*, *node*) [operation on Visitor]

visitComment(*self*, *node*) [operation on Visitor]

visitInheritance(*self*, *node*) [operation on Visitor]

Executor [module]

Executors are the implementation of the various actions. The actual Action objects themselves just contain the data needed to perform the actions, and are minimal on actual code so that they can be easily serialized. The code and data needed for the execution of an Action is implemented in the matching Executor class.

Executor [class]

Base class for executor classes, defining the common interface between each executor instance.

get_output_names(*self*) [operation on Executor]

Returns a list of (name, timestamp) tuples, representing the output from this executor. The names must be given to `get_output` in turn to retrieve the AST objects, and the timestamp may be used for build control.

prepare_output(*self*, *name*, *keep*) [operation on Executor]

Prepares an AST object for returning. For most objects, this does nothing. In the case of a cacher, this causes it to process each input in turn and store the results to disk. This is as opposed to keeping each previous input in memory while the next is parsed! Returns the AST if `keep` is set, else `None`.

get_output(*self*, *name*) [operation on Executor]

Returns the AST object for the given name. Name must one returned from the `'get_output_names'` method.

ExecutorCreator [class]

parents: parent class Creates Executor instances for Action objects

__init__(*self*, *project*, *verbose*) [operation on ExecutorCreator]

project(*self*) [operation on ExecutorCreator]

Returns the project for this creator

create(*self*, *action*) [operation on ExecutorCreator]
Creates an executor for the given action

visitAction(*self*, *action*) [operation on ExecutorCreator]
This is an unknown or incomplete Action: ignore.

visitSource(*self*, *action*) [operation on ExecutorCreator]

visitParser(*self*, *action*) [operation on ExecutorCreator]

visitLinker(*self*, *action*) [operation on ExecutorCreator]

visitCacher(*self*, *action*) [operation on ExecutorCreator]

visitFormat(*self*, *action*) [operation on ExecutorCreator]

SourceExecutor [class]
parents: parent class

__init__(*self*, *executor*, *action*) [operation on SourceExecutor]

compile_glob(*self*, *globstr*) [operation on SourceExecutor]

Returns a compiled regular expression for the given glob string. A glob string is something like "*.?pp" which gets translated into "^.*\..pp\$". Compiled regular expressions are cached in a class variable

get_output_names(*self*) [operation on SourceExecutor]

Expands the paths into a list of filenames, and return those

get_output(*self*, *name*) [operation on SourceExecutor]
Raises an exception, since the SourceAction is only used to identify files – the loading is done by the parsers themselves

ParserExecutor [class]
parents: parent class Parses the input files given by its input Source-Actions

`__init__(self, executor, action)` [operation on ParserExecutor]

`is_multi(self)` [operation on ParserExecutor]
Returns true if this parser parses multiple source files at once. This is determined by the parser type and config options.

`get_output_names(self)` [operation on ParserExecutor]
Returns the names from all connected SourceActions, and caches which source action they came from

`get_output(self, name)` [operation on ParserExecutor]

`get_parser(self)` [operation on ParserExecutor]
Returns the parser module, using the module name stored in the Action object. If the module cannot be loaded, this method will raise an exception.

LinkerExecutor [class]

parents: parent class

`__init__(self, executor, action)` [operation on LinkerExecutor]

`get_output_names(self)` [operation on LinkerExecutor]
Links multiple ASTs together, and/or performs other manipulative actions on a single AST.

`get_output(self, name)` [operation on LinkerExecutor]

`get_linker(self)` [operation on LinkerExecutor]
Returns the linker module, using the module name stored in the Action object. If the module cannot be loaded, this method will raise an exception.

CacherExecutor [class]

parents: parent class

`__init__(self, executor, action)` [operation on CacherExecutor]

`get_output_names(self)` [operation on CacherExecutor]

get_cache_filename(*self*, *name*) [operation on CacherExecutor]

Returns the filename of the cache for the input with the given name

_get_timestamp(*self*, *filename*) [operation on CacherExecutor]

Returns the timestamp of the given file, or 0 if not found

_is_up_to_date(*self*, *name*, *cache_filename*) [operation on CacherExecutor]

Returns true if the input 'name' in file 'cache_filename' is up to date. Checks all dependencies

prepare_output(*self*, *name*, *keep*) [operation on CacherExecutor]

Prepares the output, which means that it parses it, saves it to disk, and forgets about it. If keep is set, return the AST anyway

get_output(*self*, *name*) [operation on CacherExecutor]

Gets the output

FormatExecutor [class]

parents: parent class Formats the input AST given by its single input

__init__(*self*, *executor*, *action*) [operation on FormatExecutor]

get_output_names(*self*) [operation on FormatExecutor]

get_output(*self*, *name*) [operation on FormatExecutor]

FileTree [module]

FileTree [class]

Maintains a tree of directories and files.

The FileTree is constructed using the SourceFiles in the AST. Each SourceFile has a list of declarations in it already. The FileTree object organises these lists into a tree structure, where each node is a directory or a source file.

Node [class]

Base class for directories and files in the tree.

`--init--(self, path, filename)` [operation on Node]

Directory [class]

parents: parent class FileTree node for directories.

`--init--(self, path, filename)` [operation on Directory]

File [class]

parents: parent class FileTree node for files.

`--init--(self, path, filename, decls)` [operation on File]

`--init--(self)` [operation on FileTree]

`--add_dir(self, path)` [operation on FileTree]
Recursively add a directory to the tree

`--add_file(self, file, decls)` [operation on FileTree]
Recursively add a file to the tree

`set_ast(self, ast)` [operation on FileTree]
Sets the AST to use and builds the tree of Nodes

`root(self)` [operation on FileTree]
Returns the root node in the file tree, which is a Directory object. The root node is created by `set_ast()`.

Project [module]

`is_project_file(filename)` [function]

Project [class]

Encapsulates a single project. A project is a set of actions connected by channels - each action may have one or more inputs and outputs which are other actions. A project also has project-wide configuration, such as the data directory.

`--init--(self)` [operation on Project]

`filename(self)` [operation on Project]
Returns the filename of this project, or None if not set yet

`set_filename(self, filename)` [operation on Project]
Sets the filename of this file

| | |
|---|------------------------|
| data_dir (<i>self</i>) | [operation on Project] |
| set_data_dir (<i>self</i> , <i>dir</i>) | [operation on Project] |
| name (<i>self</i>) | [operation on Project] |
| set_name (<i>self</i> , <i>name</i>) | [operation on Project] |
| verbose (<i>self</i>) | [operation on Project] |
| set_verbose (<i>self</i> , <i>verbose</i>) | [operation on Project] |
| default_formatter (<i>self</i>) | [operation on Project] |
| set_default_formatter (<i>self</i> , <i>action</i>) | [operation on Project] |
| actions (<i>self</i>) | [operation on Project] |
| Returns a ProjectActions object | |
| save (<i>self</i>) | [operation on Project] |
| load (<i>self</i> , <i>filename</i>) | [operation on Project] |

ProjectActions [class]

Manages the actions in a project.

Clients can register for events related to the actions. The events supported by the listener interface are:

```
def action_added(self, action): def action_moved(self, action): def
action_removed(self, action): def channel_added(self, source, dest):
def channel_removed(self, source, dest):
```

| | |
|--|-------------------------------|
| __init__ (<i>self</i> , <i>project</i>) | [operation on ProjectActions] |
| Constructor | |

| | |
|---|-------------------------------|
| actions (<i>self</i>) | [operation on ProjectActions] |
| Returns the list of actions in this project Actions. The list returned should be considered read-only | |

| | |
|---|-------------------------------|
| get_action (<i>self</i> , <i>name</i>) | [operation on ProjectActions] |
| Returns the Action object by name. This method uses a dictionary lookup so should be preferred to iteration. Returns None if the name is not found. | |

| | |
|---|-------------------------------|
| add_listener (<i>self</i> , <i>l</i>) | [operation on ProjectActions] |
| Adds a listener to this Project Actions' events. The listener may implement any of the supported methods and will receive those events. | |

`_fire(self, signal, *args)` [operation on ProjectActions]
Fires the given event to all listeners

`add_action(self, action)` [operation on ProjectActions]
Adds the given action to this project

`move_action(self, action, x, y)` [operation on ProjectActions]
Moves the given action to the given screen coordinates

`move_action_by(self, action, dx, dy)` [operation on ProjectActions]
Moves the given action by the given screen delta-coordinates

`remove_action(self, action)` [operation on ProjectActions]
Removes the given action, and destroys all channels to/from it

`rename_action(self, action, newname)` [operation on ProjectActions]
Renames the given action to the given name

`add_channel(self, source, dest)` [operation on ProjectActions]
Adds a "channel" between two actions. Causes the output of the first to be connected to the input of the second. The event 'channel_added' is fired.

`remove_channel(self, source, dest)` [operation on ProjectActions]
Removes a "channel" between two actions. If the channel doesn't exist, it is silently ignored. The event 'channel_removed' is fired.

`action_changed(self, action)` [operation on ProjectActions]
Indicates that the given action has changed, so that listeners can update themselves

`check_name(self, action)` [operation on ProjectActions]
Checks the name, and renames if necessary

**is_valid_channel(*self*, [operation on ProjectActions]
 source, *dest*)**

Returns true if the given source-dest pair would form a valid channel. Invalid pairs (eg: formatter->formatter) return false. Cyclic channels are also disallowed.

ProjectWriter [class]

parents: parent class

**write_Project(*self*, [operation on ProjectWriter]
 project)**

**write_ProjectActions([operation on ProjectWriter]
 self, *actions*)**

**write_SimpleSourceRule([operation on ProjectWriter]
 self, *rule*)**

**write_GlobSourceRule([operation on ProjectWriter]
 self, *rule*)**

**write_ExcludeSourceRule([operation on ProjectWriter]
 self, *rule*)**

**write_SourceAction([operation on ProjectWriter]
 self, *action*)**

**write_ParserAction([operation on ProjectWriter]
 self, *action*)**

**write_LinkerAction([operation on ProjectWriter]
 self, *action*)**

**write_FormatAction([operation on ProjectWriter]
 self, *action*)**

**write_CacherAction([operation on ProjectWriter]
 self, *action*)**

ProjectReader [class]

A class that reads projects

__init__(*self*, *project*) [operation on ProjectReader]

read(*self*, *filename*) [operation on ProjectReader]

**read_Project(*self*, [operation on ProjectReader]
 proj_obj)**

| | |
|---|------------------------------|
| read_ProjectActions (<i>self</i> , <i>project_obj</i>) | [operation on ProjectReader] |
| read_SourceAction (<i>self</i> , <i>action</i>) | [operation on ProjectReader] |
| read_ParserAction (<i>self</i> , <i>action</i>) | [operation on ProjectReader] |
| read_LinkAction (<i>self</i> , <i>action</i>) | [operation on ProjectReader] |
| read_FormatAction (<i>self</i> , <i>action</i>) | [operation on ProjectReader] |
| read_SourceRule (<i>self</i> , <i>rule</i>) | [operation on ProjectReader] |
| read_CacherAction (<i>self</i> , <i>action</i>) | [operation on ProjectReader] |
| Type | [module] |
| ccmp (<i>a</i> , <i>b</i>) Compares classes of two objects | [function] |
| Error Exception class used by Type internals. | [class] |
| __init__ (<i>self</i> , <i>err</i>) | [operation on Error] |
| __repr__ (<i>self</i>) | [operation on Error] |
| Type Type abstract class. | [class] |
| __init__ (<i>self</i> , <i>language</i>) | [operation on Type] |
| language (<i>self</i>) the language this type was defined in | [operation on Type] |
| accept (<i>self</i> , <i>visitor</i>) visitor pattern accept. Visitor | [operation on Type] |
| __cmp__ (<i>self</i> , <i>other</i>) Comparison operator | [operation on Type] |

Named [class]

parents: parent class Named type abstract class

`__init__(self, language, name)` [operation on Named]`name(self)` [operation on Named]
Returns the name of this type as a scoped tuple`set_name(self, name)` [operation on Named]
Changes the name of this type**Base** [class]

parents: parent class Class for base types

`__init__(self, language, name)` [operation on Base]`accept(self, visitor)` [operation on Base]`__cmp__(self, other)` [operation on Base]
Comparison operator`__str__(self)` [operation on Base]**Dependent** [class]

parents: parent class Class for template dependent types

`__init__(self, language, name)` [operation on Dependent]`accept(self, visitor)` [operation on Dependent]`__cmp__(self, other)` [operation on Dependent]
Comparison operator`__str__(self)` [operation on Dependent]**Unknown** [class]

parents: parent class Class for not (yet) known type

`__init__(self, language, name)` [operation on Unknown]`link(self)` [operation on Unknown]
external reference this type may be associated with`resolve(self, language, name, link)` [operation on Unknown]
associate this type with an external reference, instead of a declaration

accept(*self*, *visitor*) [operation on Unknown]

__cmp__(*self*, *other*) [operation on Unknown]
Comparison operator

__str__(*self*) [operation on Unknown]

Declared [class]

parents: parent class Class for declared types

__init__(*self*, *language*, *name*, *declaration*) [operation on Declared]

declaration(*self*) [operation on Declared]
declaration object which corresponds to this type

accept(*self*, *visitor*) [operation on Declared]

__cmp__(*self*, *other*) [operation on Declared]
Comparison operator

__str__(*self*) [operation on Declared]

Template [class]

parents: parent class Class for declared parametrized types

__init__(*self*, *language*, *name*, *declaration*, *parameters*) [operation on Template]

parameters(*self*) [operation on Template]
list of type names used to declare this template

accept(*self*, *visitor*) [operation on Template]

__cmp__(*self*, *other*) [operation on Template]
Comparison operator

__str__(*self*) [operation on Template]

Modifier [class]

parents: parent class Class for alias types with modifiers (such as 'const', '&', etc.)

__init__(*self*, *language*, *alias*, *premod*, *postmod*) [operation on Modifier]

| | |
|---|-------------------------|
| alias (<i>self</i>) | [operation on Modifier] |
| the type this type refers to | |
| premod (<i>self</i>) | [operation on Modifier] |
| the modifier string | |
| postmod (<i>self</i>) | [operation on Modifier] |
| the modifier string | |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Modifier] |
| set_alias (<i>self</i> , <i>alias</i>) | [operation on Modifier] |
| __cmp__ (<i>self</i> , <i>other</i>) | [operation on Modifier] |
| Comparison operator | |
| __str__ (<i>self</i>) | [operation on Modifier] |

Array [class]

parents: parent class a modifier that adds array dimensions to a type

| | |
|--|----------------------|
| __init__ (<i>self</i> , <i>language</i> , <i>alias</i> , <i>sizes</i>) | [operation on Array] |
| alias (<i>self</i>) | [operation on Array] |
| sizes (<i>self</i>) | [operation on Array] |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Array] |
| set_alias (<i>self</i> , <i>alias</i>) | [operation on Array] |
| __cmp__ (<i>self</i> , <i>other</i>) | [operation on Array] |
| Comparison operator | |
| __str__ (<i>self</i>) | [operation on Array] |

Parametrized [class]

parents: parent class Class for parametrized type instances

| | |
|--|-----------------------------|
| __init__ (<i>self</i> , <i>language</i> , <i>template</i> , <i>parameters</i>) | [operation on Parametrized] |
| template (<i>self</i>) | [operation on Parametrized] |
| template type this is an instance of | |

parameters(*self*) [operation on Parametrized]
list of types for which this template is instantiated

accept(*self*, *visitor*) [operation on Parametrized]

set_template(*self*, *type*) [operation on Parametrized]

__cmp__(*self*, *other*) [operation on Parametrized]
Comparison operator

__str__(*self*) [operation on Parametrized]

Function [class]

parents: parent class Class for function pointer types.

__init__(*self*, *language*, *retType*, *premod*, *params*) [operation on Function]

returnType(*self*) [operation on Function]
nested return type

premod(*self*) [operation on Function]
list of premodifier strings

parameters(*self*) [operation on Function]
list of nested parameter types

accept(*self*, *visitor*) [operation on Function]

set_returnType(*self*, *returnType*) [operation on Function]

Dictionary [class]

__init__(*self*) [operation on Dictionary]

__setitem__(*self*, *name*, *type*) [operation on Dictionary]

__getitem__(*self*, *name*) [operation on Dictionary]

__delitem__(*self*, *name*) [operation on Dictionary]

has_key(*self*, *name*) [operation on Dictionary]

keys(*self*) [operation on Dictionary]

values(*self*) [operation on Dictionary]

items(*self*) [operation on Dictionary]

lookup(*self*, *name*, *scopes*) [operation on Dictionary]
locate 'name' in one of the scopes

clear(*self*) [operation on Dictionary]

merge(*self*, *dict*) [operation on Dictionary]

Visitor [class]

Visitor for Type objects

visitBaseType(*self*, *type*) [operation on Visitor]

visitUnknown(*self*, *type*) [operation on Visitor]

visitDeclared(*self*, *type*) [operation on Visitor]

visitModifier(*self*, *type*) [operation on Visitor]

visitArray(*self*, *type*) [operation on Visitor]

visitTemplate(*self*, *type*) [operation on Visitor]

visitParametrized(*self*, *type*) [operation on Visitor]

visitFunctionType(*self*, *type*) [operation on Visitor]

visitDependent(*self*, *type*) [operation on Visitor]

Util [module]

Utility functions for IDL compilers

escapifyString() – return a string with non-printing characters escaped. slashName() – format a scoped name with '/' separating components. dotName() – format a scoped name with '.' separating components. ccolonName() – format a scoped name with '::' separating components. pruneScope() – remove common prefix from a scoped name. getopt_spec(args,options,longlist) – version of getopt that adds transparent –spec= support

slashName(*scopedName*, *our_scope*) [function]

slashName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by '/' characters. If a second list is given, remove a common prefix using pruneScope().

dotName(*scopedName*, *our_scope*) [function]
dotName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by '.' characters. If a second list is given, remove a common prefix using pruneScope().

ccolonName(*scopedName*, *our_scope*) [function]
ccolonName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by ':' strings. If a second list is given, remove a common prefix using pruneScope().

pruneScope(*target_scope*, *our_scope*) [function]
pruneScope(list A, list B) -> list

Given two lists of strings (scoped names), return a copy of list A with any prefix it shares with B removed.

e.g. pruneScope(['A', 'B', 'C', 'D'], ['A', 'B', 'D']) -> ['C', 'D']

escapifyString(*str*) [function]
escapifyString(string) -> string

Return the given string with any non-printing characters escaped.

_import(*name*) [function]
import either a module, or a file.

import_object(*spec*, *defaultAttr*, *basePackage*) [function]

Imports an object according to 'spec'. spec must be either a string or a tuple of two strings. A tuple of two strings means load the module from the first string, and look for an attribute using the second string. One string is interpreted according to the optional arguments. The default is just to load the named module. 'defaultAttr' means to look for the named attribute in the module and return that. 'basePackage' means to prepend the named string to the spec before importing. Note that you may pass a list instead of a tuple, and it will have the same effect.

This is used by the HTML formatter for example, to specify page classes. Each class is in a separate module, and each module has a htmlPageAttr attribute that references the class of the Page for that module. This avoids the need to specify a list of default pages, easing maintainability.

splitAndStrip(*line*) [function]
Splits a line at the first space, then strips the second argument

open(*filename*) [function]
 open a file, generating all intermediate directories if needed

getopt_spec(*args*, *options*, *long_options*) [function]
 Transparently add `-spec=file` support to `getopt`

quote(*name*) [function]
 Quotes a base filename to remove illegal characters and keep it within a reasonable length for the filesystem.

The md5 hash function is used if the length of the name after quoting is more than 100 characters. If it is used, then as many characters at the start of the name as possible are kept intact, and the hash appended to make 100 characters.

Do not pass filenames with meaningful extensions to this function, as the hash could destroy them.

PyWriter [class]

A class that allows writing data in such a way that it can be read in by just 'exec'ing the file. You should extend it and override `write_item()`

__init__(*self*, *ostream*) [operation on PyWriter]

indent(*self*) [operation on PyWriter]

outdent(*self*) [operation on PyWriter]

ensure_import(*self*, *module*, *names*) [operation on PyWriter]

ensure_struct(*self*) [operation on PyWriter]

write_top(*self*, *str*) [operation on PyWriter]
 Writes a string to the top of the file

write(*self*, *str*) [operation on PyWriter]

write_item(*self*, *item*) [operation on PyWriter]
 Writes arbitrary items by looking up `write_Foo` functions where `Foo` is the class name of the item

flush(*self*) [operation on PyWriter]
 Writes the buffer to the stream and closes the buffer

long(*self*, *list*) [operation on PyWriter]
 Remembers list as wanting 'long' representation (an item per line)

write_list(*self*, *list*) [operation on PyWriter]
Writes a list on one line. If long(list) was previously called, the list from its cache and calls write_long_list

write_long_list(*self*, *list*) [operation on PyWriter]
Writes a list with each item on a new line

write_attr(*self*, *name*, *value*) [operation on PyWriter]

flatten_struct(*self*, *struct*) [operation on PyWriter]
Flattens a struct into a (possibly nested) list. A struct is an object with only the following members: numbers, strings, sub-structs, lists and tuples.

write_PyWriterStruct(*self*, *struct*) [operation on PyWriter]

PyWriterStruct [class]
A utility class that PyWriter uses to dump class objects. Dict is the dictionary of the class being dumped.

__init__(*self*, *dict*) [operation on PyWriterStruct]

Formatter [package]

ASCII [module]

Outputs the AST in plain ascii format similar to input.

usage() [function]
Print usage to stdout

__parseArgs(*args*) [function]

print_types(*types*) [function]

format(*args*, *ast*, *config_obj*) [function]

ASCIIFormatter [class]

parents: parent class parent class

outputs as ascii. This is to test for features still missing. The output should be compatible with the input...

__init__(*self*, *os*) [operation on ASCIIFormatter]

indent(*self*) [operation on ASCIIFormatter]

| | |
|---|-------------------------------|
| incr (<i>self</i>) | [operation on ASCIIFormatter] |
| decr (<i>self</i>) | [operation on ASCIIFormatter] |
| scope (<i>self</i>) | [operation on ASCIIFormatter] |
| set_scope (<i>self</i> , <i>name</i>) | [operation on ASCIIFormatter] |
| enterScope (<i>self</i> , <i>name</i>) | [operation on ASCIIFormatter] |
| leaveScope (<i>self</i>) | [operation on ASCIIFormatter] |
| write (<i>self</i> , <i>text</i>) | [operation on ASCIIFormatter] |
| formatType (<i>self</i> , <i>type</i> , <i>id_holder</i>) | [operation on ASCIIFormatter] |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitDependent (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitTemplate (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitDeclaration (<i>self</i> , <i>decl</i>) | [operation on ASCIIFormatter] |
| visitMacro (<i>self</i> , <i>macro</i>) | [operation on ASCIIFormatter] |
| writeComments (<i>self</i> , <i>comments</i>) | [operation on ASCIIFormatter] |

| | |
|---|-------------------------------|
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on ASCIIFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on ASCIIFormatter] |
| visitMetaModule (<i>self</i> , <i>module</i>) | [operation on ASCIIFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on ASCIIFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on ASCIIFormatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on ASCIIFormatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on ASCIIFormatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on ASCIIFormatter] |
| visitVariable (<i>self</i> , <i>var</i>) | [operation on ASCIIFormatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on ASCIIFormatter] |
| visitEnumerator (<i>self</i> , <i>enumer</i>) | [operation on ASCIIFormatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on ASCIIFormatter] |

BoostBook [module]

a BoostBook formatter

usage() [function]

Print usage to stdout

--parseArgs(*args*) [function]

format(*args*, *ast*, *config*) [function]

Formatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

| | |
|--|--|
| <code>__init__(self, os)</code> | [operation on Formatter] |
| <code>scope(self)</code> | [operation on Formatter] |
| <code>push_scope(self, newscope)</code> | [operation on Formatter] |
| <code>pop_scope(self)</code> | [operation on Formatter] |
| <code>write(self, text)</code> | [operation on Formatter] Write some text to the output stream, replacing 's with 's and indents. |
| <code>start_entity(self, __type, **__params)</code> | [operation on Formatter] Write the start of an entity, ending with a newline |
| <code>end_entity(self, type)</code> | [operation on Formatter] Write the end of an entity, starting with a newline |
| <code>write_entity(self, __type, __body, **__params)</code> | [operation on Formatter] Write a single entity on one line (though body may contain newlines) |
| <code>entity(self, __type, __body, **__params)</code> | [operation on Formatter] Return but do not write the text for an entity on one line |
| <code>reference(self, ref, label)</code> | [operation on Formatter] reference takes two strings, a reference (used to look up the symbol and generated the reference), and the label (used to actually write it) |
| <code>label(self, ref)</code> | [operation on Formatter] |
| <code>type_label(self)</code> | [operation on Formatter] |
| <code>visitBaseType(self, type)</code> | [operation on Formatter] |
| <code>visitUnknown(self, type)</code> | [operation on Formatter] |
| <code>visitDeclared(self, type)</code> | [operation on Formatter] |
| <code>visitModifier(self, type)</code> | [operation on Formatter] |
| <code>visitParametrized(self, type)</code> | [operation on Formatter] |

| | |
|---|---|
| formatType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitComment (<i>self</i> , <i>comment</i>) | [operation on Formatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on Formatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on Formatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on Formatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on Formatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on Formatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on Formatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on Formatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on Formatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on Formatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on Formatter] |
| do_function (<i>self</i> , <i>func</i>) | [operation on Formatter] Stuff common to functions and methods, constructors, de- structors |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on Formatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on Formatter] |

ClassTree [module]

Contains the utility class ClassTree, for creating inheritance trees.

sort(*list*) [function]
Utility func to sort and return the given list

ClassTree [class]
 parents: parent class Maintains a tree of classes directed by inheritance. This object always exists in HTML, since it is used for other things such as printing class bases.

__init__(*self*) [operation on ClassTree]

add_inheritance(*self*, *supername*, *subname*) [operation on ClassTree]
 Adds an edge to the graph. Supername and subname are the scoped names of the two classes involved in the edge, and are copied before being stored.

subclasses(*self*, *classname*) [operation on ClassTree]
 Returns a sorted list of all classes derived from the given class

superclasses(*self*, *classname*) [operation on ClassTree]
 Returns a sorted list of all classes the given class derives from. The classes are returned as scoped names, which you may use to lookup the class declarations in the 'types' dictionary if you need to.

classes(*self*) [operation on ClassTree]
 Returns a sorted list of all class names

add_class(*self*, *name*) [operation on ClassTree]
 Adds a class to the list of classes by name

_is_root(*self*, *name*) [operation on ClassTree]

_is_leaf(*self*, *name*) [operation on ClassTree]

roots(*self*) [operation on ClassTree]
 Returns a list of classes that have no superclasses

graphs(*self*) [operation on ClassTree]
 Returns a list of graphs. Each graph is just a list of connected classes.

leaves(*self*, *graph*) [operation on ClassTree]
 Returns a list of leaves in the given graph. A leaf is a class with no subclasses

_make_graphs(*self*) [operation on ClassTree]

visitAST(*self*, *ast*) [operation on ClassTree]
visitScope(*self*, *scope*) [operation on ClassTree]
visitClass(*self*, *clas*) [operation on ClassTree]
 Adds this class and all edges to the lists

Dia [module]

Generates a .dia file of unpositioned classes and generalizations.

k2a(*keys*) [function]
 Convert a keys dict to a string of attributes

quote(*str*) [function]
 Remove HTML chars from str

usage() [function]
 Print usage to stdout

--parseArgs(*args*) [function]

format(*args*, *ast*, *config_obj*) [function]

DiaFormatter [class]

parents: parent class parent class Outputs a Dia file

__init__(*self*, *filename*) [operation on DiaFormatter]

indent(*self*) [operation on DiaFormatter]

incr(*self*) [operation on DiaFormatter]

decr(*self*) [operation on DiaFormatter]

write(*self*, *text*) [operation on DiaFormatter]

scope(*self*) [operation on DiaFormatter]

startTag(*self*, *tagname*, [operation on DiaFormatter]
***keys*)

Starts a tag and indents, attributes using keyword arguments

startTagDict(*self*, [operation on DiaFormatter]
tagname, *attrs*)

Starts a tag and indents, attributes using dictionary argument

endTag(*self*, *tagname*) [operation on DiaFormatter]
Un-indent and closes tag

soloTag(*self*, *tagname*, [operation on DiaFormatter]
***keys*)
Writes a solo tag with attributes from keyword arguments

attribute(*self*, *name*, [operation on DiaFormatter]
type, *value*, *allow_solo*)
Writes an attribute with given name, type and value

output(*self*, [operation on DiaFormatter]
declarations)
Output declarations to file

doDiagramData(*self*) [operation on DiaFormatter]
Write the stock diagramdata stuff

doInheritance(*self*, [operation on DiaFormatter]
inherit)
Create a generalization object for one inheritance

createObjectID(*self*, [operation on DiaFormatter]
decl)
Creates a new object identifier, and remembers it with the given declaration

getObjectID(*self*, *decl*) [operation on DiaFormatter]
Returns the stored identifier for the given object

formatType(*self*, *type*) [operation on DiaFormatter]
Returns a string representation for the given type

visitBaseType(*self*, [operation on DiaFormatter]
type)

visitUnknown(*self*, [operation on DiaFormatter]
type)

visitDeclared(*self*, *type*) [operation on DiaFormatter]

visitModifier(*self*, *type*) [operation on DiaFormatter]

visitParametrized(*self*, [operation on DiaFormatter]
type)

visitFunctionType(*self*, *type*) [operation on DiaFormatter]

visitDeclaration(*self*, *decl*) [operation on DiaFormatter]

Default is to not do anything with it

visitModule(*self*, *decl*) [operation on DiaFormatter]

Just traverse child declarations

visitClass(*self*, *decl*) [operation on DiaFormatter]

Creates a Class object for one class, with attributes and operations

DocBook [module]

a DocBook formatter (producing Docbook 4.2 XML output)

usage() [function]

Print usage to stdout

--parseArgs(*args*) [function]

format(*args*, *ast*, *config*) [function]

Formatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

__init__(*self*, *os*) [operation on Formatter]

scope(*self*) [operation on Formatter]

push_scope(*self*, *newscope*) [operation on Formatter]

pop_scope(*self*) [operation on Formatter]

write(*self*, *text*) [operation on Formatter]

Write some text to the output stream, replacing 's with `s and indents.

start_entity(*self*, *__type*, ***__params*) [operation on Formatter]

Write the start of an entity, ending with a newline

| | |
|--|--------------------------|
| end_entity (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| Write the end of an entity, starting with a newline | |
| write_entity (<i>self</i> , <i>--type</i> , <i>--body</i> , <i>**--params</i>) | [operation on Formatter] |
| Write a single entity on one line (though body may contain newlines) | |
| entity (<i>self</i> , <i>--type</i> , <i>--body</i> , <i>**--params</i>) | [operation on Formatter] |
| Return but do not write the text for an entity on one line | |
| reference (<i>self</i> , <i>ref</i> , <i>label</i>) | [operation on Formatter] |
| reference takes two strings, a reference (used to look up the symbol and generated the reference), and the label (used to actually write it) | |
| label (<i>self</i> , <i>ref</i>) | [operation on Formatter] |
| type_label (<i>self</i>) | [operation on Formatter] |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitComment (<i>self</i> , <i>comment</i>) | [operation on Formatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on Formatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on Formatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on Formatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on Formatter] |

| | |
|---|--------------------------|
| visitModule (<i>self</i> , <i>module</i>) | [operation on Formatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on Formatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on Formatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on Formatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on Formatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on Formatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on Formatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on Formatter] |

DocFormatter [class]

parents: parent class A specialized version that just caters for the needs of the DocBook manual's Config section. Only modules and classes are printed, and the docbook elements classsynopsis etc are not used.

| | |
|---|-----------------------------|
| visitComment (<i>self</i> , <i>comment</i>) | [operation on DocFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on DocFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on DocFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on DocFormatter] |

Dot [module]

Uses 'dot' from graphviz to generate various graphs.

| | |
|--|------------|
| system (<i>command</i>) | [function] |
| Run the command. If the command fails, an exception SystemError is thrown. | |
| usage () | [function] |
| Print usage to stdout | |

`--parseArgs(args, config_obj)` [function]

`_rel(frm, to)` [function]
Find link to to relative to frm

`_convert_map(input, output)` [function]
convert map generated from Dot to a html region map. input and output are (open) streams

`_format(input, output, format)` [function]

`_format_png(input, output)` [function]

`_format_html(input, output)` [function]
generate (active) image for html. input and output are file names. If output ends in '.html', its stem is used with an '.png' suffix for the actual image.

`format(args, ast, config_obj)` [function]

SystemError [class]

Error thrown by the `system()` function. Attributes are 'retval', encoded as per `os.wait()`: low-byte is killing signal number, high-byte is return value of command.

`--init__(self, retval, command)` [operation on SystemError]

`--repr__(self)` [operation on SystemError]

InheritanceFormatter [class]

parents: parent class parent class A Formatter that generates an inheritance graph

`--init__(self, os, operations, attributes)` [operation on InheritanceFormatter]

`scope(self)` [operation on InheritanceFormatter]

`write(self, text)` [operation on InheritanceFormatter]

`type_ref(self)` [operation on InheritanceFormatter]

`type_label(self)` [operation on InheritanceFormatter]

`parameter(self)` [operation on InheritanceFormatter]

formatType(*self*, [operation on InheritanceFormatter]
typeObj)
Returns a reference string for the given type object

clearType(*self*) [operation on InheritanceFormatter]

writeNode(*self*, [operation on InheritanceFormatter]
ref, *name*, *label*, ****attr**)
helper method to generate output for a given node

writeEdge(*self*, [operation on InheritanceFormatter]
parent, *child*, *label*, ****attr**)

getClassname([operation on InheritanceFormatter]
self, *node*)
Returns the name of the given class node, relative to all its parents. This makes the graph simpler by making the names shorter

visitModifier([operation on InheritanceFormatter]
self, *type*)

visitUnknown([operation on InheritanceFormatter]
self, *type*)

visitBase(*self*, [operation on InheritanceFormatter]
type)

visitDependent([operation on InheritanceFormatter]
self, *type*)

visitDeclared([operation on InheritanceFormatter]
self, *type*)

visitParametrized([operation on InheritanceFormatter]
self, *type*)

visitTemplate([operation on InheritanceFormatter]
self, *type*)

visitInheritance([operation on InheritanceFormatter]
self, *node*)

visitClass(*self*, [operation on InheritanceFormatter]
node)

visitOperation([operation on InheritanceFormatter]
self, *operation*)

visitVariable([operation on InheritanceFormatter]
self, variable)

SingleInheritanceFormatter [class]

parents: parent class A Formatter that generates an inheritance graph for a specific class. This Visitor visits the AST upwards, i.e. following the inheritance links, instead of the declarations contained in a given scope.

__init__([operation on SingleInheritanceFormatter]
self, os, operations, attributes, levels, types)

visitDeclared([operation on SingleInheritanceFormatter]
self, type)

visitInheritance([operation on SingleInheritanceFormatter]
self, node)

visitClass([operation on SingleInheritanceFormatter]
self, node)

CollaborationFormatter [class]

parents: parent class A Formatter that generates a collaboration graph

__init__(*self,* [operation on CollaborationFormatter]
output)

write(*self,* [operation on CollaborationFormatter]
text)

visitClass(*self,* [operation on CollaborationFormatter]
node)

visitVariable([operation on CollaborationFormatter]
self, node)

visitBaseType([operation on CollaborationFormatter]
self, type)

visitUnknown([operation on CollaborationFormatter]
self, type)

visitDeclared([operation on CollaborationFormatter]
self, type)

visitModifier(*self*, *type*) [operation on CollaborationFormatter]

visitTemplate(*self*, *type*) [operation on CollaborationFormatter]

visitParametrized(*self*, *type*) [operation on CollaborationFormatter]

visitFunctionType(*self*, *type*) [operation on CollaborationFormatter]

DUMP [module]

Verbose attribute-oriented dump of AST. Pipe into less -r

usage() [function]
Print usage to stdout

--parseArgs(*args*) [function]

format(*args*, *ast*, *config_obj*) [function]

Dumper [class]

__init__(*self*) [operation on Dumper]

clear(*self*) [operation on Dumper]

writeln(*self*, *text*) [operation on Dumper]

lnwrite(*self*, *text*) [operation on Dumper]

write(*self*, *text*) [operation on Dumper]

indent(*self*, *str*) [operation on Dumper]

outdent(*self*) [operation on Dumper]

visit(*self*, *obj*) [operation on Dumper]

visitNone(*self*, *obj*) [operation on Dumper]

visitType(*self*, *obj*) [operation on Dumper]

visitInt(*self*, *obj*) [operation on Dumper]

visitLong(*self*, *obj*) [operation on Dumper]

| | |
|--|--------------------------------|
| visitFloat (<i>self</i> , <i>obj</i>) | [operation on Dumper] |
| visitString (<i>self</i> , <i>obj</i>) | [operation on Dumper] |
| visitTuple (<i>self</i> , <i>obj</i>) | [operation on Dumper] |
| visitList (<i>self</i> , <i>obj</i>) | [operation on Dumper] |
| _dictKey (<i>self</i> , <i>key</i>) | [operation on Dumper] |
| visitDict (<i>self</i> , <i>dict</i> , <i>namefunc</i>) | [operation on Dumper] |
| _instAttr (<i>self</i> , <i>name</i>) | [operation on Dumper] |
| visitInstance (<i>self</i> , <i>obj</i>) | [operation on Dumper] |
| HTML_Simple | [module] |
| Simpler one-page HTML output | |
| entity (<i>type</i> , <i>body</i>) | [function] |
| href (<i>ref</i> , <i>label</i>) | [function] |
| name (<i>ref</i> , <i>label</i>) | [function] |
| span (<i>clas</i> , <i>body</i>) | [function] |
| div (<i>clas</i> , <i>body</i>) | [function] |
| desc (<i>text</i>) | [function] |
| usage () | [function] |
| Print usage to stdout | |
| --parseArgs (<i>args</i>) | [function] |
| format (<i>args</i> , <i>ast</i> , <i>config_obj</i>) | [function] |
| TableOfContents | [class] |
| parents: parent class | |
| generate a dictionary of all declarations which can be looked up to create cross references. Names are fully scoped. | |
| __init__ (<i>self</i>) | [operation on TableOfContents] |
| write (<i>self</i> , <i>os</i>) | [operation on TableOfContents] |
| lookup (<i>self</i> , <i>name</i>) | [operation on TableOfContents] |

| | |
|---|--------------------------------|
| <code>insert(self, name)</code> | [operation on TableOfContents] |
| <code>visitAST(self, node)</code> | [operation on TableOfContents] |
| <code>visitDeclarator(self, node)</code> | [operation on TableOfContents] |
| <code>visitGroup(self, group)</code> | [operation on TableOfContents] |
| <code>visitModule(self, module)</code> | [operation on TableOfContents] |
| <code>visitClass(self, clas)</code> | [operation on TableOfContents] |
| <code>visitTypedef(self, typedef)</code> | [operation on TableOfContents] |
| <code>visitEnumerator(self, enumerator)</code> | [operation on TableOfContents] |
| <code>visitEnum(self, enum)</code> | [operation on TableOfContents] |
| <code>visitVariable(self, variable)</code> | [operation on TableOfContents] |
| <code>visitConst(self, const)</code> | [operation on TableOfContents] |
| <code>visitParameter(self, parameter)</code> | [operation on TableOfContents] |
| <code>visitFunction(self, function)</code> | [operation on TableOfContents] |
| <code>visitOperation(self, operation)</code> | [operation on TableOfContents] |

HTMLFormatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

| | |
|---------------------------------------|------------------------------|
| <code>__init__(self, os, toc)</code> | [operation on HTMLFormatter] |
| <code>scope(self)</code> | [operation on HTMLFormatter] |

| | |
|--|------------------------------|
| write (<i>self</i> , <i>text</i>) | [operation on HTMLFormatter] |
| reference (<i>self</i> , <i>ref</i> , <i>label</i>) | [operation on HTMLFormatter] |
| reference takes two strings, a reference (used to look up the symbol and generated the reference), and the label (used to actually write it) | |
| label (<i>self</i> , <i>ref</i>) | [operation on HTMLFormatter] |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on HTMLFormatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on HTMLFormatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on HTMLFormatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on HTMLFormatter] |
| visitGroup (<i>self</i> , <i>group</i>) | [operation on HTMLFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on HTMLFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on HTMLFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on HTMLFormatter] |

| | |
|--|---------------------------------|
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on HTMLFormatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on HTMLFormatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on HTMLFormatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on HTMLFormatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on HTMLFormatter] |
| TexInfo | [module] |
| a TexInfo formatter | |
| _replace (<i>comm</i> , <i>old</i> , <i>new</i>) | [function] |
| usage () | [function] |
| Print usage to stdout | |
| __parseArgs (<i>args</i>) | [function] |
| format (<i>args</i> , <i>ast</i> , <i>config</i>) | [function] |
| Struct | [class] |
| Dummy class. Initialise with keyword args. | |
| __init__ (<i>self</i> , <i>**keys</i>) | [operation on Struct] |
| CommentFormatter | [class] |
| A class that takes a comment Struct and formats its contents. | |
| parse (<i>self</i> , <i>comm</i>) | [operation on CommentFormatter] |
| Parse the comment struct | |
| JavadocFormatter | [class] |
| parents: parent class A formatter that formats comments similar to Javadoc @tags | |
| __init__ (<i>self</i>) | [operation on JavadocFormatter] |
| Create regex objects for regexps | |
| parse (<i>self</i> , <i>comm</i>) | [operation on JavadocFormatter] |
| Parse the comm.detail for @tags | |

extract(*self*, *regexp*, [operation on JavadocFormatter]
str)
Extracts all matches of the regexp from the text. The MatchObjects are returned in a list

parseTags(*self*, *str*, [operation on JavadocFormatter]
joiner)
Returns text, tags

parseText(*self*, *str*, [operation on JavadocFormatter]
decl)

parse_see(*self*, *str*, [operation on JavadocFormatter]
decl)
Parses inline tags

format_params(*self*, [operation on JavadocFormatter]
param_tags)
Formats a list of (param, description) tags

format_attrs(*self*, [operation on JavadocFormatter]
attr_tags)
Formats a list of (attr, description) tags

format_return(*self*, [operation on JavadocFormatter]
return_tag)
Formats a since description string

format_see(*self*, [operation on JavadocFormatter]
see_tags, *decl*)
Formats a list of (ref,description) tags

find_link(*self*, *ref*, [operation on JavadocFormatter]
decl)
Given a "reference" and a declaration, returns a HTML link. Various methods are tried to resolve the reference. First the parameters are taken off, then we try to split the ref using '.' or ':.'. The params are added back, and then we try to match this scoped name against the current scope. If that fails, then we recursively try enclosing scopes.

find_link_at(*self*, [operation on JavadocFormatter]
ref, *scope*)

`_find_method_entry(` [operation on JavadocFormatter]
 `self, name, scope)`

Tries to find a TOC entry for a method adjacent to decl. The enclosing scope is found using the types dictionary, and the `realname()`'s of all the functions compared to ref.

Escapifier [class]

parents: parent class escapify the strings to become valid texinfo text. Only replace '@' by '@@' if these are not part of valid texinfo tags.

`__init__(self)` [operation on Escapifier]

`parse(self, comm)` [operation on Escapifier]

MenuMaker [class]

parents: parent class generate a texinfo menu for the declarations of a given scope

`__init__(self, scope, os)` [operation on MenuMaker]

`write(self, text)` [operation on MenuMaker]

`start(self)` [operation on MenuMaker]

`end(self)` [operation on MenuMaker]

`visitDeclaration(self, node)` [operation on MenuMaker]

Formatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

`__init__(self, os)` [operation on Formatter]

`scope(self)` [operation on Formatter]

`write(self, text)` [operation on Formatter]

`escapify(self, label)` [operation on Formatter]

`type_label(self)` [operation on Formatter]

`decl_label(self, decl)` [operation on Formatter]

| | |
|---|--------------------------|
| formatType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| Returns a reference string for the given type object | |
| formatComments (<i>self</i> , <i>decl</i>) | [operation on Formatter] |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on Formatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on Formatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on Formatter] |
| | |
| visitConst (<i>self</i> , <i>const</i>) | [operation on Formatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on Formatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on Formatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on Formatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on Formatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on Formatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on Formatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on Formatter] |

| | |
|---|--|
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on Formatter] |
| TOC | [module] |
| Table of Contents classes | |
| Linker | [class] |
| Abstract class for linking declarations. This class has only one method, <code>link(decl)</code> , which returns the link for the given declaration. This is dependant on the type of formatter used, and so a <code>Linker</code> derivative must be passed to the <code>TOC</code> upon creation. | |
| link (<i>decl</i>) | [operation on Linker] |
| TocEntry | [class] |
| Struct for an entry in the table of contents. Vars: <code>link</code> , <code>lang</code> , <code>type</code> (all strings) Also: <code>name</code> (scoped) | |
| __init__ (<i>self</i> , <i>name</i> , <i>link</i> , <i>lang</i> , <i>type</i>) | [operation on TocEntry] |
| TableOfContents | [class] |
| parents: parent class | |
| Maintains a dictionary of all declarations which can be looked up to create cross references. Names are fully scoped. | |
| __init__ (<i>self</i> , <i>linker</i>) | [operation on TableOfContents linker is an instance that implements the <code>Linker</code> interface and is used to generate the links from declarations. |
| lookup (<i>self</i> , <i>name</i>) | [operation on TableOfContents] |
| size (<i>self</i>) | [operation on TableOfContents] |
| insert (<i>self</i> , <i>entry</i>) | [operation on TableOfContents] |
| store (<i>self</i> , <i>file</i>) | [operation on TableOfContents store the table of contents into a file, such that it can be used later when cross referencing] |
| load (<i>self</i> , <i>resource</i>) | [operation on TableOfContents] |
| visitAST (<i>self</i> , <i>ast</i>) | [operation on TableOfContents] |
| visitDeclaration (<i>self</i> , <i>decl</i>) | [operation on TableOfContents] |

visitForward(*self*, *decl*) [operation on TableOfContents]

xref [module]

CrossReferencer [class]

Handle cross-reference files

__init__(*self*) [operation on CrossReferencer]

load(*self*, *filename*) [operation on CrossReferencer]
Loads data from the given filename

get_info(*self*, *name*) [operation on CrossReferencer]
Retrieves the info for the give name. The info is returned as a 3-tuple of [list of definitions], [list of calls], [list of other references]. The element of each list is another 3-tuple of (file, line, scope of reference). The last element is the scope in which the reference was made, which is often a function scope, denoted as starting with a backtick, eg: `Builder::'add_operation(std::string)`. Returns None if there is no xref info for the given name.

get_possible_names(*self*, *name*) [operation on CrossReferencer]
Returns a list of possible scoped names that end with the given name. These scoped names can be passed to `get_info()`. Returns None if there is no scoped name ending with the given name.

get_page_for(*self*, *name*) [operation on CrossReferencer]
Returns the number of the page that the xref info for the given name is on, or None if not found.

get_page_info(*self*) [operation on CrossReferencer]
Returns a list of pages, each consisting of a list of names on that page. This method is intended to be used by whatever generates the files...

get_all_names(*self*) [operation on CrossReferencer]
Returns a list of all names

HTML [package]

ASTFormatter [module]

AST Formatting classes.

This module contains classes for formatting parts of a scope page (class, module, etc with methods, variables etc. The actual formatting of the declarations is delegated to multiple strategies for each part of the page, and are defined in the FormatStrategy module.

Part [class]

parents: parent class parent class Base class for formatting a Part of a Scope Page.

This class contains functionality for modularly formatting an AST node and its children for display. It is typically used to construct Heading, Summary and Detail formatters. Strategy objects are added according to configuration, and this base class then checks which format methods each strategy implements. For each AST declaration visited, the Part asks all Strategies which implement the appropriate format method to generate output for that declaration. The final writing of the formatted html is delegated to the writeSectionStart, writeSectionEnd, and writeSectionItem methods, which must be implemented in a subclass.

__init__(*self*, *page*) [operation on Part]

page(*self*) [operation on Part]

filename(*self*) [operation on Part]

os(*self*) [operation on Part]

scope(*self*) [operation on Part]

write(*self*, *text*) [operation on Part]

_init_formatters(*self*, [operation on Part]
 config_option, *type_msg*)
 Loads strategies from config file

addFormatter(*self*, [operation on Part]
 formatterClass)
 Adds the given formatter Class. An object is instantiated from the class passing self to the constructor. Stores the object, and stores which format methods it overrides

type_ref(*self*) [operation on Part]

type_label(*self*) [operation on Part]

declarator(*self*) [operation on Part]

parameter(*self*) [operation on Part]

reference(*self*, *name*, *label*, *keys*)** [operation on Part]

Returns a reference to the given name. The name is a scoped name, and the optional label is an alternative name to use as the link text. The name is looked up in the TOC so the link may not be local. The optional keys are appended as attributes to the A tag.

label(*self*, *name*, *label*) [operation on Part]

Create a label for the given name. The label is an anchor so it can be referenced by other links. The name of the label is derived by looking up the name in the TOC and using the link in the TOC entry. The optional label is an alternative name to use as the displayed name. If the name is not found in the TOC then the name is not anchored and just label is returned (or name if no label is given).

formatDeclaration(*self*, *decl*, *method*) [operation on Part]

Format decl using named method of each formatter. Each formatter returns two strings - type and name. All the types are joined and all the names are joined separately. The consolidated type and name strings are then passed to writeSectionItem.

process(*self*, *decl*) [operation on Part]

Formats the given decl, creating the output for this Part of the page. This method is implemented in various subclasses in different ways, for example Summary and Detail iterate through the children of 'decl' section by section, whereas Heading only formats decl itself.

visitDeclaration(*self*, *decl*) [operation on Part]

visitForward(*self*, *decl*) [operation on Part]

visitGroup(*self*, *decl*) [operation on Part]

| | |
|--|---|
| visitScope (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitModule (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitMetaModule (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitClass (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitTypedef (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitEnum (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitVariable (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitConst (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitFunction (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitOperation (<i>self</i> , <i>decl</i>) | [operation on Part] |
| formatType (<i>self</i> , <i>typeObj</i> , <i>id_holder</i>) | [operation on Part] |
| Returns a reference string for the given type object | |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's name |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's link |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's name |
| visitDependent (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be the type's name (which has no proper scope) |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Part] Adds modifiers to the formatted label of the modifier's alias |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Part] Adds the parameters to the template name in angle brackets |

visitTemplate(*self*, *type*) [operation on Part]
 Labs the template with the parameters

visitFunctionType(*self*, [operation on Part]
type)
 Labels the function type with return type, name and parameters

write_start(*self*) [operation on Part]
 Abstract method to start the output, eg table headings

writeSectionStart(*self*, [operation on Part]
heading)
 Abstract method to start a section of declaration types

writeSectionEnd(*self*, [operation on Part]
heading)
 Abstract method to end a section of declaration types

writeSectionItem(*self*, [operation on Part]
text)
 Abstract method to write the output of one formatted declaration

write_end(*self*) [operation on Part]
 Abstract method to end the output, eg close the table

Heading [class]
 parents: parent class Heading page part. Displays a header for the page – its strategies are only passed the object that the page is for; ie a Class or Module

__init__(*self*, *page*) [operation on Heading]

_init_default_formatters([operation on Heading]
self)

writeSectionItem(*self*, [operation on Heading]
text)
 Writes text and follows with a horizontal rule

process(*self*, *decl*) [operation on Heading]
 Process this Part by formatting only the given decl

Summary [class]

parents: parent class Formatting summary visitor. This formatter displays a summary for each declaration, with links to the details if there is one. All of this is controlled by the ASTFormatters.

__init__(*self*, *page*) [operation on Summary]

_init_default_formatters(
self) [operation on Summary]

set_link_detail(*self*, [operation on Summary]
boolean)
Sets link_detail flag to given value.

label(*self*, *ref*, *label*) [operation on Summary]
Override to check link_detail flag. If it's set, returns a reference instead - which will be to the detailed info

writeSectionStart(*self*, [operation on Summary]
heading)
Starts a table entity. The heading is placed in a row in a td with the class 'heading'.

writeSectionEnd(*self*, [operation on Summary]
heading)
Closes the table entity and adds a break.

writeSectionItem(*self*, [operation on Summary]
text)
Adds a table row

process(*self*, *decl*) [operation on Summary]
Print out the summaries from the given decl

Detail [class]

parents: parent class

__init__(*self*, *page*) [operation on Detail]

_init_default_formatters(
self) [operation on Detail]

writeSectionStart(*self*, [operation on Detail]
heading)
Creates a table with one row. The row has a td of class 'heading' containing the heading string

writeSectionItem(*self*, [operation on Detail]
 text)

Writes text and follows with a horizontal rule

process(*self*, *decl*) [operation on Detail]

Print out the details for the children of the given decl

Inheritance [class]

parents: parent class

__init__(*self*, *page*) [operation on Inheritance]

_init_default_formatters([operation on Inheritance]
 self)

process(*self*, *decl*) [operation on Inheritance]

Walk the hierarchy to find inherited members to print.

_process_class(*self*, [operation on Inheritance]
 clas, *names*)

Prints info for the given class, and calls
 _process_superclasses after

_short_name(*self*, [operation on Inheritance]
 decl)

_process_superclasses([operation on Inheritance]
 self, *clas*, *names*)

Iterates through the superclasses of clas and calls
 _process_clas for each

writeSectionStart([operation on Inheritance]
 self, *heading*)

Creates a table with one row. The row has a td of class
 'heading' containing the heading string

writeSectionItem([operation on Inheritance]
 self, *text*)

Adds a table row

writeSectionEnd([operation on Inheritance]
 self, *heading*)

Closes the table entity and adds a break.

CommentFormatter [module]

CommentParser, CommentFormatter and derivatives.

CommentFormatter [class]

A class that takes a Declaration and formats its comments into a string.

__init__(*self*) [operation on CommentFormatter]

format(*self*, [operation on CommentFormatter]
page, *decl*)

Formats the first comment of the given AST.Declaration. Note that the Linker.Comments.Summarizer CommentProcessor is supposed to have combined all comments first in the Linker stage.

format_summary([operation on CommentFormatter]
self, *page*, *decl*)

Formats the summary of the first comment of the given AST.Declaration. Note that the Linker.Comments.Summarizer CommentProcessor is supposed to have combined all comments first in the Linker stage.

CommentFormatterStrategy [class]

Interface class that takes a comment and formats its summary and/or detail strings.

format([operation on CommentFormatterStrategy]
self, *page*, *decl*, *text*)

Format the given comment

format_summary([operation on CommentFormatterStrategy]
self, *page*, *decl*, *summary*)

Format the given comment summary

QuoteHTML [class]

parents: parent class A formatter that quotes HTML characters like the angle brackets and the ampersand. Formats both text and summary.

format(*self*, *page*, [operation on QuoteHTML]
decl, *text*)

Replace angle brackets with HTML codes

format_summary(*self*, [operation on QuoteHTML]
page, *decl*, *text*)

Replace angle brackets with HTML codes

JavadocFormatter [class]

parents: parent class A formatter that formats comments similar to Javadoc @tags

__init__(*self*) [operation on JavadocFormatter]
Create regex objects for regexps

extract(*self*, [operation on JavadocFormatter]
regexp, *str*)
Extracts all matches of the regexp from the text. The MatchObjects are returned in a list

format(*self*, [operation on JavadocFormatter]
page, *decl*, *text*)
Format any @tags in the text, and any @tags stored by the JavaTags CommentProcessor in the Linker stage.

format_inline_see([operation on JavadocFormatter]
self, *page*, *decl*, *text*)
Formats inline tags in the text

format_params([operation on JavadocFormatter]
self, *param_tags*)
Formats a list of (param, description) tags

format_attrs([operation on JavadocFormatter]
self, *attr_tags*)
Formats a list of (attr, description) tags

format_return([operation on JavadocFormatter]
self, *return_tag*)
Formats a since description string

format_see(*self*, [operation on JavadocFormatter]
page, *see_tags*, *decl*)
Formats a list of (ref,description) tags

find_link(*self*, [operation on JavadocFormatter]
page, *ref*, *decl*)
Given a "reference" and a declaration, returns a HTML link. Various methods are tried to resolve the reference. First the parameters are taken off, then we try to split the ref using '.' or '::.'. The params are added back, and then we try to match this scoped name against the current scope. If that fails, then we recursively try enclosing scopes.

_find_link_at(*self*, *ref*, *scope*) [operation on JavadocFormatter]

_find_method_entry(*self*, *name*, *scope*) [operation on JavadocFormatter]

Tries to find a TOC entry for a method adjacent to decl. The enclosing scope is found using the types dictionary, and the realname()'s of all the functions compared to ref.

QtDocFormatter [class]

parents: parent class A formatter that uses Qt-style doc tags.

__init__(*self*) [operation on QtDocFormatter]

parseText(*self*, *str*, *decl*) [operation on QtDocFormatter]

format_see(*self*, *see_tags*, *decl*) [operation on QtDocFormatter]

Formats a list of (ref,description) tags

SectionFormatter [class]

parents: parent class A test formatter

__init__(*self*) [operation on SectionFormatter]

format(*self*, *page*, *decl*, *text*) [operation on SectionFormatter]

core [module]

Core module for the HTML Formatter.

This module is the first to be loaded, and it creates the global 'core.config' object before creating any pages. It also handles the command line parsing for this module, and coordinates the actual output generation.

sort(*list*) [function]
Utility func to sort and return the given list

old_reference(*name*, *scope*, *label*, ***keys*) [function]
Utility method to insert a reference to a name.

compile_glob(globstr) [function]
 Returns a compiled regular expression for the given glob string. A glob string is something like "*.?pp" which gets translated into "^.*\?.pp\$".

usage() [function]
 Print usage to stdout

__parseArgs(args, config_obj) [function]

format(args, ast, config_obj) [function]

configure_for_gui(ast, config_obj) [function]

Config [class]

Central configuration repository for HTML formatter. This class holds references to the current formatters, tocs, etc.

__init__(self) [operation on Config]
 Constructor - initialise objects to None.

fillDefaults(self) [operation on Config]
 Fill in empty options with defaults

use_config(self, obj) [operation on Config]
 Extracts useful attributes from 'obj' and stores them. The object itself is also stored as config.obj

_config_pages(self, pages) [operation on Config]
 Configures from the given list of pages

_config_sorter(self, sorter) [operation on Config]

_config_datadir(self, datadir) [operation on Config]

_config_output_dir(self, output_dir) [operation on Config]

_config_stylesheet(self, stylesheet) [operation on Config]

_config_stylesheet_file(self, stylesheet_file) [operation on Config]

`_config_comment_formatters(` **█** [operation on Config]
 self, comment_formatters)

`_config_toc_in(` *self,* [operation on Config]
 toc_in)

`_config_default_toc(` *self,* [operation on Config]
 page)

`_config_toc_out(` *self,* [operation on Config]
 toc_out)

`_config_tree_formatter(` [operation on Config]
 self, tree_class)

`_config_file_layout(` *self,* [operation on Config]
 layout)

`_config_base_dir(` *self,* [operation on Config]
 dir)

`_config_start_dir(` *self,* [operation on Config]
 dir)

`_config_structs_as_classes(` [operation on Config]
 self, yesno)

`_config_exclude_globs(` [operation on Config]
 self, globs)

`_config_page_format(` *self,* [operation on Config]
 page_format)

`set_contents_page(` *self,* [operation on Config]
 page)

Call this method to set the contents page. First come first served – whatever module the user puts first in the list that sets this is it. This is the frame in the top-left if you use the default frameset.

`set_index_page(` *self,* [operation on Config]
 page)

Call this method to set the index page. First come first served – whatever module the user puts first in the list that sets this is it. This is the frame on the left if you use the default frameset.

set_main_page(*self*, *page*) [operation on **Config**]

Call this method to set the main index.html page. First come first served – whatever module the user puts first in the list that sets this is it.

set_using_module_index(*self*) [operation on **Config**]

Sets the using_module_index flag. This will cause the an intermediate level of links intended to go in the left frame.

Struct [class]

Dummy class. Initialise with keyword args.

__init__(*self*, **keys**)** [operation on **Struct**]

DeclStyle [class]

This class just maintains a mapping from declaration to display style. The style is an enumeration, possible values being: SUMMARY (only display a summary for this declaration), DETAIL (summary and detailed info), INLINE (summary and detailed info, where detailed info is an inline version of the declaration even if it's a class, etc.)

Upon initiation, an instance of this class installs itself in the config object as "decl_style".

__init__(*self*) [operation on **DeclStyle**]

style_of(*self*, *decl*) [operation on **DeclStyle**]
Returns the style of the given decl

PageManager [class]

This class manages and coordinates the various pages. The user adds pages by passing their class object to the addPage method. Pages should be derived from Page.Page, and their constructors may want to call the addRootPage method of the PageManager object to register a name and link that is listed along with other root or top-level pages.

__init__(*self*) [operation on **PageManager**]

getPage(*self*, *name*) [operation on **PageManager**]
Returns the Page with the given name

globalScope(*self*) [operation on PageManager]
Return the global scope

calculateStart(*self*, *root*, *namespace*) [operation on PageManager]
Calculates the start scope using the 'namespace' config var

addPage(*self*, *pageClass*) [operation on PageManager]
Add a page of the given class. An instance is created and stored, and its root() method is called and the name,link tuple stored if None isn't returned.

addRootPage(*self*, *file*, *label*, *target*, *visibility*) [operation on PageManager]
Adds a named link to the list of root pages. Called from the constructors of Page objects. The root pages are displayed at the top of every page, depending on their visibility (higher = more visible).

formatHeader(*self*, *origin*, *visibility*) [operation on PageManager]
Formats the list of root pages to HTML. The origin specifies the generated page itself (which shouldn't be linked), such that the relative links can be generated. Only root pages of 'visibility' or above are included.

process(*self*, *root*) [operation on PageManager]
Create all pages from the start Scope, derived from the root Scope

_loadPages(*self*) [operation on PageManager]
Loads the page objects from the config.pages list. Each element is either a string or a tuple of two strings. One string means load the named module and look for a 'htmlPageClass' attribute in it. A tuple of two strings means load the module from the first string, and look for an attribute using the second string.

register_filename(*self*, *filename*, *page*, *scope*) [operation on PageManager]
Registers a file for later production. The first page to register the filename gets to keep it.

filename_info(*self*, [operation on PageManager]
filename)

Returns information about a registered file, as a (page,scope) pair. Will return None if the filename isn't registered.

DirBrowse [module]

DirBrowse [class]

parents: parent class A page that shows the entire contents of directories, in a form similar to LXR.

__init__(*self*, *manager*) [operation on DirBrowse]

filename(*self*) [operation on DirBrowse]
 since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on DirBrowse]
 since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

filename_for_dir(*self*, [operation on DirBrowse]
dir)
 Returns the output filename for the given input directory

register(*self*) [operation on DirBrowse]
 Registers a page for each file in the hierarchy

register_filenames([operation on DirBrowse]
self, *start*)
 Registers a page for every directory

process(*self*, *start*) [operation on DirBrowse]
 Recursively visit each directory below the base path given in the config.

process_dir(*self*, *path*) [operation on DirBrowse]
 Process a directory, producing an output page for it

doxygen [module]

Doxygen emulation mode classes.

DOScopeSorter [class]

parents: parent class

_section_of(*self*, [operation on DOScopeSorter]
decl)

sort_section_names([operation on DOScopeSorter]
self)
Sort by a known order..

DOSummaryAST [class]

parents: parent class

formatEnum(*self*, [operation on DOSummaryAST]
decl)
(enum, enum { list of enumerator names })

DOSummaryCommenter [class]

parents: parent class Adds summary comments to all declarations

formatDeclaration([operation on DOSummaryCommenter]
self, *decl*)

DODetailAST [class]

parents: parent class

formatFunction(*self*, [operation on DODetailAST]
decl)

PreDivFormatter [class]

parents: parent class

formatDeclaration([operation on PreDivFormatter]
self, *decl*)

PostDivFormatter [class]

parents: parent class

formatDeclaration([operation on PostDivFormatter]
self, *decl*)

PreSummaryDiv [class]

parents: parent class

formatDeclaration([operation on PreSummaryDiv]
self, decl)

PostSummaryDiv [class]
 parents: parent class

formatDeclaration([operation on PostSummaryDiv]
self, decl)

DOSummary [class]
 parents: parent class

old_init_formatters([operation on DOSummary]
self)

write_start(*self*) [operation on DOSummary]

writeSectionStart([operation on DOSummary]
self, heading)

writeSectionEnd(*self,* [operation on DOSummary]
heading)

write_end(*self*) [operation on DOSummary]
 Closes the table entity and adds a break.

writeSectionItem_Foo([operation on DOSummary]
self, type, name)
 Adds a table row with one or two data elements. If
 type is None then there is only one td with a colspan
 of 2.

DODetail [class]
 parents: parent class

old_init_formatters([operation on DODetail]
self)

writeSectionEnd(*self,* [operation on DODetail]
heading)

writeSectionItem_Foo([operation on DODetail]
self, text)
 Joins text1 and text2

FileDetails [module]

FileDetails [class]

parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

`__init__(self, manager)` [operation on FileDetails]

`filename(self)` [operation on FileDetails]
since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

`title(self)` [operation on FileDetails]
since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

`register_filenames(self, start)` [operation on FileDetails]
Registers a page for each file indexed

`process(self, start)` [operation on FileDetails]
Creates a page for each file using process_scope

`process_scope(self, filename, file)` [operation on FileDetails]
Creates a page for the given file. The page is just an index, containing a list of declarations.

FileIndexer [module]**FileIndexer** [class]

parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

`__init__(self, manager)` [operation on FileIndexer]

filename(*self*) [operation on `FileIndexer`]
 since `FileTree` generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on `FileIndexer`]
 since `FileTree` generates a whole file hierarchy, this method returns the current title, which may change over the lifetime of this object

register_filenames(*self*, *start*) [operation on `FileIndexer`]
 Registers a page for each file indexed

process(*self*, *start*) [operation on `FileIndexer`]
 Creates a page for each file using `process_scope`

process_scope(*self*, *filename*, *file*) [operation on `FileIndexer`]
 Creates a page for the given file. The page is just an index, containing a list of declarations.

FileLayout [module]
 FileLayout module. This is the base class for the FileLayout interface. The default implementation stores everything in the same directory.

FileLayout [class]
 parents: parent class Base class for naming files. You may derive from this class and reimplement any methods you like. The default implementation stores everything in the same directory (specified by `-o`).

__init__(*self*) [operation on `FileLayout`]

copyFile(*self*, *orig_name*, *new_name*) [operation on `FileLayout`]
 Copies file if newer. The file named by `orig_name` is compared to `new_name`, and if newer or `new_name` doesn't exist, it is copied.

_checkMain(*self*, *filename*) [operation on `FileLayout`]
 Checks whether the given filename is the main index page or not. If it is, then it returns the filename from `nameOfIndex()`, else it returns it unchanged

nameOfScope(*self*, [operation on FileLayout]
scope)

Return the filename of a scoped name (class or module). The default implementation is to join the names with '-' and append ".html" Additionally, special characters are Util.quoted in URL-style

_stripFilename(*self*, [operation on FileLayout]
file)

nameOfFileIndex([operation on FileLayout]
self, *file*)

Return the filename for the index of an input file. The base_dir config option is used. Default implementation is to join the path with '-', prepend "_file-" and append ".html"

nameOfFileSource([operation on FileLayout]
self, *file*)

Return the filename for the source of an input file. The base_dir config option is used. Default implementation is to join the path with '-', prepend "_source-" and append ".html"

nameOfFileDetails([operation on FileLayout]
self, *file*)

Return the filename for the details of an input file. The base_dir config option is used. Default implementation is to join the path with '-', prepend "_filedetail-" and append ".html"

nameOfIndex(*self*) [operation on FileLayout]

Return the name of the main index file. Default is index.html

nameOfSpecial(*self*, [operation on FileLayout]
name)

Return the name of a special file (tree, etc). Default is _name.html

nameOfScopedSpecial([operation on FileLayout]
self, *name*, *scope*, *ext*)

Return the name of a special type of scope file. Default is to join the scope with '-' and prepend '-' + name

nameOfModuleTree(*self*) [operation on FileLayout]

Return the name of the module tree index. Default is `_modules.html`

nameOfModuleIndex(*self*, *scope*) [operation on FileLayout]

Return the name of the index of the given module. Default is to join the name with `'-'`, prepend `"_module-"` and append `".html"`

link(*self*, *decl*) [operation on FileLayout]

Create a link to the named declaration. This method may have to deal with the directory layout.

NestedFileLayout [class]

parents: parent class generates a structured file system instead of a flat one

__init__(*self*) [operation on NestedFileLayout]

nameOfScope(*self*, *scope*) [operation on NestedFileLayout]

Return the filename of a scoped name (class or module). One subdirectory per scope

nameOfFileIndex(*self*, *file*) [operation on NestedFileLayout]

Return the filename for the index of an input file. The `base_dir` config option is used. Default implementation is to join the path with `'-'`, prepend `"_file-"` and append `".html"`

nameOfFileSource(*self*, *file*) [operation on NestedFileLayout]

Return the filename for the source of an input file. The `base_dir` config option is used. Default implementation is to join the path with `'-'`, prepend `"_source-"` and append `".html"`

nameOfFileDetails(*self*, *file*) [operation on NestedFileLayout]

Return the filename for the details of an input file. The `base_dir` config option is used. Returns the filename nested in the FileDetails directory and with `.html` appended.

nameOfIndex(*self*) [operation on NestedFileLayout]

Return the name of the main index file.

nameOfSpecial(*self, name*) [operation on NestedFileLayout]

Return the name of a special file (tree, etc).

nameOfScopedSpecial(*self, name, scope, ext*) [operation on NestedFileLayout]

Return the name of a special type of scope file

nameOfModuleTree(*self*) [operation on NestedFileLayout]

Return the name of the module tree index

nameOfModuleIndex(*self, scope*) [operation on NestedFileLayout]

Return the name of the index of the given module

FileListing [module]

FileListing [class]

parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

__init__(*self, manager*) [operation on FileListing]

filename(*self*) [operation on FileListing]
Returns the filename

title(*self*) [operation on FileListing]
Returns the title

register(*self*) [operation on FileListing]
Registers this page for the top-left frame

register_filenames(*self, start*) [operation on FileListing]
Registers a page for each file indexed

process(*self*, *start*) [operation on FileListing]
Creates the listing using the recursive processFileTreeNode method

_node_sorter(*self*, *a*, *b*) [operation on FileListing]
Compares file nodes *a* and *b* depending on whether they are leaves or not

processFileTreeNode(*self*, *node*) [operation on FileListing]
Creates a portion of the tree for the given file node. This method assumes that the file is already in progress, and just appends to it. This method is recursive, calling itself for each child of node (file or directory).

FileSource [module]

FileSource [class]

parents: parent class A module for creating a page for each file with hyperlinked source

__init__(*self*, *manager*) [operation on FileSource]

filename(*self*) [operation on FileSource]
since FileSource generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on FileSource]
since FileSource generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

process(*self*, *start*) [operation on FileSource]
Creates a page for every file

register_filenames(*self*, *start*) [operation on FileSource]
Registers a page for every source file

process_node(*self*, *file*) [operation on FileSource]
Creates a page for the given file

FileTreeJS [module]

FileTree [class]

parents: parent class

__init__(*self*, *manager*) [operation on FileTree]

filename(*self*) [operation on FileTree]
since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on FileTree]
since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

process(*self*, *start*) [operation on FileTree]

_node_sorter(*self*, *a*,
b) [operation on FileTree]

processFileTreeNode([operation on FileTree]
self, *node*)

processFileTreeNodePage([operation on FileTree]
self, *node*)

FileTree [module]

FileTree [class]

parents: parent class A page which wraps the three pages now used for creating file-related info in the doco. For backwards compatibility only. The three new pages which should be used are FileListing, FileIndexer and FileDetails.

__init__(*self*, *manager*) [operation on FileTree]

register(*self*) [operation on FileTree]

register_filenames(*self*, [operation on FileTree]
start)

process(*self*, *start*) [operation on FileTree]

FormatStrategy [module]

AST Formatting Strategies.

This module contains all the builtin formatting strategies used to make the main scope pages. These strategies are used by Strategy.Summary/Detail to create the entries for each declaration - each has a list of strategies that it calls in turn to generate a HTML fragment for each declaration. So for example, a link to source code is added just by adding a Strategy (eg: FilePages) that outputs the link to the source. This can be done without changing the strategies that generate the declaration name, type or comments.

Strategy [class]

Generates HTML fragment for a declaration. Multiple strategies are combined to generate the output for a single declaration, allowing the user to customise the output by choosing a set of strategies. This follows the Strategy design pattern.

The key concept of this class is the format* methods. Any class derived from Strategy that overrides one of the format methods will have that method called by the Summary and Detail formatters when they visit that AST type. Summary and Detail maintain a list of Strategies, and a list for each AST type.

For example, when Strategy.Summary visits a Function object, it calls the formatFunction method on all Strategies registered with SummaryFormatter that implemented that method. Each of these format methods returns a string, which may contain a TD tag to create a new column.

An important point to note is that only Strategies which override a particular format method are called - if that format method is not overridden then it is not called for that declaration type.

__init__(self, formatter) [operation on Strategy]

Store formatter as self.formatter. The formatter is either a SummaryFormatter or DetailFormatter, and is used for things like reference() and label() calls. Local references to the formatter's reference and label methods are stored in self for more efficient use of them.

formatModifiers(self, modifiers) [operation on Strategy]

Returns a HTML string from the given list of string modifiers. The modifiers are enclosed in 'keyword' spans.

| | |
|---|-------------------------|
| formatDeclaration (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatForward (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatGroup (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatScope (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatModule (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatMetaModule (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatClass (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatTypedef (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatEnum (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatVariable (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatConst (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatFunction (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatOperation (<i>self</i> , <i>decl</i>) | [operation on Strategy] |

BaseAST [class]

parents: parent class Base class for SummaryAST and DetailAST.

The two classes SummaryAST and DetailAST are actually very similar in operation, and so most of their methods are defined here. Both of them print out the definition of the declarations, including type, parameters, etc. Some things such as exception specifications are only printed out in the detailed version.

formatParameters(*self*, [operation on BaseAST]
 parameters)

Returns formatted string for the given parameter list

formatDeclaration(*self*, [operation on BaseAST]
 decl)

The default is to return no type and just the declarations name for the name

formatForward(*self*, [operation on BaseAST]
 decl)

formatGroup(*self*, *decl*) [operation on BaseAST]

formatScope(*self*, *decl*) [operation on BaseAST]

Scopes have their own pages, so return a reference to it

formatModule(*self*, [operation on BaseAST]
 decl)

formatMetaModule([operation on BaseAST]
 self, *decl*)

formatClass(*self*, *decl*) [operation on BaseAST]

formatTypedef(*self*, [operation on BaseAST]
 decl)

(typedef type, typedef name)

formatEnumerator(*self*, [operation on BaseAST]
 decl)

This is only called by formatEnum

formatEnum(*self*, *decl*) [operation on BaseAST]
(enum name, list of enumerator names)

formatVariable(*self*, [operation on BaseAST]
 decl)

formatConst(*self*, *decl*) [operation on BaseAST]
(const type, const name = const value)

formatFunction(*self*, [operation on BaseAST]
 decl)

(return type, func + params + formatFunctionExceptions)

formatOperation(*self*, [operation on BaseAST]
decl)

formatParameter(*self*, [operation on BaseAST]
parameter)

Returns one string for the given parameter

SummaryAST [class]

parents: parent class Derives from BaseStrategy to provide summary-specific methods. Currently the only one is format-OperationExceptions

formatOperationExceptions([operation on SummaryAST]
self, *oper*)

Returns a reference to the detail if there are any exceptions.

Default [class]

parents: parent class A base AST strategy that calls format-Declaration for all types

formatForward(*self*, [operation on Default]
decl)

formatGroup(*self*, *decl*) [operation on Default]

formatScope(*self*, *decl*) [operation on Default]

formatModule(*self*, [operation on Default]
decl)

formatMetaModule([operation on Default]
self, *decl*)

formatClass(*self*, *decl*) [operation on Default]

formatTypedef(*self*, [operation on Default]
decl)

formatEnum(*self*, *decl*) [operation on Default]

formatVariable(*self*, [operation on Default]
decl)

formatConst(*self*, *decl*) [operation on Default]

formatFunction(*self*, [operation on Default]
decl)

formatOperation(*self*, [operation on Default]
decl)

SummaryCommenter [class]

parents: parent class Adds summary comments to all declarations

formatDeclaration([operation on SummaryCommenter]
self, *decl*)

formatGroup([operation on SummaryCommenter]
self, *decl*)

Override for group to use the div version of commenting, and no
 before

SourceLinker [class]

parents: parent class Adds a link to the decl on the file page to all declarations

formatDeclaration([operation on SourceLinker]
self, *decl*)

XRefLinker [class]

parents: parent class Adds an xref link to all declarations

__init__(*self*, [operation on XRefLinker]
formatter)

formatDeclaration([operation on XRefLinker]
self, *decl*)

Heading [class]

parents: parent class Formats the top of a page - it is passed only the Declaration that the page is for (a Module or Class).

formatName(*self*, [operation on Heading]
scoped_name)

Formats a reference to each parent scope

formatNameInNamespace([operation on Heading]
self, *scoped_name*)

Formats a reference to each parent scope, starting at the first non-module scope

formatNamespaceOfName(`self, scoped_name`) [operation on Heading]

Formats a reference to each parent scope and this one

formatModule(`self, module`) [operation on Heading]

Formats the module by linking to each parent scope in the name

formatMetaModule(`self, module`) [operation on Heading]

Calls `formatModule`

formatClass(`self, clas`) [operation on Heading]

Formats the class by linking to each parent scope in the name

formatParameter(`self, parameter`) [operation on Heading]

Returns one string for the given parameter

DetailAST [class]

parents: parent class Derives `BaseAST` to provide detail-specific AST formatting.

formatOperationExceptions(`self, oper`) [operation on DetailAST]

Prints out the full exception spec

formatEnum(`self, enum`) [operation on DetailAST]

formatEnumerator(`self, enumerator`) [operation on DetailAST]

DetailCommenter [class]

parents: parent class Adds summary comments to all declarations

formatDeclaration(`self, decl`) [operation on DetailCommenter]

ClassHierarchySimple [class]

parents: parent class Prints a simple text hierarchy for classes

formatInheritance(*self*, *inheritance*) [operation on ClassHierarchySimple]

formatClass(*self*, *clas*) [operation on ClassHierarchySimple]

ClassHierarchyGraph [class]

parents: parent class Prints a graphical hierarchy for classes, using the Dot formatter.

formatClass(*self*, *clas*) [operation on ClassHierarchyGraph]

Inheritance [class]

parents: parent class Prints just the name of each declaration, with a link to its doc

formatDeclaration(*self*, *decl*, *label*) [operation on Inheritance]

formatFunction(*self*, *decl*) [operation on Inheritance]

formatOperation(*self*, *decl*) [operation on Inheritance]

FramesIndex [module]

FramesIndex [class]

parents: parent class A class that creates an index with frames

__init__(*self*, *manager*) [operation on FramesIndex]

filename(*self*) [operation on FramesIndex]

title(*self*) [operation on FramesIndex]

register(*self*) [operation on FramesIndex]

process(*self*, *start*) [operation on FramesIndex]
Creates a frames index file

InheritanceGraph [module]

find_common_name(*graph*) [function]

ToDecl [class]

parents: parent class

`--call--(self, name)` [operation on ToDecl]`visitBaseType(self, type)` [operation on ToDecl]`visitUnknown(self, type)` [operation on ToDecl]`visitDeclared(self, type)` [operation on ToDecl]`visitModifier(self, type)` [operation on ToDecl]`visitArray(self, type)` [operation on ToDecl]`visitTemplate(self, type)` [operation on ToDecl]`visitParametrized(self, type)` [operation on ToDecl]`visitFunctionType(self, type)` [operation on ToDecl]**InheritanceGraph** [class]

parents: parent class

`--init--(self, manager)` [operation on InheritanceGraph]`filename(self)` [operation on InheritanceGraph]`title(self)` [operation on InheritanceGraph]`register(self)` [operation on InheritanceGraph]
Registers this page with the manager`consolidate(self, graphs)` [operation on InheritanceGraph]
Consolidates small graphs into larger ones`process(self, start)` [operation on InheritanceGraph]
Creates a file with the inheritance graph**InheritanceTree** [module]

InheritanceTree [class]

parents: parent class

__init__(*self*, [operation on InheritanceTree]
manager)**filename**(*self*) [operation on InheritanceTree]**title**(*self*) [operation on InheritanceTree]**register**(*self*) [operation on InheritanceTree]**process**(*self*, [operation on InheritanceTree]
start)

Creates a file with the inheritance tree

processClassInheritance([operation on InheritanceTree]
self, *args*)**JSTree** [module]**JSTree** [class]parents: parent class Page that makes Javascript trees. The trees have expanding and collapsing nodes. call `js_init()` with the button images and default open/close policy during process**__init__**(*self*, *manager*) [operation on JSTree]**getId**(*self*) [operation on JSTree]**js_init**(*self*, *open_img*, [operation on JSTree]
close_img, *leaf_img*, *base*, *default_open*)

Initialise the JSTree page. This method copies the files to the output directory and stores the values given.

start_file(*self*) [operation on JSTree]
Overrides `start_file` to add the javascript**formatImage**(*self*, *id*, [operation on JSTree]
filename, *alt_text*)

Returns the image element for the given image

writeLeaf(*self*, *item_text*) [operation on JSTree]

Write a leaf node to the output at the current tree level.

writeNodeStart(*self*, [operation on JSTree]
 item_text)

Write a non-leaf node to the output at the current tree level, and start a new level.

writeNodeEnd(*self*) [operation on JSTree]

Finish a non-leaf node, and close the current tree level.

ModuleIndexer [module]

ModuleIndexer [class]

parents: parent class A module for indexing AST.Modules. Each module gets its own page with a list of nested scope declarations with comments. It is intended to go in the left frame...

__init__(*self*, [operation on ModuleIndexer]
 manager)

filename(*self*) [operation on ModuleIndexer]

title(*self*) [operation on ModuleIndexer]

register(*self*) [operation on ModuleIndexer]
 Register first page as index page

process(*self*, [operation on ModuleIndexer]
 start)

Creates indexes for all modules

_makePageHeading([operation on ModuleIndexer]
 self, *ns*)

Creates a HTML fragment which becomes the name at the top of the index page. This may be overridden, but the default is (now) to make a linked fully scoped name, where each scope has a link to the relevant index.

processNamespaceIndex([operation on ModuleIndexer]
 self, *ns*)

Index one module

ModuleListingJS [module]

ModuleListingJS [class]

parents: parent class Create an index of all modules with JS. The JS allows the user to expand/collapse sections of the tree!

`--init--(self, manager)` [operation on ModuleListingJS]

`_init_page(self)` [operation on ModuleListingJS]
Sets `_filename` and registers the page with the manager

`filename(self)` [operation on ModuleListingJS]

`title(self)` [operation on ModuleListingJS]

`process(self, start)` [operation on ModuleListingJS]
Create a page with an index of all modules

`_child_filter(self, child)` [operation on ModuleListingJS]

`_link_href(self, ns)` [operation on ModuleListingJS]

`get_children(self, decl)` [operation on ModuleListingJS]

`indexModule(self, ns, rel_scope)` [operation on ModuleListingJS]
Write a link for this module and recursively visit child modules.

ModuleListing [module]

ModuleListing [class]

parents: parent class Create an index of all modules with JS. The JS allows the user to expand/collapse sections of the tree!

`--init--(self, manager)` [operation on ModuleListing]

`filename(self)` [operation on ModuleListing]

`title(self)` [operation on ModuleListing]

`register(self)` [operation on ModuleListing]
registers the page with the manager for the 'contents' (top left) frame

process(*self*, *start*) [operation on ModuleListing]
 Create a page with an index of all modules

_child_filter(*self*, *child*) [operation on ModuleListing]
 Returns true if the given child declaration is to be included

_link_href(*self*, *ns*) [operation on ModuleListing]
 Returns the link to the given declaration

_get_children(*self*, *decl*) [operation on ModuleListing]
 Returns the children of the given declaration

indexModule(*self*, *ns*, *rel_scope*) [operation on ModuleListing]
 Write a link for this module and recursively visit child modules.

NameIndex [module]

NameIndex [class]

parents: parent class
 Creates an index of all names on one page in alphabetical order

__init__(*self*, *manager*) [operation on NameIndex]

filename(*self*) [operation on NameIndex]

title(*self*) [operation on NameIndex]

register(*self*) [operation on NameIndex]

process(*self*, *start*) [operation on NameIndex]
 Creates the page. It is created as a list of tables, one for each letter. The tables have a number of columns, which is 2 by default. `_processItem` is called for each item in the dictionary.

_makeDict(*self*) [operation on NameIndex]
 Returns a dictionary of items. The keys of the dictionary are the headings - the first letter of the name. The values are each a sorted list of items with that first letter.

_processItem(*self*, [operation on NameIndex]
type)

Process the given name for output

Page [module]

Page base class, contains base functionality and common interface for all Pages.

PageFormat [class]

Default and base class for formatting a page layout. The PageFormat class basically defines the HTML used at the start and end of the page. The default creates an XHTML compliant header and footer with a proper title, and link to the stylesheet.

__init__(*self*) [operation on PageFormat]

set_prefix(*self*, [operation on PageFormat]
prefix)

Sets the prefix to use to correctly reference files in the document root directory.

stylesheet(*self*) [operation on PageFormat]

Returns the relative filename of the stylesheet to use. The stylesheet specified in the user's config is copied into the output directory. If this page is not in the same directory, the url returned from this function will have the appropriate number of '..'s added.

prefix(*self*) [operation on PageFormat]

Returns the prefix to use to correctly reference files in the document root directory. This will only ever not be "" if you are using the NestedFileLayout, in which case it will be "" or '../' or '../..' etc as appropriate.

page_header(*self*, *os*, [operation on PageFormat]
title, *body*, *headextra*)

Called to output the page header to the given output stream.

page_footer(*self*, *os*, [operation on PageFormat]
body)

Called to output the page footer to the given output stream.

TemplatePageFormat [class]

parents: parent class PageFormat subclass that uses a template file to define the HTML header and footer for each page.

__init__(*self*) [operation on TemplatePageFormat]

load_file(*self*) [operation on TemplatePageFormat]
Loads and parses the template file

write(*self*, *os*, *text*) [operation on TemplatePageFormat]
Writes the text to the output stream, replacing @PREFIX@ with the prefix for this file

page_header(*self*, *os*, *title*, *body*, *headextra*) [operation on TemplatePageFormat]
Formats the header using the template file

page_footer(*self*, *os*, *body*) [operation on TemplatePageFormat]
Formats the footer using the template file

Page [class]

Base class for a Page. The base class provides a common interface, and also handles common operations such as opening the file, and delegating the page formatting to a strategy class.

__init__(*self*, *manager*) [operation on Page]
Constructor, loads the formatting class.

filename(*self*) [operation on Page]
Polymorphic method returning the filename associated with the page

title(*self*) [operation on Page]
Polymorphic method returning the title associated with the page

os(*self*) [operation on Page]
Returns the output stream opened with start_file

write(*self*, *str*) [operation on Page]
Writes the given string to the currently opened file

register(*self*) [operation on Page]
Registers this Page class with the PageManager. This method is abstract - derived Pages should implement it to call the appropriate methods in PageManager if they need to. This method is called after construction.

register_filenames(*self*, *start*) [operation on Page]
Registers filenames for each file this Page will generate, given the starting Scope.

get_toc(*self*, *start*) [operation on Page]
Retrieves the TOC for this page. This method assumes that the page generates info for the the whole AST, which could be the ScopePages, the FilePages (source code) or the XRefPages (cross reference info). The default implementation returns None. Start is the declaration to start processing from, which could be the global namespace.

process(*self*, *start*) [operation on Page]
Process the given Scope recursively. This is the method which is called to actually create the files, so you probably want to override it ;)

process_scope(*self*, *scope*) [operation on Page]
Process just the given scope

open_file(*self*) [operation on Page]
Returns a new output stream. This template method is for internal use only, but may be overridden in derived classes. The default joins config.basename and self.filename() and uses Util.open()

close_file(*self*) [operation on Page]
Closes the internal output stream. This template method is for internal use only, but may be overridden in derived classes.

start_file(*self*, *body*, *headextra*) [operation on Page]
Start a new file with given filename, title and body. This method opens a file for writing, and writes the html header crap at the top. You must specify a title, which is prepended with the project name. The

body argument is optional, and it is preferred to use stylesheets for that sort of stuff. You may want to put an `onLoad` handler in it though in which case that's the place to do it. The opened file is stored and can be accessed using the `os()` method.

end_file(*self*, *body*) [operation on Page]
 Close the file using given close body tag. The default is just a close body tag, but if you specify " then nothing will be written (useful for a frames page)

reference(*self*, *name*, *scope*, *label*, *keys*)** [operation on Page]
 Returns a reference to the given name. The name is a scoped name, and the optional label is an alternative name to use as the link text. The name is looked up in the TOC so the link may not be local. The optional keys are appended as attributes to the A tag.

BufferPage [class]

parents: parent class A page that writes to a string buffer.

_take_control(*self*) [operation on BufferPage]

open_file(*self*) [operation on BufferPage]
 Returns a new StringIO

close_file(*self*) [operation on BufferPage]
 Does nothing.

get_buffer(*self*) [operation on BufferPage]
 Returns the page as a string, then deletes the internal buffer

RawFilePages [module]

RawFilePages [class]

parents: parent class A module for creating a page for each file with hyperlinked source

__init__(*self*, *manager*) [operation on RawFilePages]

filename(*self*) [operation on RawFilePages]
 since RawFilePages generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on RawFilePages]
 since RawFilePages generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

_get_files(*self*) [operation on RawFilePages]
 Returns a list of (path, output_filename) for each file

process(*self*, *start*) [operation on RawFilePages]
 Creates a page for every file

register_filenames(*self*, *start*) [operation on RawFilePages]
 Registers a page for every file

process_file(*self*, *original*, *filename*) [operation on RawFilePages]
 Creates a page for the given file

ScopePages [module]

ScopePages [class]

parents: parent class A module for creating a page for each Scope with summaries and details. This module is highly modular, using the classes from ASTFormatter to do the actual formatting. The classes to use may be controlled via the config script, resulting in a very configurable output.

__init__(*self*, *manager*) [operation on ScopePages]

_get_parts(*self*) [operation on ScopePages]
 Loads the list of parts from config

get_toc(*self*, *start*) [operation on ScopePages]
 Returns the TOC for the whole AST starting at start

filename(*self*) [operation on ScopePages]
 since ScopePages generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on ScopePages]
 since ScopePages generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

scope(*self*) [operation on ScopePages]
return the current scope processed by this object

process(*self*, *start*) [operation on ScopePages]
Creates a page for every Scope

register_filenames(*self*, *start*) [operation on ScopePages]
Registers a page for every Scope

process_scope(*self*, *ns*) [operation on ScopePages]
Creates a page for the given scope

end_file(*self*) [operation on ScopePages]
Overrides end_file to provide synopsis logo

ScopeSorter [module]
Scope sorting class module. This module contains the class for sorting Scopes.

compare_sections(*a*, *b*) [function]

ScopeSorter [class]
A class that takes a scope and sorts its children by type. To use it call set_scope, then access the sorted list by the other methods.

__init__(*self*, *scope*) [operation on ScopeSorter]
Optional scope starts using that AST.Scope

set_scope(*self*, *scope*) [operation on ScopeSorter]
Sort children of given scope

_add_decl(*self*, *decl*, *name*, *section*) [operation on ScopeSorter]
Adds the given decl with given name and section to the internal data

_section_of(*self*, *decl*) [operation on ScopeSorter]

_sort_sections(*self*) [operation on ScopeSorter]

sort_section_names(*self*) [operation on ScopeSorter]
Sorts sections names if they need it

_set_section_names(*self*, *sections*) [operation on ScopeSorter]

_handle_group(*self*, *group*) [operation on ScopeSorter]
Handles a group

sort_sections(*self*) [operation on ScopeSorter]
Sorts the children of all sections, if they need it

child(*self*, *name*) [operation on ScopeSorter]
Returns the child with the given name. Throws Key-Error if not found.

sections(*self*) [operation on ScopeSorter]
Returns a list of available section names

children(*self*, *section*) [operation on ScopeSorter]
Returns list of children in given section, or all children

Tags [module]
HTML Tag generation utilities. You will probably find it easiest to import * from this module.

k2a(*keys*) [function]
Convert a name/value dict to a string of attributes

rel(*frm*, *to*) [function]
Find link to to relative to frm

href(*_ref*, *_label*, ***keys*) [function]
Return a href to 'ref' with name 'label' and attributes

name(*ref*, *label*) [function]
Return a name anchor with given reference and label

span(*clas*, *body*) [function]
Wrap the body in a span of the given class

div(*clas*, *body*) [function]
Wrap the body in a div of the given class

entity(*_type*, *body*, ****keys**) [function]
Wrap the body in a tag of given type and attributes

solotag(*_type*, ****keys**) [function]
Create a solo tag (no close tag) of given type and attributes

desc(*text*) [function]
Create a description div for the given text

anglebrackets(*text*) [function]
Replace angle brackets with HTML codes

replace_spaces(*text*) [function]
Replaces spaces in the given string with sequences.
Does NOT replace spaces inside tags

TreeFormatterJS [module]

TreeFormatterJS [class]

parents: parent class Javascript trees. The trees have expanding and collapsing nodes. call `js.init()` with the button images and default open/close policy during process

__init__(*self*, [operation on TreeFormatterJS]
page)

getId(*self*) [operation on TreeFormatterJS]

js_init(*self*, [operation on TreeFormatterJS]
open_img, *close_img*, *leaf_img*, *base*,
default_open)

Initialise the JSTree page. This method copies the files to the output directory and stores the values given.

startTree(*self*) [operation on TreeFormatterJS]
Writes the javascript

formatImage([operation on TreeFormatterJS]
self, *id*, *filename*, *alt_text*)
Returns the image element for the given image

writeLeaf(*self*, [operation on TreeFormatterJS]
item_text)
Write a leaf node to the output at the current tree level.

writeNodeStart(*self*, *item_text*) [operation on `TreeFormatterJS`]

Write a non-leaf node to the output at the current tree level, and start a new level.

writeNodeEnd(*self*) [operation on `TreeFormatterJS`]

Finish a non-leaf node, and close the current tree level.

endTree(*self*) [operation on `TreeFormatterJS`]

Writes the end of the tree.

TreeFormatter [module]

Tree formatter interface.

This module contains the class which defines the interface for all tree views. A tree is a structure of leaves and nodes, where each leaf has one node, each node can have many child leaves or nodes, and there is one root node. Trees are used to describe the relationship between modules, and also between files. The user can select between different ways of representing trees, for example as simple nested lists or as complex javascript trees that the user can expand and collapse individual branches of. This module contains the tree interface which is common to all implementations.

TreeFormatter [class]

Interface for all tree implementations. This tree class provides default implementations of its methods for example and basic usage. The implementation provided outputs a nested tree using the UL and LI html elements.

__init__(*self*, *page*) [operation on `TreeFormatter`]

A tree is a strategy, so it must be passed the page instance to display to.

startTree(*self*) [operation on `TreeFormatter`]

Writes anything to the file that needs to be written at the start. For example a script section for the global scripts used by a javascript tree.

endTree(*self*) [operation on `TreeFormatter`]

Writes anything that needs to be written at the end.

writeLeaf(*self*, *text*) [operation on `TreeFormatter`]

Writes a leaf to the output. A leaf is a node with no children, for example a module (not package) or a file

(not directory). The text is output verbatim in the appropriate html tags, which in the default instance is LI

writeNodeStart([operation on `TreeFormatter`]
 self, *text*)

Starts a node with children. The text is written, and then a block of children is started. This method call must be followed by a corresponding `writeNodeEnd()` call. Individual leaves inside the block may be written out using the `writeLeaf()` method.

writeNodeEnd([operation on `TreeFormatter`]
 self)

Ends a node with children. This method just closes any tags opened by the corresponding `writeNodeStart()` method for a node.

XRefPages [module]

XRefLinker [class]

parents: parent class

__init__(*self*, *xref*) [operation on `XRefLinker`]

link(*self*, *name*) [operation on `XRefLinker`]

XRefPages [class]

parents: parent class A module for creating pages full of xref infos

__init__(*self*, *manager*) [operation on `XRefPages`]

get_toc(*self*, *start*) [operation on `XRefPages`]
 Returns the toc for `XRefPages`

filename(*self*) [operation on `XRefPages`]
 Returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on `XRefPages`]
 Returns the current title, which may change over the lifetime of this object

process(*self*, *start*) [operation on `XRefPages`]
 Creates a page for every bunch of xref infos

register_filenames(*self*, *start*) [operation on XRefPages]
Registers each page

process_link(*self*, *file*, *line*, *scope*) [operation on XRefPages]
Outputs the info for one link

describe_decl(*self*, *decl*) [operation on XRefPages]
Returns a description of the declaration. Detects constructors and destructors

process_name(*self*, *name*) [operation on XRefPages]
Outputs the info for a given name

Linker [package]

AccessRestrictor [module]

AccessRestrictor [class]

parents: parent class This class processes declarations, and removes those that need greater access than the maximum passed to the constructor

__init__(*self*) [operation on AccessRestrictor]

execute(*self*, *ast*) [operation on AccessRestrictor]

push(*self*) [operation on AccessRestrictor]

pop(*self*, *decl*) [operation on AccessRestrictor]

add(*self*, *decl*) [operation on AccessRestrictor]

currscope(*self*) [operation on AccessRestrictor]

visitDeclaration(*self*, *decl*) [operation on AccessRestrictor]

visitScope(*self*, *scope*) [operation on AccessRestrictor]

Comments [module]

Comment Processor

CommentProcessor [class]

parents: parent class Base class for comment processors.

This is an AST visitor, and by default all declarations call `process()` with the current declaration. Subclasses may override just the `process` method.

processAll(*self*, [operation on CommentProcessor]
declarations)

process(*self*, *decl*) [operation on CommentProcessor]
Process comments for the given declaration

visitDeclaration([operation on CommentProcessor]
self, *decl*)

SSDComments [class]

parents: parent class A class that selects only `///` comments.

__init__(*self*) [operation on SSDComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on SSDComments]
Calls `processComment` on all comments

processComment(*self*, [operation on SSDComments]
comment)
Replaces the text in the comment. It calls `strip_star()` first to remove all multi-line star comments, then follows with `parse_ssd()`.

strip_star(*self*, *str*) [operation on SSDComments]
Strips all star-format comments from the string

parse_ssd(*self*, *str*) [operation on SSDComments]
Filters *str* and returns just the lines that start with `///`.

JavaComments [class]

parents: parent class A class that formats java `/** style comments`

__init__(*self*) [operation on JavaComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on JavaComments]
Calls `processComment` on all comments

processComment(*self*, [operation on JavaComments]
comment)

Finds comments in the java format. The format is `/** ... */`, and it has to cater for all four line forms: `/** ...`, `* ...`, `*/` and the one-line `/** ... */`.

SSComments [class]

parents: parent class A class that selects only `//` comments.

__init__(*self*) [operation on SSComments]
 Compiles the regular expressions

process(*self*, *decl*) [operation on SSComments]
 Calls processComment on all comments

processComment(*self*, [operation on SSComments]
comment)

Replaces the text in the comment. It calls strip_star() first to remove all multi-line star comments, then follows with parse_ss().

strip_star(*self*, *str*) [operation on SSComments]
 Strips all star-format comments from the string

parse_ss(*self*, *str*) [operation on SSComments]
 Filters str and returns just the lines that start with `//`

QtComments [class]

parents: parent class A class that finds Qt style comments. These have two styles: `//! ...` and `/*! ... */`. The first means "brief comment" and there must only be one. The second type is the detailed comment.

__init__(*self*) [operation on QtComments]
 Compiles the regular expressions

process(*self*, *decl*) [operation on QtComments]
 Calls processComment on all comments

processComment(*self*, [operation on QtComments]
comment)

Matches either brief or detailed comments

Transformer [class]

parents: parent class A class that creates a new AST from an old one. This is a helper base for more specialized classes that manipulate the AST based on the comments in the nodes

__init__(*self*) [operation on Transformer]
 Constructor

processAll(*self*, [operation on Transformer]
 declarations)
 Overrides the default processAll() to setup the stack

push(*self*) [operation on Transformer]
 Pushes the current scope onto the stack and starts a new one

pop(*self*, *decl*) [operation on Transformer]
 Pops the current scope from the stack, and appends the given declaration to it

add(*self*, *decl*) [operation on Transformer]
 Adds the given decl to the current scope

currscope(*self*) [operation on Transformer]
 Returns the current scope: a list of declarations

Dummies [class]

parents: parent class A class that deals with dummy declarations and their comments. This class just removes them.

visitDeclaration(*self*, *decl*) [operation on Dummies]
 Checks for dummy declarations

visitScope(*self*, *scope*) [operation on Dummies]
 Visits all children of the scope in a new scope. The value of currscope() at the end of the list is used to replace scope's list of declarations - hence you can remove (or insert) declarations from the list. Such as dummy declarations :)

visitEnum(*self*, *enum*) [operation on Dummies]
 Does the same as visitScope, but for the enum's list of enumerators

visitEnumerator(*self*, *enumerator*) [operation on Dummies]
 Removes dummy enumerators

Previous [class]

parents: parent class A class that maps comments that begin with '<' to the previous declaration

processAll(*self*, *declarations*) [operation on Previous]

decorates processAll() to initialise last and laststack

push(*self*) [operation on Previous]

decorates push() to also push 'last' onto 'laststack'

pop(*self*, *decl*) [operation on Previous]

decorates pop() to also pop 'last' from 'laststack'

visitScope(*self*, *scope*) [operation on Previous]

overrides visitScope() to set 'last' after each declaration

checkPrevious(*self*, *decl*) [operation on Previous]

Checks a decl to see if the comment should be moved. If the comment begins with a less-than sign, then it is moved to the 'last' declaration

removeSuspect(*self*, *decl*) [operation on Previous]

Removes any suspect comments from the declaration

visitDeclaration(*self*, *decl*) [operation on Previous]

Calls checkPrevious on the declaration and removes dummies

visitEnum(*self*, *enum*) [operation on Previous]

Does the same as visitScope but for enum and enumerators

visitEnumerator(*self*, *enumor*) [operation on Previous]

Checks previous comment and removes dummies

Grouper [class]

parents: parent class A class that detects grouping tags and moves the enclosed nodes into a subnode (a 'Group')

__init__(*self*) [operation on Grouper]

visitDeclaration(*self*, *decl*) [operation on Grouper]

Checks for grouping tags. If an opening tag is found in the middle of a comment, a new Group is generated, the preceding comments are associated with it, and is pushed onto the scope stack as well as the groups stack.

visitScope(*self*, *scope*) [operation on **Grouper**]
 Visits all children of the scope in a new scope. The value of `currscope()` at the end of the list is used to replace `scope`'s list of declarations - hence you can remove (or insert) declarations from the list. Such as dummy declarations :)

visitEnum(*self*, *enum*) [operation on **Grouper**]
 Does the same as `visitScope`, but for the `enum`'s list of enumerators

visitEnumerator(*self*, *enumor*) [operation on **Grouper**]
 Removes dummy enumerators

Summarizer [class]
 parents: parent class Splits comments into summary/detail parts.

`__init__`(*self*) [operation on **Summarizer**]

`process`(*self*, *decl*) [operation on **Summarizer**]
 Combine and summarize the comments of this declaration.

JavaTags [class]
 parents: parent class Extracts javadoc-style @tags from the end of comments.

`__init__`(*self*) [operation on **JavaTags**]

`process`(*self*, *decl*) [operation on **JavaTags**]
 Extract tags from each comment of the given decl

Comments [class]
 parents: parent class

`__init__`(*self*) [operation on **Comments**]
 Constructor, parses the config object

`execute`(*self*, *ast*) [operation on **Comments**]

EmptyNS [module]

EmptyNS [class]
 parents: parent class parent class A class that removes empty namespaces

`__init__(self)` [operation on EmptyNS]
 Overrides the default `processAll()` to setup the stack

`execute(self, ast)` [operation on EmptyNS]

`push(self)` [operation on EmptyNS]
 Pushes the current scope onto the stack and starts a new one

`pop(self, decl)` [operation on EmptyNS]
 Pops the current scope from the stack, and appends the given declaration to it

`pop_only(self)` [operation on EmptyNS]
 Only pops, doesn't append to scope

`add(self, decl)` [operation on EmptyNS]
 Adds the given decl to the current scope

`currscope(self)` [operation on EmptyNS]
 Returns the current scope: a list of declarations

`visitDeclaration(self, decl)` [operation on EmptyNS]
 Adds declaration to scope

`visitGroup(self, group)` [operation on EmptyNS]
 Overrides recursive behaviour to just add the group

`visitEnum(self, enum)` [operation on EmptyNS]
 Overrides recursive behaviour to just add the enum

`visitModule(self, module)` [operation on EmptyNS]
 Visits all children of the module, and if there are no declarations after that removes the module

`_count_not_forwards(self, decls)` [operation on EmptyNS]
 Returns the number of declarations not instances of `AST.Forward`

LanguageMapper [module]

LanguageMapper [class]

parents: parent class

`execute(self, ast)` [operation on LanguageMapper]

| | |
|---|-----------------------|
| Linker | [module] |
| usage() | [function] |
| --parseArgs(<i>args</i>, <i>config_obj</i>) | [function] |
| mapTypes(<i>m</i>) | [function] |
| resolve(<i>args</i>, <i>ast</i>, <i>config_obj</i>) | [function] |
| Config | [class] |
| Central configuration repository for Linker. | |
| __init__(<i>self</i>) | [operation on Config] |
| Constructor - initialise objects to None. | |
| use_config(<i>self</i>, <i>obj</i>) | [operation on Config] |
| Extracts useful attributes from 'obj' and stores them. The object itself is also stored as config.obj | |
| _config_verbose(<i>self</i>, <i>verbose</i>) | [operation on Config] |
| Configures from the given verbose boolean | |
| _config_strip(<i>self</i>, <i>strip</i>) | [operation on Config] |
| Configures from the given list of strip | |
| _config_mapper_list(<i>self</i>, <i>mapper_list</i>) | [operation on Config] |
| Configures from the given list of mapper_list | |
| _config_operations(<i>self</i>, <i>operations</i>) | [operation on Config] |
| Configures from the given list of operations | |
| _config_max_access(<i>self</i>, <i>access</i>) | [operation on Config] |
| _config_map_declaration_names(<i>self</i>, <i>names</i>) | [operation on Config] |
| _config_map_declaration_type(<i>self</i>, <i>typename</i>) | [operation on Config] |
| _config_comment_processors(<i>self</i>, <i>processors</i>) | [operation on Config] |

Operation [class]

The base class for Linker operations. The linker executes a number of operations, depending on the config option 'operations'. Each operation may use other config options

execute(*self*) [operation on Operation]
Executes this operation

CXX2IDL [class]

this function maps a C++ external reference to an IDL interface if the name either starts with 'POA_' or ends in '_ptr'

__init__(*self*) [operation on CXX2IDL]

map(*self*, *unknown*) [operation on CXX2IDL]

Mapper [class]

parents: parent class allow user to supply a mapping functor that is applied to Unknown types. This is useful for linking to externally defined types, such as when cross- referencing modules written in different languages

__init__(*self*, *mappers*) [operation on Mapper]

visitUnknown(*self*, *unknown*) [operation on Mapper]

NameMapper [module]**NameMapper** [class]

parents: parent class parent class This class adds a prefix to all declaration and type names.

visitDeclaration(*self*, *decl*) [operation on NameMapper]

Changes the name of this declaration and its associated type

visitGroup(*self*, *node*) [operation on NameMapper]
Recursively visits declarations under this group/scope/etc

execute(*self*, *ast*) [operation on NameMapper]

Stripper [module]

filterName(*name*, *prefixes*) [function]

Stripper [class]

parents: parent class parent class Strip common prefix from the declaration's name. Keep a list of root nodes, such that children whos parent scopes are not accepted but which themselves are correct can be maintained as new root nodes.

__init__(*self*) [operation on Stripper]

execute(*self*, *ast*) [operation on Stripper]

_stripName(*self*, *name*) [operation on Stripper]

stripDeclarations(*self*,
declarations) [operation on Stripper]

stripTypes(*self*, *types*) [operation on Stripper]

declarations(*self*) [operation on Stripper]

strip(*self*, *declaration*) [operation on Stripper]
test whether the declaration matches one of the prefixes, strip it off, and return success. Success means that the declaration matches the prefix set and thus should not be removed from the AST.

visitScope(*self*, *scope*) [operation on Stripper]

visitClass(*self*, *clas*) [operation on Stripper]

visitDeclaration(*self*, *decl*) [operation on Stripper]

visitEnumerator(*self*,
enumerator) [operation on Stripper]

visitEnum(*self*, *enum*) [operation on Stripper]

visitFunction(*self*, *function*) [operation on Stripper]

visitOperation(*self*,
operation) [operation on Stripper]

visitMetaModule(*self*,
module) [operation on Stripper]

Unduplicator [module]

| | |
|---|-----------------------------|
| Unduplicator | [class] |
| parents: parent class parent class Visitor that removes duplicate declarations | |
| <code>__init__(self)</code> | [operation on Unduplicator] |
| <code>execute(self, ast)</code> | [operation on Unduplicator] |
| <code>lookup(self, name)</code> | [operation on Unduplicator] |
| look whether the current scope already contains a declaration with the given name | |
| <code>append(self, declaration)</code> | [operation on Unduplicator] |
| append declaration to the current scope | |
| <code>push(self, scope)</code> | [operation on Unduplicator] |
| push new scope on the stack | |
| <code>pop(self)</code> | [operation on Unduplicator] |
| restore the previous scope | |
| <code>top(self)</code> | [operation on Unduplicator] |
| <code>top_dict(self)</code> | [operation on Unduplicator] |
| <code>linkType(self, type)</code> | [operation on Unduplicator] |
| Returns the same or new proxy type | |
| <code>visitBaseType(self, type)</code> | [operation on Unduplicator] |
| <code>visitUnknown(self, type)</code> | [operation on Unduplicator] |
| <code>visitDeclared(self, type)</code> | [operation on Unduplicator] |
| <code>visitTemplate(self, type)</code> | [operation on Unduplicator] |
| <code>visitModifier(self, type)</code> | [operation on Unduplicator] |
| <code>visitArray(self, type)</code> | [operation on Unduplicator] |
| <code>visitParametrized(self, type)</code> | [operation on Unduplicator] |

visitFunctionType(*self*, [operation on Unduplicator]
type)

visitSourceFile(*self*, [operation on Unduplicator]
file)

Resolves any duplicates in the list of declarations from this file

visitModule(*self*, [operation on Unduplicator]
module)

merge_comments(*self*, [operation on Unduplicator]
dest, *src*)

Merges the src comments into dest. Merge is just an append, unless src already exists inside dest!

visitMetaModule(*self*, [operation on Unduplicator]
module)

addDeclaration(*self*, [operation on Unduplicator]
decl)

Adds a declaration to the current (top) scope. If there is already a Forward declaration, then this replaces it unless this is also a Forward.

visitNamed(*self*, *decl*) [operation on Unduplicator]

visitFunction(*self*, *func*) [operation on Unduplicator]

visitVariable(*self*, *var*) [operation on Unduplicator]

visitTypedef(*self*, *tdef*) [operation on Unduplicator]

visitClass(*self*, *clas*) [operation on Unduplicator]

visitInheritance(*self*, [operation on Unduplicator]
parent)

visitParameter(*self*, [operation on Unduplicator]
param)

visitConst(*self*, *const*) [operation on Unduplicator]

XRefCompiler [module]

do_compile(*input_files*, *output_file*, [function]
no_locals)

XRefCompiler [class]

parents: parent class This class compiles a set of text-based xref files from the C++ parser into a cPickled data structure with a name index.

The format of the data structure is:

```
(data, index) = cPickle.load() data = dict<scoped targetname, target_data> index = dict<name, list<scoped targetname>> target_data = (definitions = list<target_info>, func calls = list<target_info>, references = list<target_info>) target_info = (filename, int(line number), scoped context name) </pre> The scoped targetnames in the index are guaranteed to exist in the data dictionary.
```

__init__(*self*) [operation on XRefCompiler]

execute(*self*, *ast*) [operation on XRefCompiler]

Parser [package]**C++** [package]**emul** [module]

get_temp_file() [function]

Returns the temporary filename. The temp file is created, but is left empty

cleanup_temp_file() [function]

Removes the temporary file and resets the filename

main() [function]

The main function - parses the arguments and controls the program

get_fallback(*preferred*, *is_first_time*) [function]

Tries to return info from a fallback compiler, and prints a warning message to the user, unless their preferred compiler was 'none'

get_compiler_info(*compiler*) [function]

Returns the compiler info for the given compiler. The info is returned as a CompilerInfo object, or None if the compiler isn't found.

load_compiler_infos() [function]

Loads the compiler infos from a file

get_compiler_timestamp(*compiler*) [function]
Returns the timestamp for the given compiler, or 0 if not found

refresh_compiler_infos(*infos*) [function]
Refreshes the list of infos, by rediscovering all non-custom compilers in the map. The map is modified in-place.

write_compiler_infos(*infos*) [function]

find_compiler_info(*compiler*) [function]

CompilerInfo [class]
Info about one compiler.

**__init__(*self*, [operation on CompilerInfo]
compiler, *is_custom*, *timestamp*,
include_paths, *macros*)**

IDL [package]

omni [module]

strip(*filename*) [function]

strip_filename(*filename*) [function]
This is aliased as strip if -b used and basename set

usage() [function]

__parseArgs(*args*, *config_obj*) [function]

**parse(*file*, *extra_files*, *args*,
config_obj)** [function]

TypeTranslator [class]
parents: parent class maps idltype objects to Synopsis.Type objects in a Type.Dictionary

**__init__(*self*, [operation on TypeTranslator]
types)**

**internalize(*self*, [operation on TypeTranslator]
idltype)**

**add(*self*, *name*, [operation on TypeTranslator]
type)**

`get(self, name)` [operation on TypeTranslator]
`visitBaseType(self, idltype)` [operation on TypeTranslator]
`visitStringType(self, idltype)` [operation on TypeTranslator]
`visitWStringType(self, idltype)` [operation on TypeTranslator]
`visitSequenceType(self, idltype)` [operation on TypeTranslator]
`visitDeclaredType(self, idltype)` [operation on TypeTranslator]

ASTTranslator [class]

parents: parent class

`__init__(self, declarations, types, mainfile_only)` [operation on ASTTranslator]
`scope(self)` [operation on ASTTranslator]
`addDeclaration(self, declaration)` [operation on ASTTranslator]
`addType(self, name, type)` [operation on ASTTranslator]
`getType(self, name)` [operation on ASTTranslator]
`visitAST(self, node)` [operation on ASTTranslator]
`visitModule(self, node)` [operation on ASTTranslator]
`visitInterface(self, node)` [operation on ASTTranslator]
`visitForward(self, node)` [operation on ASTTranslator]
`visitConst(self, node)` [operation on ASTTranslator]

| | |
|---|------------------------------|
| visitTypedef (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitMember (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitStruct (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitException (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitUnionCase (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitUnion (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitEnumerator (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitEnum (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitAttribute (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitParameter (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitOperation (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |

Python [package]

exparse [module]

Simple code to extract class & function docstrings from a module.

This code is used as an example in the library reference manual in the section on using the parser module. Refer to the manual for a thorough discussion of the operation of this code.

The code has been extended by Stephen Davies for the Synopsis project. It now also recognises parameter names and values, and baseclasses. Names are now returned in order also.

findModulePath(*module*) [function]

Return the path for the given module

format(*tree*, *depth*) [function]
 Format the given tree up to the given depth. Numbers are replaced with their symbol or token names.

stringify(*tree*) [function]
 Convert the given tree to a string

get_docs(*file*) [function]
 Retrieve information from the parse tree of a source file.
file Name of the file to read Python source code from.

filter_names(*list*) [function]

map_second(*list*) [function]

map_rest(*list*) [function]

get_names_only(*list*) [function]

match(*pattern*, *data*, *vars*) [function]
 Match 'data' to 'pattern', with variable extraction.
pattern Pattern to match against, possibly containing variables.
data Data to be checked and against which variables are extracted.
vars Dictionary of variables which have already been found. If not provided, an empty dictionary is created.
 The 'pattern' value may contain variables of the form ['varname'] which are allowed to match anything. The value that is matched is returned as part of a dictionary which maps 'varname' to the matched value. 'varname' is not required to be a string object, but using strings makes patterns and the code which uses them more readable.
 This function returns two values: a boolean indicating whether a match was found and a dictionary mapping variable names to their associated values.

dmatch(*pattern*, *data*, *vars*) [function]
 Debugging match

SuiteInfoBase [class]

__init__(*self*, *tree*, *env*) [operation on SuiteInfoBase]

`_extract_info(self, tree)` [operation on SuiteInfoBase]

`_addImport(self, names)` [operation on SuiteInfoBase]

`_addFromImport(self, module, names)` [operation on SuiteInfoBase]

`get_docstring(self)` [operation on SuiteInfoBase]

`get_name(self)` [operation on SuiteInfoBase]

`get_class_names(self)` [operation on SuiteInfoBase]

`get_class_info(self, name)` [operation on SuiteInfoBase]

`__getitem__(self, name)` [operation on SuiteInfoBase]

SuiteFuncInfo [class]

`get_function_names(self)` [operation on SuiteFuncInfo]

`get_function_info(self, name)` [operation on SuiteFuncInfo]

FunctionInfo [class]

parents: parent class parent class

`__init__(self, tree, env)` [operation on FunctionInfo]

`get_params(self)` [operation on FunctionInfo]

`get_param_defaults(self)` [operation on FunctionInfo]

ClassInfo [class]

parents: parent class

`__init__(self, tree, env)` [operation on ClassInfo]

get_method_names(*self*) [operation on ClassInfo]

get_method_info(*self*, *name*) [operation on ClassInfo]

get_base_names(*self*) [operation on ClassInfo]

ModuleInfo [class]

parents: parent class parent class

__init__(*self*, *tree*, *name*) [operation on ModuleInfo]

python [module]

Main module for the Python parser. Parsing python is achieved by using the code in the Python distribution that is an example for parsing python by using the built-in parser. This parser returns a parse tree which we can traverse and translate into Synopsis' AST. The exparse module contains the enhanced example code (it has many more features than the simple example did), and this module translates the resulting intermediate AST objects into Synopsis.Core.AST objects.

addDeclaration(*decl*) [function]

Adds the given declaration to the current top scope and to the SourceFile for this file.

push(*scope*) [function]

Pushes the given scope onto the top of the stack

pop() [function]

Pops the scope stack by one level

scopeName(*name*) [function]

Scopes the given name. If the given name is a list then it is returned verbatim, else it is concatenated with the (scoped) name of the current scope

process_ModuleInfo(*mi*) [function]

Processes a ModuleInfo object. The comments are extracted, and any functions and comments recursively processed.

add_params(*func*, *fi*) [function]

Adds the parameters of 'fi' to the AST.Function 'func'.

process_FunctionInfo(*fi*) [function]
 Process a FunctionInfo object. An AST.Function object is created and inserted into the current scope.

process_MethodInfo(*fi*) [function]
 Process a MethodInfo object. An AST.Operation object is created and inserted into the current scope.

process_ClassInfo(*ci*) [function]
 Process a ClassInfo object. An AST.Class object is created and inserted into the current scope. The inheritance of the class is also parsed, and nested classes and methods recursively processed.

usage() [function]
 Prints a usage message

--parseArgs(*args*, *config_obj*) [function]
 Parses the command line arguments and the config object

get_synopsis(*file*) [function]
 Returns the docstring from the top of an open file

parse(*file*, *extra_files*, *parser_args*, *config_obj*) [function]
 Entry point for the Python parser

UI [package]

Qt [package]

actionvis [module]

CanvasStrategy [class]

An interface for a strategy to handle mouse events

__init__(*self*, *canvas*, *view*) [operation on CanvasStrategy]

reset(*self*) [operation on CanvasStrategy]

set(*self*) [operation on CanvasStrategy]

press(*self*, *event*) [operation on CanvasStrategy]

release(*self*, *event*) [operation on CanvasStrategy]

move(*self*, *event*) [operation on CanvasStrategy]

doubleClick(*self*, *event*) [operation on CanvasStrategy]

key(*self*, *event*) [operation on CanvasStrategy]

SelectionStrategy [class]

parents: parent class The normal CanvasStrategy to handle mouse actions.

MenuHandler [class]

Wraps menu callbacks with an action or line object

__init__(*self*, *sel*, *obj*) [operation on MenuHandler]

on_properties(*self*) [operation on MenuHandler]

on_delete_action(*self*) [operation on MenuHandler]

on_delete_line(*self*) [operation on MenuHandler]

on_default_formatter(*self*) [operation on MenuHandler]

__init__(*self*, *canvas*, *view*) [operation on SelectionStrategy]

reset(*self*) [operation on SelectionStrategy]

press(*self*, *event*) [operation on SelectionStrategy]

release(*self*, *event*) [operation on SelectionStrategy]

move(*self*, *event*) [operation on SelectionStrategy]

hilite(*self*, *obj*) [operation on SelectionStrategy]

Sets the currently hilited object. The object may be None, an Icon or a Line

doubleClick(*self*, *event*) [operation on SelectionStrategy]

key(*self*, *event*) [operation on SelectionStrategy]

Override default set-mode-to-select behaviour

on_delete_line(*self*, *line*) [operation on SelectionStrategy]

on_delete_action(*self*, *action*) [operation on SelectionStrategy]

on_default_formatter(*self*, *action*) [operation on SelectionStrategy]

on_properties(*self*, *action*) [operation on SelectionStrategy]

ConnectStrategy [class]

parents: parent class

__init__(*self*, *canvas*, *view*) [operation on ConnectStrategy]

set(*self*) [operation on ConnectStrategy]

reset(*self*) [operation on ConnectStrategy]

move(*self*, *event*) [operation on ConnectStrategy]

press(*self*, *event*) [operation on ConnectStrategy]

release(*self*, *event*) [operation on ConnectStrategy]

setSource(*self*, *action*, *event*) [operation on ConnectStrategy]

setDest(*self*, *action*) [operation on ConnectStrategy]

AddActionStrategy [class]

parents: parent class

`--init--(self, [operation on AddActionStrategy]
 canvas, view, action_type)`

`set(self) [operation on AddActionStrategy]`

`reset(self) [operation on AddActionStrategy]`

`move(self, [operation on AddActionStrategy]
 event)`

`release(self, [operation on AddActionStrategy]
 event)`

ActionPropertiesDialog [class]
parents: parent class

`--init--([operation on ActionPropertiesDialog]
 self, parent, action)`

ActionColorizer [class]
parents: parent class

`--init--(self, [operation on ActionColorizer]
 action)`

`visitAction(self, [operation on ActionColorizer]
 action)`

`visitSource(self, [operation on ActionColorizer]
 action)`

`visitParser(self, [operation on ActionColorizer]
 action)`

`visitLinker(self, [operation on ActionColorizer]
 action)`

`visitCacher(self, [operation on ActionColorizer]
 action)`

`visitFormat(self, [operation on ActionColorizer]
 action)`

ActionIcon [class]
parents: parent class

`--init--(self, action) [operation on ActionIcon]`

| | |
|---|---------------------------|
| visitAction (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |
| visitSource (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |
| visitParser (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |
| visitLinker (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |
| visitCacher (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |
| visitFormat (<i>self</i> , <i>action</i>) | [operation on ActionIcon] |

Icon [class]

Encapsulates the canvas display of an Action

| | |
|--|---------------------|
| __init__ (<i>self</i> , <i>canvas</i> , <i>action</i>) | [operation on Icon] |
| hide (<i>self</i>) Hides icon on canvas | [operation on Icon] |
| set_hilite (<i>self</i> , <i>yesno</i>) | [operation on Icon] |
| update_pos (<i>self</i>) | [operation on Icon] |

Line [class]

Encapsulates the canvas display of a channel between two Actions

| | |
|--|---------------------|
| __init__ (<i>self</i> , <i>canvas</i> , <i>source</i> , <i>dest</i>) | [operation on Line] |
| hide (<i>self</i>) Hide line on canvas | [operation on Line] |
| set_hilite (<i>self</i> , <i>yesno</i>) | [operation on Line] |
| update_pos (<i>self</i>) | [operation on Line] |

ActionCanvas [class]

parents: parent class Extends QCanvas to automatically fill and update the canvas when notified of events by an Action-Manager

__init__(*self*, [operation on ActionCanvas]
actions, *parent*, *project*)

get_action_at(*self*, [operation on ActionCanvas]
x, *y*)
Returns the Action (if any) at the given coordinates

get_line_at(*self*, *x*, [operation on ActionCanvas]
y)
Returns the Actions forming a line which crosses *x,y*,
as a line object

get_icon_for(*self*, [operation on ActionCanvas]
action)

action_added(*self*, [operation on ActionCanvas]
action)
Callback from ActionManager. Adds an Icon for the
new Action

action_removed([operation on ActionCanvas]
self, *action*)
Callback from ActionManager. Removes the Icon for
the Action

action_moved(*self*, [operation on ActionCanvas]
action)
Callback from ActionManager. Moves the Icon to fol-
low the Action

channel_added(*self*, [operation on ActionCanvas]
source, *dest*)
Callback from ActionManager. Adds a channel be-
tween the given actions

channel_removed([operation on ActionCanvas]
self, *source*, *dest*)
Callback from ActionManager. Adds a channel be-
tween the given actions

action_changed([operation on ActionCanvas]
self, *action*)
Callback from ProjectActions. Indicates changes, in-
cluding rename

CanvasView [class]

parents: parent class

`--init--(self, canvas, parent)` [operation on CanvasView]`modeChanged(self, tool)` [operation on CanvasView]`contentsMouseEvent(self, event)` [operation on CanvasView]`contentsMouseReleaseEvent(self, event)` [operation on CanvasView]`contentsMouseMoveEvent(self, event)` [operation on CanvasView]`contentsMouseDoubleClickEvent(self, event)` [operation on CanvasView]`keyPressEvent(self, event)` [operation on CanvasView]**CanvasWindow** [class]

parents: parent class

`--init--(self, parent, main_window, project)` [operation on CanvasWindow]`_makeNewTool(self, name, short_name, icon_id)` [operation on CanvasWindow]`resizeEvent(self, ev)` [operation on CanvasWindow]`setMode(self, mode)` [operation on CanvasWindow]`windowActivated(self, widget)` [operation on CanvasWindow]`activate(self)` [operation on CanvasWindow]`deactivate(self)` [operation on CanvasWindow]`setSelect(self)` [operation on CanvasWindow]

setConnect(*self*) [operation on CanvasWindow]

newSourceAction(*self*) [operation on CanvasWindow]

newParserAction(*self*) [operation on CanvasWindow]

newLinkerAction(*self*) [operation on CanvasWindow]

newCacheAction(*self*) [operation on CanvasWindow]

newFormatterAction(*self*) [operation on CanvasWindow]

saveProject(*self*) [operation on CanvasWindow]

saveProjectAs(*self*) [operation on CanvasWindow]

actionwiz [module]

make_relative(*base, file*) [function]
Useful function to make a filename relative to the given base filename

Struct [class]
Dummy class. Initialise with keyword args.

__init__(*self, **keys*) [operation on Struct]

SourceActionEditor [class]
Presents a modal GUI for editing a Source Action

__init__(*self, parent, project*) [operation on SourceActionEditor]

edit(*self, action*) [operation on SourceActionEditor]
Edits the given action

keep_changes(*self*) [operation on SourceActionEditor]

init_dialog(*self*) [operation on SourceActionEditor]

on_insert(self) [operation on SourceActionEditor]

on_insert_finished(self, title) [operation on SourceActionEditor]

update_list(self) [operation on SourceActionEditor]

get_selected_index(self) [operation on SourceActionEditor]

Returns the index of the selected item, or None

on_selection(self) [operation on SourceActionEditor]

on_delete(self) [operation on SourceActionEditor]

on_up(self) [operation on SourceActionEditor]

on_down(self) [operation on SourceActionEditor]

on_edit(self) [operation on SourceActionEditor]

on_test(self) [operation on SourceActionEditor]

CacherPage [class]

parents: parent class The Page (portion of dialog) that displays options for a Cacher Action

__init__(self, parent, action_container) [operation on CacherPage]

title(self) [operation on CacherPage]

action(self) [operation on CacherPage]

set_action(self, action) [operation on CacherPage]

_make_layout(self) [operation on CacherPage]

pre_show(self) [operation on CacherPage]

showEvent(self, ev) [operation on CacherPage]

onActionName(*self*, *name*) [operation on CacherPage]

onButtonFile(*self*) [operation on CacherPage]

onButtonDir(*self*) [operation on CacherPage]

onFileChanged(*self*, *text*) [operation on CacherPage]

onDirChanged(*self*, *text*) [operation on CacherPage]

onBrowseFile(*self*) [operation on CacherPage]

onBrowseDir(*self*) [operation on CacherPage]

ParserPage [class]

parents: parent class The Page (portion of dialog) that displays options for a Parser Action

__init__(*self*, *parent*, *action_container*) [operation on ParserPage]

title(*self*) [operation on ParserPage]

action(*self*) [operation on ParserPage]

set_action(*self*, *action*) [operation on ParserPage]

_make_layout(*self*) [operation on ParserPage]

pre_show(*self*) [operation on ParserPage]

showEvent(*self*, *ev*) [operation on ParserPage]

onActionName(*self*, *name*) [operation on ParserPage]

onModuleSelected(*self*, *item*) [operation on ParserPage]

CppParserPage [class]

parents: parent class The Page (portion of dialog) that displays options for a C++ Parser Action

| | |
|--|------------------------------|
| <code>__init__(self, parent, action_container)</code> | [operation on CppParserPage] |
| <code>title(self)</code> | [operation on CppParserPage] |
| <code>action(self)</code> | [operation on CppParserPage] |
| <code>set_action(self, action)</code> | [operation on CppParserPage] |
| <code>_make_layout(self)</code> | [operation on CppParserPage] |
| <code>_update_path_list(self)</code> | [operation on CppParserPage] |
| <code>_update_def_list(self)</code> | [operation on CppParserPage] |
| <code>pre_show(self)</code> | [operation on CppParserPage] |
| <code>showEvent(self, ev)</code> | [operation on CppParserPage] |
| <code>onPathSelected(self, item)</code> | [operation on CppParserPage] |
| <code>onDefSelected(self, item)</code> | [operation on CppParserPage] |
| <code>onAddPath(self)</code> | [operation on CppParserPage] |
| <code>onRemovePath(self)</code> | [operation on CppParserPage] |
| <code>onAddGCC(self)</code> | [operation on CppParserPage] |
| <code>onMainFile(self, on)</code> | [operation on CppParserPage] |
| <code>onAddDefine(self)</code> | [operation on CppParserPage] |
| <code>onRemoveDefine(self)</code> | [operation on CppParserPage] |

FormatterPage [class]
 parents: parent class The Page (portion of dialog) that displays options for a Formatter Action

```

__init__( self,          [operation on FormatterPage]
          parent, action_container)

title( self)             [operation on FormatterPage]

action( self)           [operation on FormatterPage]

set_action( self,       [operation on FormatterPage]
          action)

_make_layout( self)     [operation on FormatterPage]

pre_show( self)         [operation on FormatterPage]

showEvent( self,       [operation on FormatterPage]
          ev)

onActionName(          [operation on FormatterPage]
          self, name)

```

DetailsPage [class]

parents: parent class The Page (portion of dialog) that displays config details

```

__init__( self, parent,  [operation on DetailsPage]
          action_container)

title( self)             [operation on DetailsPage]

action( self)           [operation on DetailsPage]

set_action( self,       [operation on DetailsPage]
          action)

_make_layout( self)     [operation on DetailsPage]

pre_show( self)         [operation on DetailsPage]

_fill_list( self, obj,  [operation on DetailsPage]
          parent)

```

ActionDialog [class]

parents: parent class

```

__init__( self, parent,  [operation on ActionDialog]
          action, project)

_get_pages( self)       [operation on ActionDialog]

```

AddWizard [class]

parents: parent class

`__init__`(*self*, *parent*, *action*, *project*) [operation on AddWizard]**`_makeStartPage`**(*self*) [operation on AddWizard]**`onActionType`**(*self*, *id*) [operation on AddWizard]**`_makeSourcePage`**(*self*) [operation on AddWizard]**`_makeParserPage`**(*self*) [operation on AddWizard]**`_makeCacherPage`**(
self) [operation on AddWizard]**`_makeFormatterPage`**(
self) [operation on AddWizard]**`onFormatModule`**(*self*, *id*) [operation on AddWizard]**`_copyAttrs`**(*self*, *dest*, *src*, *overwrite*) [operation on AddWizard]

Copies attributes from *src* to *dest* objects. *src* may be a class, and recursive structs/classes are copied properly. Note that the copy module doesn't copy classes, so we do that bit ourselves

`_makeFormatPropPage`(
self) [operation on AddWizard]**`onActionName`**(*self*, *name*) [operation on AddWizard]**`onSourceAddPath`**(
self, *name*) [operation on AddWizard]**`_makeFinishPage`**(*self*) [operation on AddWizard]**`showPage`**(*self*, *page*) [operation on AddWizard]

layOutTitleRow(*self*, [operation on AddWizard]
 hbox, *title*)

browse [module]

format_source(*text*) [function]
 The source relies on stylesheets, and Qt doesn't have powerful enough stylesheets. This function manually converts the html...

BrowserWindow [class]
 parents: parent class The browser window displays an AST in the familiar (from JavaDoc) three-pane view. In addition to JavaDoc, the right pane can display documentation, source or a class hierarchy graph. The bottom-left pane can also display classes or files.

SelectionListener [class]
 Defines the interface for an object that listens to the browser selection

current_decl_changed([operation on SelectionListener]
 self, *decl*)
 Called when the current decl changes

current_package_changed([operation on SelectionListener]
 self, *package*)
 Called when the current package changes

current_ast_changed([operation on SelectionListener]
 self, *ast*)
 Called when the current AST changes. Browser's glob will be updated first

__init__(*self*, [operation on BrowserWindow]
 main_window, *filename*, *project_window*,
 config)

_make_left(*self*) [operation on BrowserWindow]

_make_right(*self*) [operation on BrowserWindow]

add_listener(*self*, [operation on BrowserWindow]
 listener)
 Adds a listener for changes. The listener must implement the SelectionListener interface

current_decl(*self*) [operation on `BrowserWindow`]
Returns the current declaration being viewed by the project

set_current_decl(*self*, *decl*) [operation on `BrowserWindow`]
Sets the current declaration being viewed by the project. This will also notify all displays

set_current_package(*self*, *package*) [operation on `BrowserWindow`]
Sets the current package (a Scope declaration) being viewed by the project. This will also notify all displays

set_current_ast(*self*, *ast*) [operation on `BrowserWindow`]

current_ast(*self*) [operation on `BrowserWindow`]

windowActivated(*self*, *widget*) [operation on `BrowserWindow`]

activate(*self*) [operation on `BrowserWindow`]

deactivate(*self*) [operation on `BrowserWindow`]

openGraph(*self*) [operation on `BrowserWindow`]

setGraphEnabled(*self*, *enable*) [operation on `BrowserWindow`]

tabChanged(*self*, *widget*) [operation on `BrowserWindow`]

load_file(*self*) [operation on `BrowserWindow`]
Loads the AST from disk.

ListFiller [class]
parents: parent class A visitor that fills in a `QListView` from an AST

__init__(*self*, *main*, *listview*, *types*, *anti_types*) [operation on `ListFiller`]

clear(*self*) [operation on `ListFiller`]

fillFrom(*self*, *decl*) [operation on `ListFiller`]

visitDeclaration(*self*, [operation on ListFiller]
 decl)

addDeclaration(*self*, [operation on ListFiller]
 decl)

visitGroup(*self*, [operation on ListFiller]
 group)

addGroup(*self*, [operation on ListFiller]
 group)

visitForward(*self*, [operation on ListFiller]
 fwd)

visitEnum(*self*, *enum*) [operation on ListFiller]

PackageBrowser [class]

parents: parent class Browser that manages the package view

__init__(*self*, [operation on PackageBrowser]
 browser)

select_package_item([operation on PackageBrowser]
 self, *item*)
 Show a given package (by item)

DISABLED_current_package_changed([operation on PackageBrowser]
 self, *decl*)

current_ast_changed([operation on PackageBrowser]
 self, *ast*)

ClassBrowser [class]

parents: parent class Browser display that manages the class view

__init__(*self*, [operation on ClassBrowser]
 browser)

select_decl_item([operation on ClassBrowser]
 self, *item*)
 Show a given declaration (by item)

current_package_changed([operation on ClassBrowser]
 self, *package*)
 Refill the tree with the new package as root

selfish_expansion(*self*, *item*) [operation on ClassBrowser]

Selfishly makes item the only expanded node

current_ast_changed(*self*, *ast*) [operation on ClassBrowser]

DocoBrowser [class]
parents: parent class Browser that manages the documenta-
tion view

BufferScopePages [class]
parents: parent class parent class

__init__(*self*, *manager*) [operation on BufferScopePages]

__init__(*self*, *browser*) [operation on DocoBrowser]

generator(*self*) [operation on DocoBrowser]

current_decl_changed(*self*, *decl*) [operation on DocoBrowser]

current_ast_changed(*self*, *ast*) [operation on DocoBrowser]

get_mime_data(*self*, *name*) [operation on DocoBrowser]

SourceMimeFactory [class]
parents: parent class

set_browser(*self*, *browser*) [operation on SourceMimeFactory]

data(*self*, *name*) [operation on SourceMimeFactory]

SourceBrowser [class]
parents: parent class Browser that manages the source view

BufferFilePages [class]
parents: parent class parent class

`__init__(self, manager)` [operation on BufferFilePages]

`__init__(self, browser)` [operation on SourceBrowser]

`highlighted(self, text)` [operation on SourceBrowser]

`generator(self)` [operation on SourceBrowser]

`current_decl_changed(self, decl)` [operation on SourceBrowser]

`current_ast_changed(self, ast)` [operation on SourceBrowser]

GraphBrowser [class]

parents: parent class Browser that manages the graph view

`__init__(self, browser)` [operation on GraphBrowser]

`current_decl_changed(self, decl)` [operation on GraphBrowser]

igraph [module]

IGraphWindow [class]

parents: parent class

Icon [class]

`__init__(self, canvas, classname, x, y)` [operation on Icon]

`move_to(self, x, y)` [operation on Icon]

`width(self)` [operation on Icon]
returns the width of this icon

`mid(self)` [operation on Icon]
Returns the x, y coords of the middle of this icon

`add_sub(self, sub, line)` [operation on Icon]

move_sub(*self*, *line*) [operation on Icon]

add_super(*self*, *super*, [operation on Icon]
line)

move_super(*self*, *line*) [operation on Icon]

clear(*self*) [operation on Icon]
Removes all references to other icons etc so they
can be garbage collected

__init__(*self*, *parent*, [operation on IGraphWindow]
main_window, *classTree*)

set_class(*self*, [operation on IGraphWindow]
classname)

organize(*self*) [operation on IGraphWindow]

un_offset(*self*) [operation on IGraphWindow]
Moves the graph to the top-left corner of the display

rank_down(*self*, [operation on IGraphWindow]
node)

space_ranks(*self*) [operation on IGraphWindow]
Spaces out the ranks vertically

sort_ranks(*self*) [operation on IGraphWindow]

main [module]

MainWindow [class]
parents: parent class The main window of the applet. It
controls the whole GUI.

__init__(*self*) [operation on MainWindow]

open_project(*self*) [operation on MainWindow]
Opens a project

do_open_project(*self*, [operation on MainWindow]
filename)
Opens a project for the given filename

open_file(*self*) [operation on MainWindow]
 A file selection dialog is opened to prompt the user for a filename.

do_open_file(*self*, *filename*) [operation on MainWindow]
 Opens a file in a new project.

new_project(*self*) [operation on MainWindow]

execute_project(*self*) [operation on MainWindow]

add_window(*self*, *window*) [operation on MainWindow]

remove_window(*self*, *window*) [operation on MainWindow]

project [module]

The project window. Displays the output of the project.

ProjectWindow [class]

parents: parent class

__init__(*self*, *main_window*, *filename*) [operation on ProjectWindow]

windowActivated(*self*, *widget*) [operation on ProjectWindow]

activate(*self*) [operation on ProjectWindow]

deactivate(*self*) [operation on ProjectWindow]

execute_project(*self*) [operation on ProjectWindow]

Tries to show some output in the browser parts of the project window. To do this it searches for an appropriate FormatAction (ie, one that has HTML options), checks that everything is up to date, and loads the AST into the browser windows.

checkThreadOutput(*self*) [operation on ProjectWindow]

This method is called periodically to check output on the tempfile

sourceeditexclude [module]

SourceEditExclude [class]

parents: parent class

`__init__(self, [operation on SourceEditExclude]
parent, name, modal, fl)`

`OkButton_clicked([operation on SourceEditExclude]
self)`

`CancelButton_clicked([operation on SourceEditExclude]
self)`

sourceeditglob [module]

SourceEditGlob [class]

parents: parent class

`__init__(self, [operation on SourceEditGlob]
parent, name, modal, fl)`

`AddDirectoryButton_clicked([operation on SourceEditGlob]
self)`

`RemoveDirectoryButton_clicked([operation on SourceEditGlob]
self)`

`DirList_selectionChanged([operation on SourceEditGlob]
self)`

`MakeRelativeButton_clicked([operation on SourceEditGlob]
self)`

`OkButton_clicked([operation on SourceEditGlob]
self)`

`CancelButton_clicked([operation on SourceEditGlob]
self)`

sourceeditssimple [module]

SourceEditSimple [class]

parents: parent class

`__init__(self, [operation on SourceEditSimple]
parent, name, modal, fl)`

AddFilesButton_clicked([operation on SourceEditSimple]
self)

RemoveFileButton_clicked([operation on SourceEditSimple]
self)

FileList_selectionChanged([operation on SourceEditSimple]
self)

MakeRelativeButton_clicked([operation on SourceEditSimple]
self)

OkButton_clicked([operation on SourceEditSimple]
self)

CancelButton_clicked([operation on SourceEditSimple]
self)

sourceinsertwizard [module]

SourceInsertWizard [class]

parents: parent class

__init__(*self*, [operation on SourceInsertWizard]
parent, name, modal, fl)

init(*self*) [operation on SourceInsertWizard]

SelectFilesButton_clicked([operation on SourceInsertWizard]
self)

IncludeRadio_clicked([operation on SourceInsertWizard]
self)

ExcludeRadio_clicked([operation on SourceInsertWizard]
self)

SimpleRadio_clicked([operation on SourceInsertWizard]
self)

DirRadio_clicked([operation on SourceInsertWizard]
self)

BaseRadio_clicked([operation on SourceInsertWizard]
self)

AddFilesButton_clicked([operation on SourceInsertWizard]
self)

RemoveFileButton_clicked(on SourceInsertWizard
self)

FileList_selectionChanged(on SourceInsertWizard
self)

RelativeNoneRadio_clicked(on SourceInsertWizard
self)

RelativeCurrentRadio_clicked(on SourceInsertWizard
self)

RelativeThisRadio_clicked(on SourceInsertWizard
self)

RelativeThisButton_clicked(on SourceInsertWizard
self)

AddDirectoryButton_clicked(on SourceInsertWizard
self)

RemoveDirectoryButton_clicked(on SourceInsertWizard
self)

DirList_selectionChanged(on SourceInsertWizard
self)

sourceoptionsdialog [module]

SourceOptionsDialog [class]

parents: parent class

__init__(self, [operation on SourceOptionsDialog]
parent, name, modal, fl)

init(self) [operation on SourceOptionsDialog]

OkButton_clicked(operation on SourceOptionsDialog
self)

CancelButton_clicked(operation on SourceOptionsDialog
self)

2.2 The Type Module

| | |
|---|----------------------|
| ccmp (<i>a</i> , <i>b</i>) | [function] |
| Compares classes of two objects | |
| Error | [class] |
| Exception class used by Type internals. | |
| __init__ (<i>self</i> , <i>err</i>) | [operation on Error] |
| __repr__ (<i>self</i>) | [operation on Error] |
| Type | [class] |
| Type abstract class. | |
| __init__ (<i>self</i> , <i>language</i>) | [operation on Type] |
| language (<i>self</i>) | [operation on Type] |
| the language this type was defined in | |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Type] |
| visitor pattern accept. Visitor | |
| __cmp__ (<i>self</i> , <i>other</i>) | [operation on Type] |
| Comparison operator | |
| Named | [class] |
| parents: parent class Named type abstract class | |
| __init__ (<i>self</i> , <i>language</i> , <i>name</i>) | [operation on Named] |
| name (<i>self</i>) | [operation on Named] |
| Returns the name of this type as a scoped tuple | |
| set_name (<i>self</i> , <i>name</i>) | [operation on Named] |
| Changes the name of this type | |
| Base | [class] |
| parents: parent class Class for base types | |
| __init__ (<i>self</i> , <i>language</i> , <i>name</i>) | [operation on Base] |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Base] |

`--cmp--(self, other)` [operation on Base]
Comparison operator

`--str--(self)` [operation on Base]

Dependent [class]

parents: parent class Class for template dependent types

`--init--(self, language, name)` [operation on Dependent]

`accept(self, visitor)` [operation on Dependent]

`--cmp--(self, other)` [operation on Dependent]
Comparison operator

`--str--(self)` [operation on Dependent]

Unknown [class]

parents: parent class Class for not (yet) known type

`--init--(self, language, name)` [operation on Unknown]

`link(self)` [operation on Unknown]
external reference this type may be associated with

`resolve(self, language, name, link)` [operation on Unknown]
associate this type with an external reference, instead of a declaration

`accept(self, visitor)` [operation on Unknown]

`--cmp--(self, other)` [operation on Unknown]
Comparison operator

`--str--(self)` [operation on Unknown]

Declared [class]

parents: parent class Class for declared types

`--init--(self, language, name, declaration)` [operation on Declared]

`declaration(self)` [operation on Declared]
declaration object which corresponds to this type

`accept(self, visitor)` [operation on Declared]

`--cmp--(self, other)` [operation on Declared]
Comparison operator

`--str--(self)` [operation on Declared]

Template [class]

parents: parent class Class for declared parametrized types

`--init--(self, language, name, declaration, parameters)` [operation on Template]

`parameters(self)` [operation on Template]
list of type names used to declare this template

`accept(self, visitor)` [operation on Template]

`--cmp--(self, other)` [operation on Template]
Comparison operator

`--str--(self)` [operation on Template]

Modifier [class]

parents: parent class Class for alias types with modifiers (such as 'const', '&', etc.)

`--init--(self, language, alias, premod, postmod)` [operation on Modifier]

`alias(self)` [operation on Modifier]
the type this type refers to

`premod(self)` [operation on Modifier]
the modifier string

`postmod(self)` [operation on Modifier]
the modifier string

`accept(self, visitor)` [operation on Modifier]

`set_alias(self, alias)` [operation on Modifier]

`--cmp--(self, other)` [operation on Modifier]
Comparison operator

`--str--(self)` [operation on Modifier]

Array [class]

parents: parent class a modifier that adds array dimensions to a type

`--init--(self, language, alias, sizes)` [operation on Array]

| | |
|--|----------------------|
| alias (<i>self</i>) | [operation on Array] |
| sizes (<i>self</i>) | [operation on Array] |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Array] |
| set_alias (<i>self</i> , <i>alias</i>) | [operation on Array] |
| __cmp__ (<i>self</i> , <i>other</i>) Comparison operator | [operation on Array] |
| __str__ (<i>self</i>) | [operation on Array] |

Parametrized [class]

parents: parent class Class for parametrized type instances

| | |
|--|-----------------------------|
| __init__ (<i>self</i> , <i>language</i> , <i>template</i> , <i>parameters</i>) | [operation on Parametrized] |
| template (<i>self</i>) template type this is an instance of | [operation on Parametrized] |
| parameters (<i>self</i>) list of types for which this template is instantiated | [operation on Parametrized] |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Parametrized] |
| set_template (<i>self</i> , <i>type</i>) | [operation on Parametrized] |
| __cmp__ (<i>self</i> , <i>other</i>) Comparison operator | [operation on Parametrized] |
| __str__ (<i>self</i>) | [operation on Parametrized] |

Function [class]

parents: parent class Class for function pointer types.

| | |
|---|-------------------------|
| __init__ (<i>self</i> , <i>language</i> , <i>retType</i> , <i>premod</i> , <i>params</i>) | [operation on Function] |
| returnType (<i>self</i>) nested return type | [operation on Function] |
| premod (<i>self</i>) list of premodifier strings | [operation on Function] |

| | |
|--|---------------------------|
| parameters (<i>self</i>) | [operation on Function] |
| list of nested parameter types | |
| accept (<i>self</i> , <i>visitor</i>) | [operation on Function] |
| set_returnType (<i>self</i> , <i>returnType</i>) | [operation on Function] |
| Dictionary | [class] |
| __init__ (<i>self</i>) | [operation on Dictionary] |
| __setitem__ (<i>self</i> , <i>name</i> , <i>type</i>) | [operation on Dictionary] |
| __getitem__ (<i>self</i> , <i>name</i>) | [operation on Dictionary] |
| __delitem__ (<i>self</i> , <i>name</i>) | [operation on Dictionary] |
| has_key (<i>self</i> , <i>name</i>) | [operation on Dictionary] |
| keys (<i>self</i>) | [operation on Dictionary] |
| values (<i>self</i>) | [operation on Dictionary] |
| items (<i>self</i>) | [operation on Dictionary] |
| lookup (<i>self</i> , <i>name</i> , <i>scopes</i>) | [operation on Dictionary] |
| locate 'name' in one of the scopes | |
| clear (<i>self</i>) | [operation on Dictionary] |
| merge (<i>self</i> , <i>dict</i>) | [operation on Dictionary] |
| Visitor | [class] |
| Visitor for Type objects | |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitArray (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitTemplate (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Visitor] |
| visitDependent (<i>self</i> , <i>type</i>) | [operation on Visitor] |

2.3 The Util Module

slashName(*scopedName*, *our_scope*) [function]

slashName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by '/' characters. If a second list is given, remove a common prefix using pruneScope().

dotName(*scopedName*, *our_scope*) [function]

dotName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by '.' characters. If a second list is given, remove a common prefix using pruneScope().

ccolonName(*scopedName*, *our_scope*) [function]

ccolonName(list, [list]) -> string

Return a scoped name given as a list of strings as a single string with the components separated by '::' strings. If a second list is given, remove a common prefix using pruneScope().

pruneScope(*target_scope*, *our_scope*) [function]

pruneScope(list A, list B) -> list

Given two lists of strings (scoped names), return a copy of list A with any prefix it shares with B removed.

e.g. pruneScope(['A', 'B', 'C', 'D'], ['A', 'B', 'D']) -> ['C', 'D']

escapifyString(*str*) [function]

escapifyString(string) -> string

Return the given string with any non-printing characters escaped.

_import(*name*) [function]

import either a module, or a file.

import_object(*spec*, *defaultAttr*, *basePackage*) [function]

Imports an object according to 'spec'. spec must be either a string or a tuple of two strings. A tuple of two strings means load the module from the first string, and look for an attribute using the second string. One string is interpreted according to the optional arguments. The default is just to load the named module. 'defaultAttr' means to look for the named attribute in the module and return that. 'basePackage' means to prepend the named string to the spec before importing. Note that you may pass a list instead of a tuple, and it will have the same effect.

This is used by the HTML formatter for example, to specify page classes. Each class is in a separate module, and each module has a htmlPageAttr attribute that references the class of the Page for that module. This avoids the need to specify a list of default pages, easing maintainability.

- splitAndStrip(*line*)** [function]
Splits a line at the first space, then strips the second argument
- open(*filename*)** [function]
open a file, generating all intermediate directories if needed
- getopt_spec(*args, options, long_options*)** [function]
Transparently add `-spec=file` support to `getopt`
- quote(*name*)** [function]
Quotes a base filename to remove illegal characters and keep it within a reasonable length for the filesystem.
The md5 hash function is used if the length of the name after quoting is more than 100 characters. If it is used, then as many characters at the start of the name as possible are kept intact, and the hash appended to make 100 characters.
Do not pass filenames with meaningful extensions to this function, as the hash could destroy them.

PyWriter [class]

A class that allows writing data in such a way that it can be read in by just 'exec'ing the file. You should extend it and override `write_item()`

- __init__(*self, ostream*)** [operation on PyWriter]
- indent(*self*)** [operation on PyWriter]
- outdent(*self*)** [operation on PyWriter]
- ensure_import(*self, module, names*)** [operation on PyWriter]
- ensure_struct(*self*)** [operation on PyWriter]
- write_top(*self, str*)** [operation on PyWriter]
Writes a string to the top of the file
- write(*self, str*)** [operation on PyWriter]
- write_item(*self, item*)** [operation on PyWriter]
Writes arbitrary items by looking up `write_Foo` functions where `Foo` is the class name of the item
- flush(*self*)** [operation on PyWriter]
Writes the buffer to the stream and closes the buffer
- long(*self, list*)** [operation on PyWriter]
Remembers list as wanting 'long' representation (an item per line)

write_list(*self*, *list*) [operation on PyWriter]
Writes a list on one line. If long(list) was previously called, the list from its cache and calls write_long_list

write_long_list(*self*, *list*) [operation on PyWriter]
Writes a list with each item on a new line

write_attr(*self*, *name*, *value*) [operation on PyWriter]

flatten_struct(*self*, *struct*) [operation on PyWriter]
Flattens a struct into a (possibly nested) list. A struct is an object with only the following members: numbers, strings, sub-structs, lists and tuples.

write_PyWriterStruct(*self*, *struct*) [operation on PyWriter]

PyWriterStruct [class]
A utility class that PyWriter uses to dump class objects. Dict is the dictionary of the class being dumped.

__init__(*self*, *dict*) [operation on PyWriterStruct]

3 The Parser Module

Synopsis provides a set of parsers that generate an Abstract Syntax Tree from various programming languages. They are usually part of third party projects, such that we only need to provide a thin wrapper to bind them to the synopsis core.

3.1 The C++ Module

| | |
|---|-----------------------------|
| emul | [module] |
| get_temp_file() | [function] |
| Returns the temporary filename. The temp file is created, but is left empty | |
| cleanup_temp_file() | [function] |
| Removes the temporary file and resets the filename | |
| main() | [function] |
| The main function - parses the arguments and controls the program | |
| get_fallback(<i>preferred, is_first_time</i>) | [function] |
| Tries to return info from a fallback compiler, and prints a warning message to the user, unless their preferred compiler was 'none' | |
| get_compiler_info(<i>compiler</i>) | [function] |
| Returns the compiler info for the given compiler. The info is returned as a CompilerInfo object, or None if the compiler isn't found. | |
| load_compiler_infos() | [function] |
| Loads the compiler infos from a file | |
| get_compiler_timestamp(<i>compiler</i>) | [function] |
| Returns the timestamp for the given compiler, or 0 if not found | |
| refresh_compiler_infos(<i>infos</i>) | [function] |
| Refreshes the list of infos, by rediscovering all non-custom compilers in the map. The map is modified in-place. | |
| write_compiler_infos(<i>infos</i>) | [function] |
| find_compiler_info(<i>compiler</i>) | [function] |
| CompilerInfo | [class] |
| Info about one compiler. | |
| __init__(<i>self, compiler, is_custom, timestamp, include_paths, macros</i>) | [operation on CompilerInfo] |
| Synopsis | [Package] |
| Parser | [Package] |

| | |
|---|----------------------|
| C++ | [Package] |
| AST | [namespace] |
| A namespace for the AST hierarchy | |
| Access | [enum] |
| An enumeration of accessibility specifiers | |
| Default | [enumerator] |
| Public | [enumerator] |
| Protected | [enumerator] |
| Private | [enumerator] |
| Reference | [struct] |
| A struct to hold cross-reference info | |
| std::string file | [data member] |
| int line | [data member] |
| ScopedName scope | [data member] |
| std::string context | [data member] |
| Reference Reference() | [member on function] |
| Reference Reference(const std::string & _file, int _line, const ScopedName & _scope, const std::string & _context) | [member on function] |
| Reference Reference(const Reference & other) | [member on function] |
| Reference Reference & operator=(const Reference & other) | [member on function] |
| Comment | [class] |
| parents: parent class Encapsulation of one Comment, which may span multiple lines. Each comment encapsulates one block or a block of // comments on adjacent lines. If extract_tails is set, then comments will be added even when they are not adjacent to a declaration - these comments will be marked as "suspect". Most of these will be discarded by the Linker, unless they have appropriate markings such as "//.< comment for previous decl" | |

```

std::vector< Comment *> vector [typedef]
    A vector of Comments

Comment Comment( [member on function]
    SourceFile * file, int line, const
    std::string & text, bool suspect)
    Constructor

Comment SourceFile * [member on function]
    file()
    Returns the filename of this comment

Comment int line() [member on function]
    Returns the line number of the start of this comment

Comment const std::string & [member on function]
    text()
    Returns the text of this comment

Comment void set_suspect( [member on function]
    bool suspect)
    Sets whether this comment is suspicious

Comment bool is_suspect() [member on function]
    Returns whether this comment is suspicious

SourceFile * m_file [data member]
    The file

int m_line [data member]
    The first line number

std::string m_text [data member]
    The text

bool m_suspect [data member]
    True if this comment is probably not needed. The ex-
    ception is comments which will be used as "tails", eg:
    //.< comment for previous decl

```

Declaration [class]

parents: parent class The base class of the Declaration hierarchy. All declarations have a scoped Name, comments, etc. The filename and type name are constant strings. This is enforced so that the strings will reference the same data, saving both memory and cpu time. For this to work however, you must be careful to use the same strings for constructing the names from, for example from a dictionary.

std::vector< Declaration *> vector [typedef]
A vector of Declaration objects

Declaration Declaration([member on function]
*SourceFile * file*, *int line*, *const*
std::string & type, *const ScopedName &*
name)
Constructor

Declaration ~Declaration() [member on function]
Destructor. Recursively deletes the comments for this
declaration

Declaration void accept([member on function]
*Visitor **)
Accept the given AST::Visitor

Declaration ScopedName [member on function]
& name()
Returns the scoped name of this declaration

Declaration const [member on function]
ScopedName & name()
Constant version of name()

Declaration SourceFile * [member on function]
file()
Returns the filename of this declaration

Declaration void set_file([member on function]
*SourceFile * file*)
Changes the filename of this declaration

Declaration int line() [member on function]
Returns the line number of this declaration

Declaration const [member on function]
std::string & type()
Returns the name of the type (not class) of this decla-
ration

Declaration void set_type([member on function]
const std::string & type)
Change the type name. Currently only used to indicate
template types

Declaration Access [member on function]
access()

Returns the accessability of this declaration

Declaration void [member on function]
set_access(Access axs)

Sets the accessability of this declaration

Declaration const [member on function]
Comment::vector & comments()

Constant version of comments()

Declaration [member on function]
Comment::vector & comments()

Returns the vector of comments. The vector returned is the private member vector of this Declaration, so modifications will affect the Declaration.

Declaration [member on function]
Types::Declared * declared()

Return a cached Types::Declared for this Declaration. It is created on demand and returned every time you call the method on this object.

Declaration const [member on function]
Types::Declared * declared()

Return a cached Types::Declared for this Declaration. It is created on demand and returned every time you call the method on this object. This is the const version.

SourceFile * m_file [data member]
 The filename

int m_line [data member]
 The first line number

std::string m_type [data member]
 The string type name

ScopedName m_name [data member]
 The scoped name

Comment::vector m_comments [data member]
 The vector of Comment objects

Access m_access [data member]
The accessibility spec

Types::Declared * m_declared [data member]
The Types::Declared cache

Include [class]

parents: parent class Information about an `#include` or `#include_next` directive. This object is a thin wrapper around the target `SourceFile`, with some attributes to indicate whether the directive included a macro expansion, and whether it was an `#include_next` or not.

A macro expansion is flagged because often you don't want to include those in an include graph. For example, some headers in the Boost PP library will include other files multiple times, where the file included is given by a macro. It is rare that you actually want to show this macro-dependent include in the documentation or in a graph.

std::vector< Include *> vector [typedef]
A vector of Includes

Include Include([member on function]
`SourceFile * target, bool is_macro, bool is_next`)
Constructor

Include SourceFile * [member on function]
`target()`
Returns the target of this include

Include bool is_macro() [member on function]
Returns whether the include filename was a macro expansion

Include bool is_next() [member on function]
Returns whether the include was an `#include_next` directive

SourceFile * m_target [data member]
The target file of the include or `include_next`

bool m_is_macro [data member]
Whether the include filename was a macro expansion

bool m_is_next [data member]
Whether the include was an `#include_next` directive

SourceFile [class]

parents: parent class Information about a source file used to generate the AST.

Generally an AST will include SourceFile objects for **all** files, including headers, that were used. The difference is that the main files (those named to the parser) are flagged as "main", and others will not have lists of declarations.

std::vector< SourceFile *> vector [typedef]
A vector of SourceFiles

SourceFile SourceFile([member on function]
const std::string & filename, **const**
std::string & full_filename, **bool is_main**)
Constructor

SourceFile const [member on function]
std::string & filename()
Returns the filename of this SourceFile (may be stripped by a basename)

SourceFile const [member on function]
std::string & full_filename()
Returns the full filename of this SourceFile

SourceFile bool is_main() [member on function]
Returns whether this is a main file (as opposed to extra included file just being stored for its list of includes)

SourceFile [member on function]
Declaration::vector & declarations()
Returns the vector of declarations in this file

SourceFile const [member on function]
Declaration::vector & declarations()
Returns a const vector of declarations in this file

SourceFile Include::vector [member on function]
& includes()
Returns a vector of includes in this file

SourceFile const [member on function]
Include::vector & includes()
 Returns a const vector of includes in this file

std::string m_filename [data member]
 The filename

std::string m_full_filename [data member]
 The full filename

bool m_is_main [data member]
 Whether this file is a main file

Declaration::vector [data member]
m_declarations
 The vector of declarations

Include::vector m_includes [data member]
 The vector of includes

Macro [class]

parents: parent class Encapsulates a preprocessor macro. Macros are stored in the AST, but since they are not regular C++ syntax they are treated specially: They will be in order compared to other macros, but not to the rest of the AST since the preprocessing stage occurs first. They will always be in the global scope. Note that the parameters is a pointer to a vector - if the pointer is null, then the macro is not function-like. If the pointer is non-null then it points to a vector of parameter names. If the macro is function-like but with no parameters, it is a pointer to an empty vector.

std::vector<std::string> Parameters [typedef]
 The type of the parameters

Macro Macro(SourceFile * [member on function]
file, int line, const ScopedName & name,
Parameters * params, const std::string &
text)
 Constructor. Assumes ownership of the Parameters vector if it is not a null pointer.

Macro ~Macro() [member on function]
 Destructor

Macro **const Parameters *** [member on function]
parameters()

The parameters of the macro. May be a null pointer if the macro is not function-like

Macro **const std::string &** [member on function]
text()

The expansion text of the macro.

Macro **void accept(Visitor *** [member on function]
***)**

Accepts the given visitor

Parameters * m_parameters [data member]
The parameters

std::string m_text [data member]
The expansion text

Scope [class]

parents: parent class Base class for scopes with contained declarations. Each scope has its own Dictionary of names so far accumulated for this scope. Each scope also as a complete vector of scopes where name lookup is to proceed if unsuccessful in this scope. Name lookup is not recursive.

Scope **Scope(SourceFile *** [member on function]
file, int line, const std::string & type,
const ScopedName & name)

Constructor

Scope **~Scope()** [member on function]

Destructor. Recursively destroys contained declarations

Scope **void accept(Visitor *** [member on function]
***)**

Accepts the given visitor

Scope **const** [member on function]
Declaration::vector & declarations()

Constant version of declarations()

Scope Declaration::vector [member on function]
 & declarations()

Returns the vector of declarations. The vector returned is the private member vector of this Scope, so modifications will affect the Scope.

Declaration::vector [data member]
m_declarations

The vector of contained declarations

Namespace [class]

parents: parent class Namespace class

Namespace Namespace([member on function]
 SourceFile * file, int line, const
 std::string & type, const ScopedName &
 name)

Constructor

Namespace ~Namespace() [member on function]
 Destructor

Namespace void accept([member on function]
 Visitor *)

Accepts the given AST::Visitor

Inheritance [class]

Inheritance class. This class encapsulates the information about an inheritance, namely its accessibility. Note that classes inherit from types, not class declarations. As such it's possible to inherit from a parameterized type, or a declared typedef or class/struct.

std::vector< Inheritance *> vector [typedef]
 A vector of Inheritance objects

std::vector<std::string> Attributes [typedef]
 A typedef of the Attributes type

Inheritance Inheritance([member on function]
 Types::Type * parent, const Attributes &
 attributes)

Constructor

Inheritance **void** **accept**([member on function]
 Visitor *)
 Accepts the given AST::Visitor

Inheritance **Types::Type** [member on function]
 ***** **parent**()
 Returns the Class object this inheritance refers to. The method returns a Type since typedefs to classes are preserved to enhance readability of the generated docs. Note that the parent may also be a non-declaration type, such as vector<int>

Inheritance **const** [member on function]
 Attributes & **attributes**()
 Returns the attributes of this inheritance

Types::Type ***** **m_parent** [data member]
 The parent class or typedef to class

Attributes **m_attrs** [data member]
 The attributes

Class [class]

parents: parent class Class class

Class **Class**(**SourceFile *** [member on function]
 file, **int** **line**, **const** **std::string &** **type**,
 const **ScopedName &** **name**)
 Constructor

Class **~Class**() [member on function]
 Destructor. Recursively destroys Inheritance objects

Class **void** **accept**(**Visitor** [member on function]
 *****)
 Accepts the given AST::Visitor

Class **const** [member on function]
 Inheritance::vector & **parents**()
 Constant version of parents()

Class **Inheritance::vector** [member on function]
 & **parents**()
 Returns the vector of parent Inheritance objects. The vector returned is the private member vector of this Class, so modifications will affect the Class.

Class **Types::Template *** [member on function]
template_type()

Returns the Template object if this is a template

Class void [member on function]
set_template_type(Types::Template * type)

Sets the Template object for this class. NULL means not a template

Inheritance::vector m_parents [data member]
The vector of parent Inheritance objects

Types::Template * m_template [data member]
The Template Type for this class if it's a template

Forward [class]

parents: parent class Forward declaration. Currently this has no extra attributes.

Forward Forward([member on function]
SourceFile * file, int line, const
std::string & type, const ScopedName &
name)

Constructor

Forward Forward([member on function]
Declaration * decl)

Constructor that copies an existing declaration

Forward void accept([member on function]
Visitor *)

Accepts the given AST::Visitor

Forward Types::Template [member on function]
*** template_type()**

Returns the Template object if this is a template

Forward void [member on function]
set_template_type(Types::Template * type)

Sets the Template object for this class. NULL means not a template

Types::Template * m_template [data member]
The Template Type for this forward if it's a template

Typedef [class]

parents: parent class Typedef declaration

```
Typedef Typedef( [member on function]
    SourceFile * file, int line, const
    std::string & type, const ScopedName &
    name, Types::Type * alias, bool constr)
    Constructor
```

```
Typedef ~Typedef() [member on function]
    Destructor
```

```
Typedef void accept( [member on function]
    Visitor * )
    Accepts the given AST::Visitor
```

```
Typedef Types::Type * [member on function]
    alias()
    Returns the Type object this typedef aliases
```

```
Typedef bool constructed() [member on function]
    Returns true if the Type object was constructed inside
    the typedef
```

```
Types::Type * m_alias [data member]
    The alias Type
```

```
bool m_constr [data member]
    True if constructed
```

Variable [class]

parents: parent class Variable declaration

```
std::vector<size_t> Sizes [typedef]
    The type of the vector of sizes
```

```
Variable Variable( [member on function]
    SourceFile * file, int line, const
    std::string & type, const ScopedName &
    name, Types::Type * vtype, bool constr)
    Constructor
```

```
Variable ~Variable() [member on function]
    Destructor
```

Variable **void** `accept(Visitor *)` [member on function]
 Accepts the given AST::Visitor

Variable **Types::Type *** `vtype()` [member on function]
 Returns the Type object of this variable

Variable **bool** `constructed()` [member on function]
 Returns true if the Type object was constructed inside the variable

Variable **Sizes &** `sizes()` [member on function]
 Returns the array sizes vector

Types::Type * m_vtype [data member]
 The variable Type

bool m_constr [data member]
 True if constructed

Sizes m_sizes [data member]
 Vector of array sizes. zero length indicates not an array.

Enumerator [class]

parents: parent class Enumerator declaration. This is a name with a value in the containing scope. Enumerators only appear inside Enums via their `enumerators()` attribute.

std::vector< Enumerator *> vector [typedef]
 Type of a vector of Enumerator objects

Enumerator `Enumerator(SourceFile * file, int line, const std::string & type, const ScopedName & name, const std::string & value)` [member on function]
 Constructor

Enumerator void `accept(Visitor *)` [member on function]
 Accept the given AST::Visitor

Enumerator const `std::string & value()` [member on function]
 Returns the value of this enumerator

`std::string m_value` [data member]
The value of this enumerator

Enum [class]

parents: parent class Enum declaration. An enum contains multiple enumerators.

Enum Enum(SourceFile * [member on function]
file, int line, const std::string & type,
const ScopedName & name)
Constructor

Enum ~Enum() [member on function]
Destructor. Recursively destroys Enumerators

Enum void accept(Visitor * [member on function]
)
Accepts the given AST::Visitor

Enum Enumerator::vector [member on function]
& enumerators()
Returns the vector of Enumerators

Enumerator::vector m_enums [data member]
The vector of Enumerators

Const [class]

parents: parent class A const is a name with a value and declared type.

Const Const(SourceFile * [member on function]
file, int line, const std::string & type,
const ScopedName & name, Types::Type *
ctype, const std::string & value)
Constructor

Const void accept(Visitor [member on function]
*)
Accept the given AST::Visitor

Const Types::Type * [member on function]
ctype()
Returns the Type object of this const

Const const std::string & [member on function]
 value()

Returns the value of this enumerator

Types::Type * m_ctype [data member]
 The const Type

std::string m_value [data member]
 The value of this enumerator

Parameter [class]

parents: parent class Parameter encapsulates one parameter to a function

std::vector<std::string> Mods [typedef]
 The type of modifiers such as 'in', 'out'

std::vector< Parameter *> vector [typedef]
 A vector of Parameter objects

Parameter **Parameter**(const [member on function]
 Mods & pre, Types::Type * type, const Mods
 & post, const std::string & name, const
 std::string & value)
 Constructor

Parameter **~Parameter()** [member on function]
 Destructor

Parameter **void** **accept**([member on function]
 Visitor *)
 Accept the given AST::Visitor. Note this is not derived from Declaration so it is not a virtual method.

Parameter **Mods &** [member on function]
 premodifier()
 Returns the premodifier

Parameter **Mods &** [member on function]
 postmodifier()
 Returns the postmodifier

Parameter **Types::Type *** [member on function]
 type()
 Returns the type of the parameter

Parameter **const** [member on function]
Types::Type * type()
 Const version of `type()`

Parameter **const std::string** [member on function]
& name()
 Returns the name of the parameter

Parameter **const std::string** [member on function]
& value()
 Returns the value of the parameter

Parameter **void set_name()** [member on function]
const std::string & name)
 Sets the name of the parameter

Mods **m_pre** [data member]

Mods **m_post** [data member]

Types::Type * m_type [data member]

std::string m_name [data member]

std::string m_value [data member]

Function [class]

parents: parent class `Function` encapsulates a function declaration. Note that names may be stored in mangled form, and formatters should use `realname()` to get the unmangled version. If this is a function template, use the `template_type()` method to get at the template type

std::vector<std::string> Mods [typedef]
 The type of premodifiers

std::vector< Function *> vector [typedef]
 A vector of `Function` objects

Function Function() [member on function]
SourceFile * file, int line, const
std::string & type, const ScopedName &
name, const Mods & premod, Types::Type *
ret, const std::string & realname)
 Constructor

| | | |
|-------------------|--|----------------------|
| Function | <code>~Function()</code> | [member on function] |
| | Destructor. Recursively destroys parameters | |
| Function | <code>void accept(Visitor *)</code> | [member on function] |
| | Accept the given visitor | |
| Function | <code>Mods &premodifier()</code> | [member on function] |
| | Returns the premodifier vector | |
| Function | <code>Types::Type *return_type()</code> | [member on function] |
| | Returns the return Type | |
| Function | <code>const std::string &realname()</code> | [member on function] |
| | Returns the real name of this function | |
| Function | <code>Parameter::vector &parameters()</code> | [member on function] |
| | Returns the vector of parameters | |
| Function | <code>Types::Template *template_type()</code> | [member on function] |
| | Returns the Template object if this is a template | |
| Function | <code>void set_template_type(Types::Template * type)</code> | [member on function] |
| | Sets the Template object for this class. NULL means not a template | |
| Mods | <code>m_pre</code> | [data member] |
| | The premodifier vector | |
| Types::Type * | <code>m_ret</code> | [data member] |
| | The return type | |
| std::string | <code>m_realname</code> | [data member] |
| | The real (unmangled) name | |
| Parameter::vector | <code>m_params</code> | [data member] |
| | The vector of parameters | |

Types::Template * m_template [data member]
The Template Type for this class if it's a template

Operation [class]

parents: parent class Operations are similar to functions but Not Quite Right

Operation **Operation**([member on function]
SourceFile * *file*, int *line*, const
std::string & *type*, const ScopedName &
name, const Function::Mods & *modifiers*,
Types::Type * *ret*, const std::string &
realname)
Constructor

Operation **void** **accept**([member on function]
Visitor *)
Accept the given visitor

Visitor [class]

The Visitor for the AST hierarchy. This class is just an interface really. It is abstract, and you must reimplement any methods you want. The default implementations of the methods call the visit methods for the subclasses of the visited type, eg visit_namespace calls visit_scope which calls visit_declaration.

Visitor ~**Visitor**() [member on function]

Visitor **void** [member on function]
visit_declaration(Declaration *)

Visitor **void** **visit_macro**([member on function]
Macro *)

Visitor **void** **visit_scope**([member on function]
Scope *)

Visitor **void** [member on function]
visit_namespace(Namespace *)

Visitor **void** **visit_class**([member on function]
Class *)

Visitor **void** [member on function]
visit_inheritance(Inheritance *)

| | |
|---|----------------------|
| Visitor void visit_forward(| [member on function] |
| Forward *) | |
| Visitor void visit_typedef(| [member on function] |
| Typedef *) | |
| Visitor void | [member on function] |
| visit_variable(Variable *) | |
| Visitor void visit_const(| [member on function] |
| Const *) | |
| Visitor void visit_enum(| [member on function] |
| Enum *) | |
| Visitor void | [member on function] |
| visit_enumerator(Enumerator *) | |
| Visitor void | [member on function] |
| visit_function(Function *) | |
| Visitor void | [member on function] |
| visit_operation(Operation *) | |
| Visitor void | [member on function] |
| visit_parameter(Parameter *) | |
| std::vector< ScopeInfo *> ScopeSearch | [typedef] |
| NamespaceType | [enum] |
| Enumeration of namespace types for use in Builder::start_namespace() | |
| NamespaceNamed | [enumerator] |
| NamespaceAnon | [enumerator] |
| < Normal, named, namespace. name is its given name | |
| NamespaceUnique | [enumerator] |
| < An anonymous namespace. name is the filename | |
| NamespaceTemplate | [enumerator] |
| < A unique namespace. name is the type (for, while, etc.) | |
| Builder | [class] |
| AST Builder. This class manages the building of an AST, including queries on the existing AST such as name and type lookups. The building functions are called by SWalker as it walks the parse tree. | |

Builder **Builder**([member on function]
 AST::SourceFile * file)
 Constructor

Builder **~Builder**()[member on function]
 Destructor. Recursively destroys all AST objects

Builder **void** **set_swalker**([member on function]
 SWalker * swalker)
 Sets the swalker

Builder **void** **set_access**([member on function]
 AST::Access)
 Changes the current accessability for the current scope

Builder **AST::SourceFile ***[member on function]
 file()
 Returns the current file

Builder **void** **set_file**([member on function]
 AST::SourceFile *)
 Changes the current file

Builder **const**[member on function]
 AST::Declaration::vector & builtin_decls()
 Returns the list of builtin decls ("_null_t", "true", etc.)

Builder **AST::Scope *** **scope**()[member on function]
 Returns the current scope

Builder **ScopeInfo *** **scopeinfo**()[member on function]

 Returns the current ScopeInfo for the current Scope

Builder **AST::Scope *** **global**()[member on function]
 Returns the global scope

Builder **Builder::Lookup ***[member on function]
 lookup()
 Returns the Lookup object for the builder

Builder **void** **add**([member on function]
 AST::Declaration * declaration, **bool**
 is_template)
 Add the given Declaration to the current scope. If *is_template* is true, then it is added to the parent of the current scope, assuming that the current scope is the temporary template scope

Builder **void** `add(Types::Named * named)` [member on function]

Add the given non-declaration type to the current scope

Builder **void** `add_macros(const std::vector< AST::Macro *> &)` [member on function]

Adds the given Macros to the global scope. This method should only be called once, with the macros stored in order from the preprocessing stage.

Builder **AST::Namespace *** `start_namespace(const std::string & name, NamespaceType type)` [member on function]

Construct and open a new Namespace. The Namespace becomes the current scope, and the old one is pushed onto the stack. If name is empty then a unique name is generated of the form 'ns1

Builder **void** `end_namespace()` [member on function]

End the current namespace and pop the previous Scope off the stack

Builder **AST::Namespace *** `start_template()` [member on function]

Starts a new template namespace

Builder **void** `end_template()` [member on function]

End the current template namespace

Builder **AST::Class *** `start_class(int , const std::string & type, const std::string & name, AST::Parameter::vector * templ_params)` [member on function]

Construct and open a new Class. The Class becomes the current scope, and the old one is pushed onto the stack. The type argument is the type, ie: "class" or "struct". This is tested to determine the default accessibility. If this is a template class, the *templ_params* vector must be non-null pointer

Builder **AST::Class *** `start_class(int , const std::string & type, const ScopedName & names)` [member on function]

Construct and open a new Class with a qualified name

Builder void `update_class_base_search()` [member on function]
`update_class_base_search()`

Update the search to include base classes. Call this method after `startClass()`, and after filling in the `parents()` vector of the returned `AST::Class` object. After calling this method, name and type lookups will correctly search the base classes of this class.

Builder void `end_class()` [member on function]

End the current class and pop the previous Scope off the stack

Builder void `start_function_impl(const ScopedName & name)` [member on function]

`start_function_impl(const ScopedName & name)`
 Start function impl scope

Builder void `end_function_impl()` [member on function]

End function impl scope

Builder AST::Function * `add_function(int, const std::string & name,` [member on function]
`const std::vector<std::string> & premod,`
`Types::Type * ret, const std::string & realname,`
`AST::Parameter::vector * templ_params)`

`add_function(int, const std::string & name, const std::vector<std::string> & premod, Types::Type * ret, const std::string & realname, AST::Parameter::vector * templ_params)`
 Add an function

Builder AST::Variable * `add_variable(int, const std::string & name,` [member on function]
`Types::Type * vtype, bool constr, const`
`std::string & type)`

`add_variable(int, const std::string & name, Types::Type * vtype, bool constr, const std::string & type)`
 Add a variable

Builder void `add_this_variable()` [member on function]

Add a variable to represent 'this', iff we are in a method

Builder AST::Typedef * `add_typedef(int, const std::string & name,` [member on function]
`Types::Type * alias, bool constr)`

`add_typedef(int, const std::string & name, Types::Type * alias, bool constr)`
 Add a typedef

Builder AST::Enumerator * `add_enumerator(int, const std::string & name,` [member on function]
`const std::string & value)`

`add_enumerator(int, const std::string & name, const std::string & value)`
 Add an enumerator

Builder **AST::Enum** * `add_enum(` [member on function]
`int , const std::string & name, const`
`AST::Enumerator::vector &)`

Add an enum

Builder **AST::Declaration** * [member on function]
`add_tail_comment(int line)`

Add a tail comment. This will be a dummy declaration with an empty name and type "dummy"

Builder **void** [member on function]
`add_using_namespace(Types::Named * type)`

Add a namespace using declaration.

Builder **void** [member on function]
`add_aliased_using_namespace(Types::Named * type,`
`const std::string & alias)`

Add a namespace alias using declaration.

Builder **void** [member on function]
`add_using_declaration(Types::Named * type)`

Add a using declaration.

Builder **bool** `mapName(const` [member on function]
`ScopedName & name, std::vector< AST::Scope * > &`
`, Types::Named * &)`

Maps a scoped name into a vector of scopes and the final type. Returns true on success.

Builder **Types::Base** * [member on function]
`create_base(const std::string & name)`

Create a Base type for the given name in the current scope

Builder **Types::Dependent** * [member on function]
`create_dependent(const std::string & name)`

Create a Dependent type for the given name in the current scope

Builder **Types::Unknown** * [member on function]
`create_unknown(const std::string & name)`

Create an Unknown type for the given name in the current scope

Builder **Types::Template** * [member on function]
 create_template(const std::string & *name*, const
 std::vector< Types::Type *> &)
 Create a Template type for the given name in the current
 scope

Builder **Types::Unknown** * [member on function]
 add_unknown(const std::string & *name*)
 Add an Unknown decl for given name if it doesnt already
 exist

Builder **AST::Forward** * [member on function]
 add_forward(int *lineno*, const std::string &
name, AST::Parameter::vector * *templ_params*)
 Add an Templated Forward decl for given name if it doesnt
 already exist

AST::SourceFile * **m_file** [data member]
 Current file

AST::Scope * **m_global** [data member]
 The global scope object

AST::Scope * **m_scope** [data member]
 Current scope object

int **m_unique** [data member]
 A counter used to generate unique namespace names

std::vector< ScopeInfo *> **m_scopes** [data member]
 The stack of Builder::Scopes

Private * **m** [data member]
 Private data which uses map instance

Builder **ScopeInfo** * find_info([member on function]
 AST::Scope *)
 Return a ScopeInfo* for the given Declaration. This method
 first looks for an existing Scope* in the Private map.

Builder **void** add_class_bases([member on function]
 AST::Class * *clas*, ScopeSearch & *search*)
 Utility method to recursively add base classes to given search

Builder **std::string** **dump_search**([member on function]
 ScopeInfo * *scope*)

Formats the search of the given Scope for logging

Builder **void** [member on function]
do_add_using_namespace(ScopeInfo * *target*,
 ScopeInfo * *scope*)

Recursively adds 'target' as using in 'scope'

SWalker * **m_swalker** [data member]
 A pointer to the SWalker. This is set explicitly by the
 SWalker during its constructor (which takes a Builder).

Builder::Lookup * **m_lookup** [data member]
 A pointer to the Lookup

Private [struct]

std::map< AST::Scope *, ScopeInfo *> [typedef]
ScopeMap

ScopeMap **map** [data member]

std::map<ScopedName, [typedef]
std::vector<AST::Reference>> RefMap

A map of name references

RefMap **refs** [data member]

Declaration::vector [data member]
builtin_decls

A list of builtin declarations

EqualScope [class]

AST::Scope * **target** [data member]

EqualScope EqualScope([member on function]
 ScopeInfo * *t*)

EqualScope **bool** **operator**()([member on function]
 ScopeInfo * *s*)

std::vector<std::string> ScopedName [typedef]

A scoped name, containing zero or more elements. This typedef
 makes it easier to use scoped name types, and also makes it clearer
 than using the raw vector in your code.

`std::ostream & operator<<(std::ostream & out, [function]
const ScopedName & name)`
 Prototype for scoped name output operator (defined in builder.cc)

`std::string join(const ScopedName & strs, const [function]
std::string & sep)`
 Joins the elements of the scoped name using the separator string

`std` [namespace]

`char_traits<unsigned char>` [struct]
 A specialization of the `char_traits` for unsigned char

`unsigned char char_type` [typedef]

`int int_type` [typedef]

`std::streampos pos_type` [typedef]

`std::streamoff off_type` [typedef]

`mbstate_t state_type` [typedef]

`char_traits<unsigned [member on function]
char> void assign(char_type & __c1, const
char_type & __c2)`

`char_traits<unsigned [member on function]
char> bool eq(const char_type & __c1,
const char_type & __c2)`

`char_traits<unsigned [member on function]
char> bool lt(const char_type & __c1,
const char_type & __c2)`

`char_traits<unsigned [member on function]
char> int compare(const char_type * __s1,
const char_type * __s2, size_t __n)`

`char_traits<unsigned [member on function]
char> size_t length(const char_type * __s)`

`char_traits<unsigned [member on function]
char> const char_type * find(const
char_type * __s, size_t __n, const
char_type & __a)`

```

char_traits<unsigned char> char_type * move( char_type * __s1,
const char_type * __s2, size_t __n) [member on function]

char_traits<unsigned char> char_type * copy( char_type * __s1,
const char_type * __s2, size_t __n) [member on function]

char_traits<unsigned char> char_type * assign( char_type *
__s, size_t __n, char_type __a) [member on function]

char_traits<unsigned char> char_type to_char_type( const
int_type & __c) [member on function]

char_traits<unsigned char> int_type to_int_type( const
char_type & __c) [member on function]

char_traits<unsigned char> bool eq_int_type( const int_type &
__c1, const int_type & __c2) [member on function]

char_traits<unsigned char> int_type eof() [member on function]

char_traits<unsigned char> int_type not_eof( const int_type &
__c) [member on function]

__locale_t __c_locale [typedef]

int __convert_from_v( char * __out, const [function]
int __size, const char * __fmt, const char * __v,
const __c_locale & __cloc, int __prec)

std::basic_string<unsigned char > code [typedef]
A string type for the encoded names and types

code::iterator code_iter [typedef]
A string iterator type for the encoded names and types

code make_code( const char * c) [function]
A function to make a code string from a normal string

```

`std::ostream & operator<<(std::ostream & o, [function]
const code & s)`

Insertion operator for encoded names and types

Decoder [class]

Decoder for OCC encodings. This class can be used to decode the names and types encoded by OCC for function and variable types and names.

`Decoder Decoder(Builder *) [member on function]
Constructor`

`Decoder code toCode(char *) [member on function]
Convert a char* to a 'code' type`

`Decoder void init(char *) [member on function]
Initialise the type decoder`

`Decoder code_iter & iter() [member on function]
Returns the iterator used in decoding`

`Decoder Types::Type * [member on function]
decodeType()
Return a Type object from the encoded type.`

`Decoder Types::Type * [member on function]
decodeQualType()
Decodes a Qualified type. iter must be just after the Q`

`Decoder Types::Type * [member on function]
decodeTemplate()
Decodes a Template type. iter must be just after the T`

`Decoder Types::Type * [member on function]
decodeFuncPtr(std::vector<std::string> &)
Decodes a FuncPtr type. iter must be just after the F. The vector is the postmod - if it contains a * then it will be removed and given to the funcptr instead`

`Decoder std::string decodeName() [member on function]
Decode a name`

`Decoder ScopedName [member on function]
decodeQualified()
Decode a qualified name`

Decoder **std::string** decodeName([member on function]
code_iter)

Decode a name starting from the given iterator. Note the iterator passed need not be from the currently decoding string since this is a simple method.

Decoder **std::string** decodeName([member on function]
char *)

Decode a name starting from the given char*

Decoder **void** decodeQualName([member on function]
ScopedName & names)

Decode a qualified name with only names in it

Decoder **bool** isName(char * ptr) [member on function]

Returns true if the char* is pointing to a name (that starts with a length). This is needed since char can be signed or unsigned, and explicitly casting to one or the other is ugly

code **m_string** [data member]
The encoded type string currently being decoded

code_iter **m_iter** [data member]
The current position in m_enc_iter

Builder * **m_builder** [data member]
The builder

Lookup * **m_lookup** [data member]
The lookup

Dictionary [class]

parents: parent class Dictionary of named declarations with lookup. This class maintains a dictionary of names, which index types, supposedly declared in the scope that has this dictionary. There may be only one declaration per name, except in the case of function names.

Dictionary Dictionary() [member on function]
Constructor

Dictionary ~Dictionary() [member on function]
Destructor

std::vector< Types::Named *> Type_vector [typedef]
 The type of multiple entries. We don't want to include type.hh just for this, so this is a workaround

MultipleError [struct]
 Exception thrown when multiple declarations are found when one is expected. The list of declarations is stored in the exception.

Type_vector types [data member]

KeyError [struct]
 Exception thrown when a name is not found in lookup*()

KeyError **KeyError**(const [member on function]
 std::string & n)
 Constructor

std::string name [data member]
 The name which was not found

Dictionary bool **has_key**(const [member on function]
 std::string & name)
 Returns true if name is in dict

Dictionary Types::Named * [member on function]
lookup(const std::string & name)
 Lookup a name in the dictionary. If more than one declaration has this name then an exception is thrown.

Dictionary Type_vector [member on function]
lookupMultiple(const std::string & name)
 Lookup a name in the dictionary expecting multiple decls. Use this method if you expect to find more than one declaration, eg importing names via a using statement.

Dictionary void **insert**([member on function]
 AST::Declaration * decl)
 Add a declaration to the dictionary. The name() is extracted from the declaration and its last string used as the key. The declaration is stored as a Type::Declared which is created inside this method.

Dictionary void **insert**([member on function]
 Types::Named * named)
 Add a named type to the dictionary. The name() is extracted from the type and its last string used as they key.

Dictionary **void** dump() [member on function]
Dump the contents for debugging

Data * **m** [data member]
The private data. This is a forward declared * to speed compilation since std::map is a large template.

Data [struct]
Define nested class

name_map **map** [data member]

TypeFormatter [class]
parents: parent class Formats Types in a way suitable for output

TypeFormatter TypeFormatter() [member on function]

TypeFormatter **void** push_scope([member on function]
const ScopedName & scope)
Sets the current scope, pushing the previous onto a stack

TypeFormatter **void** pop_scope() [member on function]
Pops the previous scope from the stack

TypeFormatter **std::string** [member on function]
format(const Types::Type * , const std::string *
* id)
Returns a formatter string for given type. The option string pointer refers to the parameter name (where applicable) so that it can be put in the right place for function pointer types. The pointed to pointer will be set to NULL if the identifier is used

TypeFormatter **void** visit_type([member on function]
Types::Type *)

TypeFormatter **void** [member on function]
visit_unknown(Types::Unknown *)

TypeFormatter **void** [member on function]
visit_modifier(Types::Modifier *)

TypeFormatter **void** visit_named([member on function]
Types::Named *)

TypeFormatter **void** visit_base([member on function]
Types::Base *)

TypeFormatter **void** [member on function]
 visit_declared(Types::Declared *)

TypeFormatter **void** [member on function]
 visit_template_type(Types::Template *)

TypeFormatter **void** [member on function]
 visit_parameterized(Types::Parameterized *)

TypeFormatter **void** [member on function]
 visit_func_ptr(Types::FuncPtr *)

std::string **m_type** [data member]
 The Type String

ScopedName **m_scope** [data member]
 The current scope name

TypeFormatter **std::string** [member on function]
 colonate(const ScopedName & name)
 Returns the given Name relative to the current scope

std::vector<ScopedName> **m_scope_stack** [data member]
 A stack of previous scopes

const std::string * * **m_fptr_id** [data member]
 A pointer to the identifier for function pointers

Dumper [class]
 parents: parent class parent class Dumper displays the AST to the screen

Dumper Dumper() [member on function]

Dumper **void** onlyShow(const [member on function]
 std::string & fname)
 Sets to only show decls with given filename

Dumper **std::string** formatParam([member on function]
 AST::Parameter *)

Dumper **void** visit(const [member on function]
 std::vector< AST::Declaration *> &)

Dumper **void** visit(const [member on function]
 std::vector< AST::Comment *> &)

| | |
|--|----------------------|
| Dumper void visit_macro(AST::Macro *) | [member on function] |
| Dumper void visit_declaration(AST::Declaration *) | [member on function] |
| Dumper void visit_scope(AST::Scope *) | [member on function] |
| Dumper void visit_namespace(AST::Namespace *) | [member on function] |
| Dumper void visit_forward(AST::Forward *) | [member on function] |
| Dumper void visit_class(AST::Class *) | [member on function] |
| Dumper void visit_function(AST::Function *) | [member on function] |
| Dumper void visit_variable(AST::Variable *) | [member on function] |
| Dumper void visit_typedef(AST::Typedef *) | [member on function] |
| Dumper void visit_enum(AST::Enum *) | [member on function] |
| Dumper void visit_enumerator(AST::Enumerator *) | [member on function] |
| int m_indent The indent depth | [data member] |
| std::string m_indent_string The indent string | [data member] |
| Dumper void indent() Increases indent | [member on function] |
| Dumper void undent() Decreases indent | [member on function] |
| std::string m_filename Only show this filename, if set | [data member] |

| | |
|--|----------------------|
| FakeGC | [namespace] |
| The fake Gargage Collector namespace | |
| cleanup | [struct] |
| Base class of objects that will be cleaned up | |
| cleanup * cleanup_next | [data member] |
| A pointer to the next node in the list | |
| cleanup cleanup() | [member on function] |
| Constructorinline constructor for efficiency | |
| cleanup ~cleanup() | [member on function] |
| Virtual Destructor | |
| cleanup * head | [namespace variable] |
| A pointer to the head of the linked list | |
| void delete_all() | [function] |
| Delete all memory blocks allocated with the GC. *CAUTION* Make sure they are not still in use first! | |
| FileFilter | [class] |
| FileFilter FileFilter() | [member on function] |
| Constructor | |
| FileFilter ~FileFilter() | [member on function] |
| Destructor | |
| FileFilter void set_only_main(bool value) | [member on function] |
| Sets whether only declarations in the main file(s) should be stored | |
| FileFilter void set_main_filename(const char * filename) | [member on function] |
| Sets the main filename | |
| FileFilter void add_extra_filenames(const std::vector<const char *> &) | [member on function] |
| Adds a list of extra filenames to store info for | |

FileFilter void `set_basename(` [member on function]
`const char * basename)`
Sets the basename

FileFilter void [member on function]
`set_syntax_filename(const char * filename)`
Sets the filename for syntax output. This should only be called if there are no extra files.

FileFilter void [member on function]
`set_xref_filename(const char * filename)`
Sets the filename for xref output. This should only be called if there are no extra files.

FileFilter void [member on function]
`set_syntax_prefix(const char * filename)`
Sets the prefix for syntax output filenames. The syntax filename will be the source filename prepended with the prefix, so you probably want the prefix to be a directory. This can be called whether there are extra files or not.

FileFilter void `set_xref_prefix(` [member on function]
`const char * filename)`
Sets the prefix for xref output filenames. The xref filename will be the source filename prepended with the prefix, so you probably want the prefix to be a directory. This can be called whether there are extra files or not.

FileFilter AST::SourceFile * [member on function]
`get_sourcefile(const char * filename, size_t`
`length)`
Returns the AST::SourceFile for the given filename. If length is given, then the filename is assumed to be that length. This is useful since OCC returns filenames as a reference to the #line directive in the preprocessor output and is not null-terminated.

FileFilter bool [member on function]
`should_visit_function_impl(AST::SourceFile *`
`file)`
Returns whether a function implementation in the given file should have its Ptree walked. This will be true only if the given file is one of the files to be stored.

FileFilter bool `should_link(` [member on function]
 `AST::SourceFile * file)`

Returns true if links should be generated for the given sourcefile

FileFilter bool `should_xref(` [member on function]
 `AST::SourceFile * file)`

Returns true if xref info should be generated for the given sourcefile

FileFilter bool `should_store(` [member on function]
 `AST::Declaration * decl)`

Returns true if the given declaration should be stored in the final AST. Note that a Declaration is taken instead of a SourceFile because Namespaces may be declared (and contain declarations) in multiple files, and Classes may have nested classes defined in other files.

FileFilter std::string [member on function]
 `strip_basename(const std::string & filename)`

Strips a filename of the basename if present

FileFilter std::string [member on function]
 `get_syntax_filename(AST::SourceFile * file)`

Returns the filename to use for storing syntax info

FileFilter std::string [member on function]
 `get_xref_filename(AST::SourceFile * file)`

Returns the filename to use for storing xref info

FileFilter void [member on function]
 `get_all_sourcefiles(AST::SourceFile::vector &`
 `)`

Returns a list of all the sourcefiles

FileFilter void [member on function]
 `get_all_filenames(const std::string * & main,`
 `const std::vector<std::string> * & extra)`

Returns all the filenames given to the parser.

FileFilter FileFilter * [member on function]
 `instance()`

Returns a pointer to the Filter instance. Note that Filter is *not* a regular singleton: `instance()` will return NULL if the Filter doesn't exist, and the constructor/destructor control

this. The reason for this method is so the C function `synopsis_include_hook` can use the Filter object without having a reference to it.

Private * m [data member]
Compiler firewalled private data

FileFilter bool is_main([member on function]
`std::string filename`)
Returns true if the given SourceFile is one of the main files
(including extra files)

Private [struct]

bool only_main [data member]
Whether only main declarations should be stored

std::string main_filename [data member]
The main filename

std::string basename [data member]
The basename

std::vector<std::string> string_vector [typedef]
A vector of strings

string_vector extra_filenames [data member]
The extra filenames

std::string syntax_filename [data member]
The main syntax filename

std::string xref_filename [data member]
The main xref filename

std::string syntax_prefix [data member]
The syntax filename prefix

std::string xref_prefix [data member]
The xref filename prefix

std::map<std::string, AST::SourceFile [typedef]
***> file_map_t**
Type of the map from filename to SourceFile

file_map_t file_map [data member]
A map from filename to SourceFile

StoreType [enum]
The type of syntax or xref being used

None [enumerator]

Filename [enumerator]

Prefix [enumerator]

StoreType syntax [data member]
The type of syntax filename being used

StoreType xref [data member]
The type of xref filename being used

std::vector< AST::Macro *> * [global variable]

syn_macro_defines

Declared in link_map.cc just because all the ucpp->synopsis hooks are there for now.

LinkMap [class]

LinkMap is a map from preprocessed file positions to input file positions

LinkMap LinkMap() [member on function]
Constructor

LinkMap LinkMap * instance() [member on function]
Returns a reference to a singleton instance of link_map

LinkMap void add(const char * name, int line, int out_start, int out_end, int diff) [member on function]

Adds a map at the given line. out_{start,end} define the start and past-the-end markers for the macro in the output file. diff defines the signed difference to add to the pos.

LinkMap int map(int line, int col) [member on function]

Applies added maps to the given column number. The differentials of the various macros are applied in turn, and the final column position is returned. Returns -1 if col is inside a macro

LinkMap **void** **clear()** [member on function]
 Clears the map. Should be called at the very start of processing, since the map is stored statically and hence potentially across parser invocations.

Private * m [data member]

Private [struct]

LinkMap::Private::LineMap lines [data member]

LinkStore [class]

Stores link information about the file. Link info is stored in two files with two purposes.

The first file stores all links and non-link spans, in a simple text file with one record per line and with spaces as field separators. The fields themselves are encoded using URL-style %FF encoding of non alpha-numeric characters (including spaces, brackets, commas etc). The purpose of this file is for syntax-highlighting of source files.

The second file stores only cross-reference information, which is a subset of the first file.

Context [enum]
 Enumeration of record types

Reference [enumerator]

Definition [enumerator]
 < General name reference

Span [enumerator]
 < Definition of the declaration

Implementation [enumerator]
 < Non-declarative span of text

UsingDirective [enumerator]
 < Implementation of a declaration

UsingDeclaration [enumerator]
 < Referenced in a using directive

FunctionCall [enumerator]
 < Referenced in a using declaration

NumContext [enumerator]
 < Called as a function

LinkStore **LinkStore**(**FileFilter** [member on function]
 ** filter*, **SWalker** ** swalker*)
 Constructor.

LinkStore **~LinkStore**() [member on function]
 Destructor. Closes all opened file streams

LinkStore **void** **link**(**Ptree** *** [member on function]
 node, **Context** *,* **ScopedName** *& name*, **const**
 std::string *& desc*, **const** **AST::Declaration** ***
 decl)
 Store a link for the given Ptree node. If a decl is given, store
 an xref too

LinkStore **void** **link**(**Ptree** *** [member on function]
 node, **const** **AST::Declaration** ** decl*)
 Store a Definition link for the given Ptree node using the
 AST node

LinkStore **void** **link**(**Ptree** *** [member on function]
 node, **Types::Type** *** *,* **Context** *)*
 Store a link for the given node using the given Context, which
 defaults to a Reference

LinkStore **void** **span**(**int** *line*, [member on function]
 int *col*, **int** *len*, **const** **char** ** desc*)
 Store a span

LinkStore **void** **span**(**Ptree** *** [member on function]
 node, **const** **char** ** desc*)
 Store a span for the given Ptree node

LinkStore **void** **long_span**(**Ptree** [member on function]
 ** node*, **const** **char** ** desc*)
 Store a long (possibly multi-line) span

LinkStore **SWalker** *** **swalker**() [member on function]
 Returns the SWalker

LinkStore **void** [member on function]
 store_syntax_record(**AST::SourceFile** *** *,* **int**
 line, **int** *col*, **int** *len*, **Context** *context*, **const**
 ScopedName *& name*, **const** **std::string** *& desc*)
 Store a link in the Syntax File

LinkStore void [member on function]
store_xref_record(AST::SourceFile * , const
 AST::Declaration * *decl*, const std::string &
file, int *line*, Context *context*)
 Store a link in the CrossRef File

LinkStore std::ostream & [member on function]
get_syntax_stream(AST::SourceFile *)
 Gets the ostream for a syntax file

LinkStore std::ostream & [member on function]
get_xref_stream(AST::SourceFile *)
 Gets the ostream for a xref file

LinkStore int find_col(int *line*, [member on function]
 const char * *ptr*)
 Calculates the column number of 'ptr'. m.buffer_start is used
 as a lower bounds, since the function counts backwards until
 it finds a newline. As an added bonus, the returned column
 number is adjusted using the link map generated from ex-
 panding macros so it can be output straight to the link file :)
 The adjustment requires the line number.

Private * m [data member]
 Compiler firewalled private data

Private [struct]

const char * buffer_start [data member]
 The start of the program buffer

FileFilter * filter [data member]
 The filter

Parser * parser [data member]
 The Parser object

SWalker * walker [data member]
 The SWalker object

Streams [struct]
 Struct for storing a pair of ostream pointers

std::ostream * syntax [data member]

```

std::ofstream * xref           [data member]

Streams Streams()             [member on function]

Streams Streams( const       [member on function]
Streams & o )

Streams Streams &           [member on function]
operator=( const Streams & o)

std::map<    AST::SourceFile  *, [typedef]
Streams> StreamsMap
The type of a map of streams

StreamsMap streams           [data member]
A map of streams for each file

encode                       [class]

const char * str             [data member]

encode encode( const char *   [member on function]
s )

encode encode( const         [member on function]
std::string & s )

encode_name                   [class]

const ScopedName & name     [data member]

encode_name encode_name(     [member on function]
const ScopedName & N)

Lookup                       [class]
AST Builder. This class manages the building of an AST, including
queries on the existing AST such as name and type lookups. The
building operations are called by SWalker as it walks the parse tree.

Lookup Lookup( Builder * )   [member on function]
Constructor

Lookup ~Lookup()             [member on function]
Destructor. Recursively destroys all AST objects

Lookup void set_access(      [member on function]
AST::Access )
Changes the current accessibility for the current scope

```

Lookup **AST::Scope** * *scope*() [member on function]
Returns the current scope

Lookup **AST::Scope** * *global*() [member on function]
Returns the global scope

Lookup **Types::Named** * [member on function]
lookupType(const std::string & *name*, bool
func_okay)
Looks up the name in the current scope. This method always succeeds – if the name is not found it forward declares it.

Lookup **Types::Named** * [member on function]
lookupType(const ScopedName & *names*, bool
func_okay, AST::Scope * *scope*)
Looks up the qualified name in the current scope. This method always succeeds – if the name is not found it forwards declares it.

Lookup **Types::Named** * [member on function]
lookupType(const std::string & *name*, AST::Scope
* *scope*)
Looks up the name in the scope of the given scope. This method may return a NULL ptr if the lookup failed.

Lookup **AST::Function** * [member on function]
lookupFunc(const std::string & , AST::Scope * ,
const std::vector< Types::Type * > &)
Looks up the function in the given scope with the given args.

Lookup **AST::Function** * [member on function]
lookupOperator(const std::string & *oper*,
Types::Type * *left_type*, Types::Type *
right_type)
Looks up the function operator in the current scope with the given types. May return NULL if builtin operator or no operator is found.

Lookup **bool** *mapName*(const [member on function]
ScopedName & *name*, std::vector< AST::Scope * > &
, Types::Named * &)
Maps a scoped name into a vector of scopes and the final type. Returns true on success.

Lookup **Types::Type *** [member on function]
 arrayOperator(Types::Type * *object*,
 Types::Type * *arg*, AST::Function * &)

Returns the types for an array operator on the given type with an argument of the given type. If a function is used then it is stored in the function ptr ref given, else the ptr is set to NULL.

Lookup **Types::Named *** [member on function]
 resolveType(Types::Named * *maybe_unknown*)

Resolves the final type of the given type. If the given type is an Unknown, it checks to see if the type has been defined yet and returns that instead.

Lookup **Types::Named *** lookup([member on function]
 const std::string & *name*, bool *func_okay*)

Looks up the name in the current scope. This method may fail and return a NULL ptr.

Lookup **Types::Named *** lookup([member on function]
 const std::string & *name*, const ScopeSearch & ,
 bool *func_okay*)

Searches for name in the list of Scopes. This method may return NULL if the name is not found.

Lookup **Types::Named *** [member on function]
 lookupQual(const std::string & *name*, const
 ScopeInfo * , bool *func_okay*)

Searches for name in the given qualified scope. This method may return NULL if the name is not found. Lookup proceeds according to the spec: if 'scope' is a Class scope, then scope and all base classes are searched, else if it's a 'namespace' scope then all usings are checked.

Lookup **ScopeInfo *** find_info([member on function]
 AST::Scope *)

Return a ScopeInfo* for the given Declaration. This method first looks for an existing Scope* in the Private map.

Lookup **void** findFunctions(const [member on function]
 std::string & , ScopeInfo * , std::vector<
 AST::Function *> &)

Utility class to add all functions with the given name in the given Scope's dictionary to the given vector. May throw an error if the types looked up are not functions.

Lookup **AST::Function *** [member on function]
bestFunction(const std::vector< AST::Function
 *> & , const std::vector< Types::Type *> & , int
 & *cost*)

Determines the best function from the given list for the given arguments using heuristics. Returns the function and stores the cost

Lookup **std::string** **dumpSearch**([member on function]
 ScopeInfo * *scope*)

Formats the search of the given Scope for logging

Builder * **m_builder** [data member]

A pointer to the Builder.

ScopeInfo [struct]

parents: parent class This class encapsulates information about a Scope and its dictionary of names. Since the AST::Scope is meant purely for documentation purposes, this class provides the extra information needed for operations such as name lookup (a dictionary of types, and links to using scopes).

The using directives work as follows: Using directives effectively add all contained declarations to the namespace containing the directive, but they still need to be processed separately. To handle this each ScopeInfo remembers which other namespaces it is using {using_scopes}.

One quirk is that all used namespaces are considered as a whole for overload resolution, *not* one at a time. To facilitate this 'dummy' scopeinfos are inserted into the ScopeSearch of a containing namespace, which the algorithms recognize. (TODO: consider a wrapper(?) a.la AST::Inheritance)

ScopeInfo ScopeInfo(AST::Scope [member on function]
 * *s*)

Constructor

ScopeInfo ScopeInfo(ScopeInfo [member on function]
 * *s*)

Constructor that creates a Dummy 'using' scope referencing 's'

ScopeInfo ~ScopeInfo() [member on function]

Destructor

Dictionary * **dict** [data member]

Dictionary for this scope

| | |
|--|----------------------|
| AST::Scope * scope_decl | [data member] |
| The declaration for this scope | |
| ScopeSearch search | [data member] |
| The list of scopes to search for this scope, including this | |
| ScopeSearch using_scopes | [data member] |
| The list of scopes in the search because they are 'using'd | |
| ScopeSearch used_by | [data member] |
| The list of scopes 'using' this one | |
| bool is_using | [data member] |
| True only if this is a dummy Scope representing the use of another. If this is the case, then only 'dict' is set | |
| AST::Access access | [data member] |
| Current accessibility | |
| std::map<std::string, int> nscounts | [data member] |
| Counts of named sub-namespaces. The names are things like "if", "while" etc. This is for the purely aesthetic purpose of being able to name anonymous namespaces like "'if 'while 'if2 'do" instead of "'if1 'while2 'if3 'do4" or even "'0001 '0002 '0003 '0004" as OpenC++ does. | |
| ScopeInfo int getCount(const std::string & name) | [member on function] |
| | |
| STrace | [class] |
| Dummy STrace guard for release code - should be optimized away | |
| STrace STrace(const std::string &) | [member on function] |
| | |
| STrace ~STrace() | [member on function] |
| | |
| TranslateError | [class] |
| parents: parent class Exception thrown by errors when translating the Ptree into an AST | |
| TranslateError char * str() | [member on function] |
| | |
| TranslateError const char * what() | [member on function] |
| | |

TranslateError **void** set_node([member on function]
Ptree *)

SWalker [class]

parents: parent class A walker that creates an AST. All Translate* methods have been overridden to remove the translation code.

SWalker SWalker(FileFilter * , [member on function]
Parser * , Builder * , Program *)
Constructor

SWalker ~SWalker() [member on function]

SWalker **void** set_extract_tails([member on function]
bool value)
Sets extract tails to true. This will cause the parser to create dummy declarations for comments before close braces or the end of the file

SWalker **void** set_store_links([member on function]
LinkStore *)
Sets store links to true. This will cause the whole ptree to be traversed, and any linkable identifiers found will be stored

SWalker **std::string** parse_name([member on function]
Ptree *)
Get a name from the ptree

SWalker **Parser** * parser() [member on function]
Get the Parser object

SWalker **Program** * program() [member on function]
Get the Program object

SWalker **Builder** * builder() [member on function]
Get the Builder object

SWalker **TypeFormatter** * [member on function]
type_formatter()
Get the TypeFormatter object

SWalker **int** line_of_ptree(Ptree [member on function]
*)
Get the line number of the given Ptree node

SWalker **void** update_line_number([member on function]
Ptree *)

Update the line number

SWalker **void** add_comments([member on function]
AST::Declaration * decl, Ptree * comments)

SWalker **void** add_comments([member on function]
AST::Declaration * decl, CommentedLeaf * node)

SWalker **void** add_comments([member on function]
AST::Declaration * decl, PtreeDeclarator *
node)

SWalker **void** add_comments([member on function]
AST::Declaration * decl, PtreeDeclaration *
decl)

SWalker **void** add_comments([member on function]
AST::Declaration * decl, PtreeNamespaceSpec *
decl)

SWalker **void** find_comments([member on function]
Ptree * node)

Traverses left side of tree till it finds a leaf, and if that is a
CommentedLeaf then it adds those comments as spans

SWalker **void** [member on function]
TranslateFunctionArgs(Ptree * args)

SWalker **void** [member on function]
TranslateParameters(Ptree * p_params,
std::vector< AST::Parameter *> & params)

SWalker **void** [member on function]
TranslateFunctionName(char * encname,
std::string & realname, Types::Type * &
returnType)

SWalker **Ptree *** [member on function]
TranslateDeclarator(Ptree *)

SWalker **Ptree *** [member on function]
TranslateFunctionDeclarator(Ptree * , bool
is_const)

SWalker **Ptree *** [member on function]
 TranslateVariableDeclarator(Ptree * , bool
is_const)

SWalker **void** [member on function]
 TranslateTypedefDeclarator(Ptree * *node*)

SWalker **std::vector<**
AST::Inheritance *> [member on function]
 TranslateInheritanceSpec(Ptree * *node*)

SWalker **std::string** [member on function]
 format_parameters(std::vector< AST::Parameter
 *> & *params*)

Returns a formatter string of the parameters. The idea is that this string will be appended to the function name to form the 'name' of the function.

SWalker **Ptree *** TranslatePtree([member on function]
 Ptree *)

SWalker **void** Translate(Ptree *) [member on function]

Overridden to catch exceptions

SWalker **Ptree *** [member on function]
 TranslateTypedef(Ptree *)

SWalker **Ptree *** [member on function]
 TranslateTemplateDecl(Ptree *)

SWalker **Ptree *** [member on function]
 TranslateTemplateInstantiation(Ptree *)

SWalker **Ptree *** [member on function]
 TranslateExternTemplate(Ptree *)

SWalker **Ptree *** [member on function]
 TranslateTemplateClass(Ptree * , Ptree *)

SWalker **Ptree *** [member on function]
 TranslateTemplateFunction(Ptree * , Ptree *)

SWalker **Ptree *** [member on function]
 TranslateMetaclassDecl(Ptree *)

SWalker **Ptree *** [member on function]
 TranslateLinkageSpec(Ptree *)

```

SWalker Ptree * [member on function]
    TranslateNamespaceSpec( Ptree * )

SWalker Ptree * TranslateUsing( [member on function]
    Ptree * )

SWalker Ptree * [member on function]
    TranslateDeclaration( Ptree * )

SWalker Ptree * [member on function]
    TranslateStorageSpecifiers( Ptree * )

SWalker Ptree * [member on function]
    TranslateDeclarators( Ptree * )

SWalker Ptree * [member on function]
    TranslateArgDeclList( bool , Ptree * , Ptree * )

SWalker Ptree * [member on function]
    TranslateInitializeArgs( PtreeDeclarator * ,
    Ptree * )

SWalker Ptree * [member on function]
    TranslateAssignInitializer( PtreeDeclarator * ,
    Ptree * )

SWalker Ptree * [member on function]
    TranslateFunctionImplementation( Ptree * )

SWalker Ptree * [member on function]
    TranslateFunctionBody( Ptree * )

SWalker Ptree * TranslateBrace( [member on function]
    Ptree * )

SWalker Ptree * TranslateBlock( [member on function]
    Ptree * )

SWalker Ptree * [member on function]
    TranslateClassSpec( Ptree * )

SWalker Ptree * [member on function]
    TranslateEnumSpec( Ptree * )

SWalker Ptree * [member on function]
    TranslateAccessSpec( Ptree * )

```

SWalker **Ptree *** [member on function]
TranslateAccessDecl(Ptree *)

SWalker **Ptree *** [member on function]
TranslateUserAccessSpec(Ptree *)

SWalker **Ptree *** TranslateIf([member on function]
Ptree *)

SWalker **Ptree *** [member on function]
TranslateSwitch(Ptree *)

SWalker **Ptree *** TranslateWhile([member on function]
Ptree *)

SWalker **Ptree *** TranslateDo([member on function]
Ptree *)

SWalker **Ptree *** TranslateFor([member on function]
Ptree *)

SWalker **Ptree *** TranslateTry([member on function]
Ptree *)

SWalker **Ptree *** TranslateBreak([member on function]
Ptree *)

SWalker **Ptree *** [member on function]
TranslateContinue(Ptree *)

SWalker **Ptree *** [member on function]
TranslateReturn(Ptree *)

SWalker **Ptree *** TranslateGoto([member on function]
Ptree *)

SWalker **Ptree *** TranslateCase([member on function]
Ptree *)

SWalker **Ptree *** [member on function]
TranslateDefault(Ptree *)

SWalker **Ptree *** TranslateLabel([member on function]
Ptree *)

SWalker **Ptree *** [member on function]
TranslateExprStatement(Ptree *)

| | | |
|---------|---|----------------------|
| SWalker | Ptree * | [member on function] |
| | TranslateTypespecifier(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateTypeof(Ptree * , Ptree * declarations) | |
| SWalker | Ptree * | [member on function] |
| | TranslateComma(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateAssign(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateCond(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateInfix(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslatePm(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateCast(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateUnary(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateThrow(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateSizeof(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateNew(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateNew3(Ptree * type) | |
| SWalker | Ptree * | [member on function] |
| | TranslateDelete(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateThis(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateVariable(Ptree *) | |

| | | |
|------------|---|----------------------|
| SWalker | Ptree * | [member on function] |
| | TranslateFstyleCast(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateArray(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateFuncall(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslatePostfix(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateUserStatement(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateDotMember(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateArrowMember(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateParen(Ptree *) | |
| SWalker | Ptree * | [member on function] |
| | TranslateStaticUserStatement(Ptree *) | |
| SWalker | AST::SourceFile * | [member on function] |
| | current_file() | |
| SWalker | int | [member on function] |
| | current_lineno() | |
| SWalker | SWalker * | [member on function] |
| | instance() | |
| SWalker | * g_swalker | [data member] |
| Parser | * m_parser | [data member] |
| Builder | * m_builder | [data member] |
| FileFilter | * m_filter | [data member] |
| Program | * m_program | [data member] |
| Decoder | * m_decoder | [data member] |
| Lookup | * m_lookup | [data member] |

| | |
|--|---------------|
| Ptree * m_declaration | [data member] |
| A pointer to the current declaration ptree, if any, used to get the return type and modifiers, etc. | |
| std::vector< AST::Parameter *> * m_template | [data member] |
| A pointer to the current template parameters, if any, used to get the template parameters and set in the declaration. Should be NULL if not in a template. | |
| char * m_filename_ptr | [data member] |
| This pointer is used as a comparison to avoid redundant effort. Do not use it to get the filename, since the string is a reference to the preprocessed file in memory and is <i>*not*</i> null terminated! | |
| AST::SourceFile * m_file | [data member] |
| The current file, set by <code>update_line_number</code> | |
| int m_lineno | [data member] |
| bool m_extract_tails | [data member] |
| True if should try and extract tail comments before <code>}</code> 's | |
| LinkStore * m_links | [data member] |
| Storage for links. This is only set if we should be storing links, so it must be checked before every use | |
| bool m_store_decl | [data member] |
| True if this <code>TranslateDeclarator</code> should try to store the decl type | |
| std::vector<std::string> m_dummysname | [data member] |
| A dummy name used for tail comments | |
| TypeFormatter * m_type_formatter | [data member] |
| An instance of <code>TypeFormatter</code> for formatting types | |
| AST::Function * m_function | [data member] |
| The current function, if in a function block | |
| std::vector< AST::Parameter *> m_param_cache | [data member] |
| The params found before a function block. These may be different from the ones that are in the original declaration(s), but it is these names we need for referencing names inside the block, so a reference is stored here. | |

std::vector< Types::Type *> m_params [data member]

The types accumulated for function parameters in function calls

Types::Type * m_type [data member]

The type returned from the expression-type translators

AST::Scope * m_scope [data member]

The Scope to use for name lookups, or NULL to use enclosing default scope rules. This is for when we are at a Variable, and already know it must be part of a given class (eg, foo->bar .. bar must be in foo's class)

Postfix_Flag [enum]

The state of postfix translation. This is needed for constructs like foo->var versus var or foo->var(). The function call resolution needs to be done in the final TranslateVariable, since that's where the last name (which is to be linked) is handled.

Postfix_Var [enumerator]

Postfix_Func [enumerator]

< Lookup as a variable

Postfix_Flag m_postfix_flag [data member]

foo->var versus var or foo->var(). The function call resolution needs to be done in the final TranslateVariable, since that's where the last name (which is to be linked) is handled.

FuncImplCache [struct]

Info about one stored function impl. Function impls must be stored till the end of a class.

AST::Function * func [data member]

**std::vector< AST::Parameter *>
params** [data member]

Ptree * body [data member]

std::vector<FuncImplCache> FuncImplVec [typedef]

A vector of function impls

std::vector<FuncImplVec> FuncImplStack [typedef]

A stack of function impl vectors

FuncImplStack **m_func_impl_stack** [data member]
The stack of function impl vectors

SWalker **int** **find_col**(**const char *** [member on function]
start, **const char * find**)
Finds the column given the start ptr and the current position.
The derived column number is processed with the link_map
before returning, so -1 may be returned to indicate "inside
macro".

Trace [class]

Trace **Trace**(**const std::string &** [member on function]
)

Trace **~Trace**() [member on function]

Synopsis [class]

parents: parent class parent class The Synopsis class maps from
C++ objects to Python objects

Synopsis **Synopsis**(**FileFilter *** [member on function]
, **PyObject * decls**, **PyObject * types**)

Synopsis **~Synopsis**() [member on function]

Synopsis **void** **translate**([member on function]
AST::Scope * global, **PyObject * ast**)

Synopsis **void** **set_builtin_decls**([member on function]
const AST::Declaration::vector & builtin_decls)

Synopsis **PyObject *** **Base**([member on function]
Types::Base *)

Synopsis **PyObject *** **Unknown**([member on function]
Types::Named *)

Synopsis **PyObject *** **Declared**([member on function]
Types::Declared *)

Synopsis **PyObject *** **Dependent**([member on function]
Types::Dependent *)

Synopsis **PyObject *** **Template**([member on function]
Types::Template *)

Synopsis **PyObject *** Modifier([member on function]
Types::Modifier *)

Synopsis **PyObject *** Array([member on function]
Types::Array *)

Synopsis **PyObject *** [member on function]
Parameterized(Types::Parameterized *)

Synopsis **PyObject *** FuncPtr([member on function]
Types::FuncPtr *)

Synopsis **PyObject *** [member on function]
SourceFile(AST::SourceFile *)

Synopsis **PyObject *** Include([member on function]
AST::Include *)

Synopsis **PyObject *** [member on function]
Declaration(AST::Declaration *)

Synopsis **PyObject *** Macro([member on function]
AST::Macro *)

Synopsis **PyObject *** Forward([member on function]
AST::Forward *)

Synopsis **PyObject *** Scope([member on function]
AST::Scope *)

Synopsis **PyObject *** Namespace([member on function]
AST::Namespace *)

Synopsis **PyObject *** [member on function]
Inheritance(AST::Inheritance *)

Synopsis **PyObject *** Class([member on function]
AST::Class *)

Synopsis **PyObject *** Typedef([member on function]
AST::Typedef *)

Synopsis **PyObject *** [member on function]
Enumerator(AST::Enumerator *)

Synopsis **PyObject *** Enum([member on function]
AST::Enum *)

| | |
|---|----------------------|
| Synopsis PyObject * Variable(AST::Variable *) | [member on function] |
| Synopsis PyObject * Const(AST::Const *) | [member on function] |
| Synopsis PyObject * Parameter(AST::Parameter *) | [member on function] |
| Synopsis PyObject * Function(AST::Function *) | [member on function] |
| Synopsis PyObject * Operation(AST::Operation *) | [member on function] |
| Synopsis PyObject * Comment(AST::Comment *) | [member on function] |
| Synopsis void visit_declaration(AST::Declaration *) | [member on function] |
| Synopsis void visit_macro(AST::Macro *) | [member on function] |
| Synopsis void visit_scope(AST::Scope *) | [member on function] |
| Synopsis void visit_namespace(AST::Namespace *) | [member on function] |
| Synopsis void visit_class(AST::Class *) | [member on function] |
| Synopsis void visit_inheritance(AST::Inheritance *) | [member on function] |
| Synopsis void visit_forward(AST::Forward *) | [member on function] |
| Synopsis void visit_typedef(AST::Typedef *) | [member on function] |
| Synopsis void visit_variable(AST::Variable *) | [member on function] |
| Synopsis void visit_const(AST::Const *) | [member on function] |

| | |
|---|----------------------|
| Synopsis void visit_enum(AST::Enum *) | [member on function] |
| Synopsis void visit_enumerator(AST::Enumerator *) | [member on function] |
| Synopsis void visit_function(AST::Function *) | [member on function] |
| Synopsis void visit_operation(AST::Operation *) | [member on function] |
| Synopsis void visit_parameter(AST::Parameter *) | [member on function] |
| Synopsis void visit_comment(AST::Comment *) | [member on function] |
| Synopsis void visit_unknown(Types::Unknown *) | [member on function] |
| Synopsis void visit_modifier(Types::Modifier *) | [member on function] |
| Synopsis void visit_array(Types::Array *) | [member on function] |
| Synopsis void visit_base(Types::Base *) | [member on function] |
| Synopsis void visit_dependent(Types::Dependent *) | [member on function] |
| Synopsis void visit_declared(Types::Declared *) | [member on function] |
| Synopsis void visit_template_type(Types::Template *) | [member on function] |
| Synopsis void visit_parameterized(Types::Parameterized *) | [member on function] |
| Synopsis void visit_func_ptr(Types::FuncPtr *) | [member on function] |
| Private * m | [data member] |

Synopsis **void** addComments([member on function]
 PyObject * pydecl, AST::Declaration * cdecl)
 helper methods

PyObject * m_ast [data member]

PyObject * m_type [data member]

PyObject * m_declarations [data member]

PyObject * m_dictionary [data member]

FileFilter * m_filter [data member]

Private [struct]

Private Private(Synopsis * [member on function]
 s)
 Constructor

Synopsis * m_syn [data member]
 Reference to parent synopsis object

PyObject * m_cxx [data member]
 Interned string for "C++"

Private PyObject * cxx() [member on function]
 Returns the string for "C++" as a borrowed reference

std::map< void *, PyObject *> ObjMap [typedef]

ObjMap obj_map [data member]

std::set< AST::Declaration *>
builtin_decl_set [data member]

Private PyObject * py([member on function]
 AST::SourceFile * file)
 Return the PyObject for the given AST::SourceFile

Private PyObject * py([member on function]
 AST::Include * incl)
 Return the PyObject for the given AST::Include

Private PyObject * py([member on function]
 AST::Declaration * decl)
 Return the PyObject for the given AST::Declaration

Private **PyObject** * py([member on function]
AST::Inheritance * decl)

Return the PyObject for the given AST::Inheritance

Private **PyObject** * py([member on function]
AST::Parameter * decl)

Return the PyObject for the given AST::Parameter

Private **PyObject** * py([member on function]
AST::Comment * decl)

Return the PyObject for the given AST::Comment

Private **PyObject** * py([member on function]
Types::Type * type)

Return the PyObject for the given Types::Type

Private **PyObject** * py([member on function]
const std::string & str)

Return the PyObject for the given string

Private **void** add(void * [member on function]
cobj, PyObject * pyobj)

Add the given pair

Private **PyObject** * List([member on function]
const std::vector< std::vector * > & vec)

Private **PyObject** * Tuple([member on function]
const std::vector< std::vector * > & vec)

Private **PyObject** * List([member on function]
const std::vector<std::string> & vec)

Private **PyObject** * Tuple([member on function]
const std::vector<std::string> & vec)

Class [class]

A test class

Enum [enum]

An enum

ALPHA [enumerator]

Comment before ALPHA

BETA [enumerator]
 < PrevComment after ALPHA

Types [namespace]
 The Type hierarchy

Type [class]
 parents: parent class The base class of the Type hierarchy

std::vector< Type *> vector [typedef]
 A vector of Type objects

std::vector<std::string> Mods [typedef]
 Typedef for modifier list

Type Type() [member on function]
 Constructor

Type ~Type() [member on function]
 Destructor. Note that Types::Type becomes abstract,
 unlike AST::Declaration

**Type void accept(Visitor *
)** [member on function]
 Accept the given visitor

Named [class]
 parents: parent class Base class of types with Names

std::vector< Named *> vector [typedef]
 A vector of Types::Named objects

**Named Named(const
 ScopedName & name)** [member on function]
 Constructor

Named ~Named() [member on function]
 Destructor

**Named void accept(Visitor
 *)** [member on function]
 Accept the given visitor

**Named const ScopedName
 & name()** [member on function]
 Constant version of name()

Named ScopedName & [member on function]
name()

Return the scoped ScopedName of this type

ScopedName m_name [data member]
 The scoped ScopedName of this type

Base [class]
 parents: parent class Base types are the (implicitly declared)
 builtin types such as 'int', 'bool' etc

Base Base(const ScopedName [member on function]
& name)
 Constructor

Base void accept(Visitor * [member on function]
)
 Accept the given visitor

Unknown [class]
 parents: parent class Unknown type

Unknown Unknown(const [member on function]
ScopedName & name)
 Constructor

Unknown void accept([member on function]
Visitor *)
 Accept the given visitor

Dependent [class]
 parents: parent class Template parameter dependent types
 have a possibly scoped name, but no known type. Member
 types of template-parameters will be scoped, eg:

```
template<class T> class A { T::B b; }
```

Here T::B is a dependant type, and so is T.

Dependent Dependent(const [member on function]
ScopedName & name)
 Constructor

Dependent void accept([member on function]
Visitor *)
 Accept the given visitor

Declared [class]

parents: parent class Declared types have a name and a reference to their declaration

```
Declared Declared( const      [member on function]
                  ScopedName & name, AST::Declaration *
                  decl)
Constructor
```

```
Declared void accept(      [member on function]
                    Visitor * )
Accept the given visitor
```

```
Declared const          [member on function]
AST::Declaration * declaration()
Const version of declaration()
```

```
Declared                [member on function]
AST::Declaration * declaration()
Returns the Declaration referenced by this type
```

```
AST::Declaration * m_decl      [data member]
The declaration referenced by this type
```

Template [class]

parents: parent class Template types are declared template types. They have a name, a declaration (which is an AST::Class) and a vector of parameters declare this template. Each parameter (using AST::Parameter) should be either the correct type for non-type parameters, or a Dependent for type parameters. In either case, there may be default values.

```
std::vector< Template *> vector      [typedef]
A vector of Types::Template objects
```

```
std::vector< AST::Parameter *>      [typedef]
param_vector
A vector of Parameter objects
```

```
Template Template( const      [member on function]
                  ScopedName & name, AST::Declaration *
                  decl, const param_vector & params)
Constructor
```

Template **void** `accept(Visitor *)` [member on function]
 Accept the given visitor

Template **const** `param_vector & parameters()` [member on function]
 Constant version of `parameters()`

Template **param_vector &** `parameters()` [member on function]
 Returns the vector of parameter Types

Template **const vector &** `specializations()` [member on function]
 Constant version of `specializations()`

Template **vector &** `specializations()` [member on function]
 Returns the vector of specializations for this template.
 If this template is a specialization, the vector must be empty

param_vector m_params [data member]
 The parameters

vector m_specs [data member]
 The vector of specializations for this template

Modifier [class]
 parents: parent class Type Modifier. This type has a nested type, and wraps it with modifiers such as `const`, `reference`, etc.

Modifier `Modifier(Type * alias, const Type::Mods & pre, const Type::Mods & post)` [member on function]
 Constructor

Modifier `~Modifier()` [member on function]
 Destructor

Modifier **void** `accept(Visitor *)` [member on function]
 Accept the given visitor

Modifier **Type * alias()** [member on function]
Returns the alias (modified) Type object

Modifier **Type::Mods & pre()** [member on function]
Returns the premodifiers

Modifier **Type::Mods & post()** [member on function]
Returns the postmodifiers

Type * **m_alias** [data member]
The alias type

Type::Mods **m_pre** [data member]
The pre and post modifiers

Type::Mods **m_post** [data member]

Array [class]

parents: parent class Type Array. This type adds array dimensions to a primary type

Array **Array(Type * alias, const Type::Mods & sizes)** [member on function]
Constructor

Array **~Array()** [member on function]
Destructor

Array **void accept(Visitor *)** [member on function]
Accept the given visitor

Array **Type * alias()** [member on function]
Returns the alias (modified) Type object

Array **Type::Mods & sizes()** [member on function]
Returns the sizes

Type * **m_alias** [data member]
The alias type

Type::Mods **m_sizes** [data member]
The sizes

Parameterized [class]

parents: parent class Parameterized type. A parameterized type is an instantiation of a Template type with concrete Types as parameters.

Parameterized [member on function]
Parameterized(*Template * templ*, const
Type::vector & params)
 Constructor

Parameterized [member on function]
 ~**Parameterized**()
 Destructor

Parameterized void **accept**([member on function]
*Visitor **)
 Accept the given visitor

Parameterized Template * [member on function]
template_type()
 Returns the Template type this is an instance of

Parameterized const [member on function]
Type::vector & parameters()
 Constant version of parameters()

Parameterized [member on function]
Type::vector & parameters()
 Returns the vector of parameter Types

Template * m_template [data member]
 The Template object

Type::vector m_params [data member]
 The vector of parameter Types

FuncPtr [class]

parents: parent class Function Pointer type. Function ptr types have a return type and a list of parameter types, along with a list of premodifiers.

FuncPtr **FuncPtr**(*Type ** [member on function]
ret, const *Type::Mods & premods*, const
Type::vector & params)
 Constructor

| | |
|--|----------------------|
| <code>FuncPtr ~FuncPtr()</code> Destructor | [member on function] |
| <code>FuncPtr void accept(Visitor *)</code> Accept the given visitor | [member on function] |
| <code>FuncPtr Type * return_type()</code> Returns the Return Type object | [member on function] |
| <code>FuncPtr const Type::Mods & pre()</code> Const version of pre() | [member on function] |
| <code>FuncPtr Type::Mods & pre()</code> Returns the premodifier vector | [member on function] |
| <code>FuncPtr const Type::vector & parameters()</code> Constant version of parameters() | [member on function] |
| <code>FuncPtr Type::vector & parameters()</code> Returns the vector of parameter Types | [member on function] |
| <code>Type * m_return</code> The Return Type | [data member] |
| <code>Type::Mods m_premod</code> The premodifiers | [data member] |
| <code>Type::vector m_params</code> The parameters | [data member] |
| Visitor The Visitor base class | [class] |
| <code>Visitor ~Visitor()</code> | [member on function] |
| <code>Visitor void visit_type(Type *)</code> | [member on function] |
| <code>Visitor void visit_unknown(Unknown *)</code> | [member on function] |

Visitor **void** [member on function]
 visit_modifier(Modifier *)

Visitor **void** visit_array([member on function]
 Array *)

Visitor **void** visit_named([member on function]
 Named *)

Visitor **void** visit_base([member on function]
 Base *)

Visitor **void** [member on function]
 visit_dependent(Dependent *)

Visitor **void** [member on function]
 visit_declared(Declared *)

Visitor **void** [member on function]
 visit_template_type(Template *)

Visitor **void** [member on function]
 visit_parameterized(Parameterized *)

Visitor **void** [member on function]
 visit_func_ptr(FuncPtr *)

wrong_type_cast [class]

parents: parent class The exception thrown by the special
 cast templates

wrong_type_cast [member on function]
 wrong_type_cast()
 Constructor

wrong_type_cast [member on function]
 ~wrong_type_cast()
 Destructor

wrong_type_cast **const char** [member on function]
 * what()
 Returns name of this class

const char * * type_cast(Type * type_ptr) [function]
 Casts Types::Type types to derived types safely. The cast
 is done using dynamic_cast, and wrong_type_cast is thrown
 upon failure.

Type * * named_cast(Named * *named_ptr*) [function]
 Casts Types::Named types to derived types safely. The cast is done using `dynamic_cast`, and `wrong_type_cast` is thrown upon failure.

Named * * declared_cast(Named * *named_ptr*) [function]
 Safely extracts typed Declarations from Named types. The type is first safely cast to Types::Declared, then the `declaration()` safely cast to the template type.

Named * * declared_cast(Type * *type_ptr*) [function]
 Safely extracts typed Declarations from Type types. The type is first safely cast to Types::Declared, then the `declaration()` safely cast to the template type.

TypeInfo [class]
 parents: parent class

Types::Type * type [data member]

bool is_const [data member]

bool is_volatile [data member]

bool is_null [data member]

size_t deref [data member]

TypeInfo TypeInfo(Types::Type * *t*) [member on function]
 Constructor

TypeInfo void set(Types::Type * *t*) [member on function]
 Set to the given type

TypeInfo void visit_base(Types::Base * *base*) [member on function]
 Base – null is flagged since it is special

TypeInfo void visit_modifier(Types::Modifier * *mod*) [member on function]
 Modifiers – recurse on the alias type


```

TypeInfo void visit_declared(           [member on function]
    Types::Declared * t)
    Declared – check for typedef

std::ostream & operator<<( std::ostream & o,       [function]
    TypeInfo & i)
    Output operator for debugging

Program                                     [class]
    parents: parent class

Private                                     [struct]

    LineMapNode first                         [data member]

    LineMapNode last                         [data member]

    Private uint lastPos()                   [member on function]

    Private void start(                       [member on function]
        LineMapNode & node)

    Private void insert(                     [member on function]
        LineMapNode & node)

Replacement                                 [class]
    parents: parent class

    Replacement Replacement(                 [member on function]
        Replacement * , uint , uint , Ptree * )

    Replacement * next                       [data member]

    uint startpos                             [data member]

    uint endpos                               [data member]

    Ptree * text                               [data member]

LineMapNode                                 [struct]

    uint pos                                   [data member]

    int line                                   [data member]

    char * filename                           [data member]

```

| | |
|---|-------------------|
| <code>int filename_len</code> | [data member] |
| <code>const int Quantum</code> | [global variable] |
| <code>const int BufSize</code> | [global variable] |
| <code>int wait(int *)</code> | [function] |
| <code>const char * compilerName</code> | [global variable] |
| <code>const char * linkerName</code> | [global variable] |
| <code>const char * opencxxErrorMessage</code> | [global variable] |
| <code>bool showProgram</code> | [global variable] |
| <code>bool doCompile</code> | [global variable] |
| <code>bool makeExecutable</code> | [global variable] |
| <code>bool doPreprocess</code> | [global variable] |
| <code>bool doTranslate</code> | [global variable] |
| <code>bool verboseMode</code> | [global variable] |
| <code>bool regularCpp</code> | [global variable] |
| <code>bool makeSharedLibrary</code> | [global variable] |
| <code>char * sharedLibraryName</code> | [global variable] |
| <code>bool preprocessTwice</code> | [global variable] |
| <code>[] const char * cppArgv</code> | [global variable] |
| <code>[] const char * ccArgv</code> | [global variable] |
| <code>void ParseCmdOptions(int from, int argc, char ** argv, char * & source)</code> | [function] |
| <code>void AddCppOption(const char * arg)</code> | [function] |
| <code>void AddCcOption(const char * arg)</code> | [function] |
| <code>void CloseCcOptions()</code> | [function] |
| <code>void ShowCommandLine(const char * cmd, const char ** args)</code> | [function] |

| | |
|---|-------------------|
| <code>bool ParseTargetSpecificOptions(char * arg, char * & source_file)</code> | [function] |
| <code>void RunLinker()</code> | [function] |
| <code>char * RunPreprocessor(const char * src)</code> | [function] |
| <code>char * OpenCxxOutputFileName(const char * src)</code> | [function] |
| <code>void RunCompiler(const char * src, const char * ocsrc)</code> | [function] |
| <code>void RunSoCompiler(const char * src_file)</code> | [function] |
| <code>void * LoadSoLib(char * file_name)</code> | [function] |
| <code>void * LookupSymbol(void * handle, char * symbol)</code> | [function] |
| <code>char * MakeTempFilename(const char * src, const char * suffix)</code> | [function] |
| <code>[] char thisVersion</code> | [global variable] |
| <code>[] char copyingNote</code> | [global variable] |
| <code>const int NARGS</code> | [global variable] |
| <code>[] const char * cc2Argv</code> | [global variable] |
| <code>int cppArgc</code> | [global variable] |
| <code>int ccArgc</code> | [global variable] |
| <code>int cc2Argc</code> | [global variable] |
| <code>int numOfObjectFiles</code> | [global variable] |
| <code>void LoadMetaclass(char *)</code> | [function] |
| <code>void Compile(int argc, char * * argv)</code> | [function] |
| <code>bool IsCxxSource(char * fname)</code> | [function] |
| <code>void ReadStdin()</code> | [function] |
| <code>void ReadFile(const char * src)</code> | [function] |

| | |
|---|-------------------|
| <code>char * RunOpencxx(const char * src)</code> | [function] |
| <code>int ParseOpencxx(Program * parse)</code> | [function] |
| <code>char * ParseOptions(int argc, char * * argv)</code> | [function] |
| <code>void ShowVersion()</code> | [function] |
| <code>void ShowHelp(char * *)</code> | [function] |
| <code>void AddCc2Option(const char * arg)</code> | [function] |
| <code>void RecordCmdOption(char * option)</code> | [function] |
| <code>void ParseCcOptions(char * arg, char * & source_file)</code> | [function] |
| <code>const int DigitOffset</code> | [global variable] |
| HashTableEntry | [struct] |
| <code>char * key</code> | [data member] |
| <code>HashValue value</code> | [data member] |
| <code>int main(int argc, char * * argv)</code> | [function] |
| <code>opcxx_ListOfMetaclass * QuoteClassCreator</code> | [global variable] |
| <code>opcxx_ListOfMetaclass * metaclassCreator</code> | [global variable] |
| | |
| <code>Class * CreateQuoteClass(Ptree * def, Ptree * marg)</code> | [function] |
| <code>opcxx_ListOfMetaclass * opcxx_init_QuoteClass()</code> | [function] |
| <code>Class * CreateMetaclass(Ptree * def, Ptree * marg)</code> | [function] |
| <code>opcxx_ListOfMetaclass * opcxx_init_Metaclass()</code> | [function] |
| <code>opcxx_ListOfMetaclass * classCreator</code> | [global variable] |
| <code>opcxx_ListOfMetaclass * templateCreator</code> | [global variable] |

| | |
|--|-------------------|
| <code>Class * CreateClass(Ptree * def, Ptree * marg)</code> | [function] |
| <code>Class * CreateTemplateClass(Ptree * def, Ptree * marg)</code> | [function] |
| <code>opcxx_ListOfMetaClass * opcxx_init_Class()</code> | [function] |
| <code>opcxx_ListOfMetaClass * opcxx_init_TemplateClass()</code> | [function] |
| <code>const int MaxErrors</code> | [global variable] |
| <code>const int MAX</code> | [global variable] |
| <code>[] Ptree * * resultsArgs</code> | [global variable] |
| <code>int resultsIndex</code> | [global variable] |
| <code>int CountArgs(char * pat)</code> | [function] |
| <code>char * SkipSpaces(char * pat)</code> | [function] |
| <code>void MopErrorMessage(char * where, char * msg)</code> | [function] |
| <code>void MopErrorMessage2(char * msg1, char * msg2)</code> | [function] |
| <code>void MopWarningMessage(char * where, char * msg)</code> | [function] |
| <code>void MopWarningMessage2(char * msg1, char * msg2)</code> | [function] |
| <code>void MopMoreWarningMessage(char * msg1, char * msg2)</code> | [function] |
| <code>function_type func</code> | [global variable] |
| <code>__locale_struct</code> | [struct] |
| <code>[] __locale_struct::locale_data * __locales</code> | [data member] |
| <code>const unsigned short * __ctype_b</code> | [data member] |
| <code>const int * __ctype_tolower</code> | [data member] |

```

    const int * __ctype_toupper           [data member]

    [] const char * __names               [data member]

    __locale_struct * __locale_t         [typedef]

    __locale_t uselocale( __locale_t __dataset) [function]

    __gnu_cxx                             [namespace]

    function_type __uselocale             [namespace variable]

void InitializeOtherKeywords()           [function]

rw_table                                 [struct]

    char * name                           [data member]

    long value                             [data member]

    [] rw_table table                     [global variable]

ScopedName extend( const ScopedName & name, const [function]
                  std::string & str)
    Utility method

{builder.cc}                             [module]

    ostream_ptr_iterator                  [class]
    This class is very similar to ostream_iterator, except that it
    works on pointers to types

    std::ostream * out                    [data member]

    const char * sep                      [data member]

    ostream_ptr_iterator                  [member on function]
    ostream_ptr_iterator( std::ostream & o,
                          const char * s)

    ostream_ptr_iterator                  [member on function]
    const char *<const char *> & operator=(
    const const char *<const char *> & * value)

    ostream_ptr_iterator                  [member on function]
    const const char *<const char *> &
    *<const const char *<const char *> &
    *> & operator*()

```

```
ostream_ptr_iterator [member on function]
    const const char *<const char *> &
    *<const const char *<const char *> &
    *> &< const const char *<const char *>
    & *<const const char *<const char *> &
    *> &> & operator++()
```

```
ostream_ptr_iterator [member on function]
    const const char *<const char *> &
    *<const const char *<const char *> &
    *> &< const const char *<const char *>
    & *<const const char *<const char *> &
    *> &> &< const const char *<const char
    *> & *<const const char *<const char
    *> & *> &< const const char *<const
    char *> & *<const const char *<const
    char *> & *> &> & operator++( int )
```

InheritanceAdder [class]

```
std::list< AST::Class *> & open_list [data member]
```

```
InheritanceAdder [member on function]
    InheritanceAdder( std::list< AST::Class *> & l )
```

```
InheritanceAdder [member on function]
    InheritanceAdder( const InheritanceAdder & i )
```

```
InheritanceAdder void [member on function]
    operator()( AST::Inheritance * i )
```

```
std::multimap<std::string, Types::Named *> [typedef]
name_map
```

```
std::string append( const [function]
    std::vector<std::string> & strs, const std::string &
    sep)
```

```
bool isStructor( const AST::Function * func) [function]
```

```
{filter.cc} [module]
```

```
FileFilter * filter_instance [module variable]
    The FileFilter instance
```

```
{link.cc} [module]
    Static namespace for link module
```

| | |
|--|---------------|
| Link | [struct] |
| Encapsulation of a link fragment. Links can be of several types, such as start, end, or some other formatting type. They are split into start and end so that spans can be nested properly | |
| std::vector<std::string> Name | [typedef] |
| A scoped name type | |
| int line | [data member] |
| The line and column of the link | |
| int col | [data member] |
| Type | [enum] |
| Enumeration of the type of link. These should be in order of priority, so nested types should be nested in this list: <a>foo means enum ordering: A_START,B_START,B_END,A_END | |
| LINK_START | [enumerator] |
| REF_START | [enumerator] |
| < Start of a label link | |
| SPAN_START | [enumerator] |
| < Start of reference link | |
| SPAN_END | [enumerator] |
| REF_END | [enumerator] |
| LINK_END | [enumerator] |
| Type type | [data member] |
| The type of this link | |
| Name name | [data member] |
| The scoped name of this link | |
| std::string desc | [data member] |
| The description of this link | |
| lt_col | [struct] |
| Less-than functor that compares column and type | |


```

    lt_col bool operator()( [member on function]
                           const Link * a, const Link * b)

std::set< Link *, lt_col> Line [typedef]
    Set of links sorted by column

std::map<int, Line> Map [typedef]
    Map of Lines keyed by line number

Link std::ostream & write( [member on function]
                          std::ostream & o)
    Write link to the output stream for debuggingDebug-
    ging output method for Links

std::ostream & operator<<( std::ostream & [function]
                          o, const Link::Map::value_type & linepair)
    Debugging output operator for members of the Link Map

const char * input_filename [module variable]
    Filename of the input source file

const char * output_filename [module variable]
    Filename of the output HTML file

const char * links_filename [module variable]
    Filename of the links file

const char * links_scope [module variable]
    Scope to prepend to links before to find in TOC

std::vector<std::string> [module variable]
toc_filenames
    A list of TOC's to load

bool links_append [module variable]
    True if should append to output file. Note that this only
    works if the process previously using the file has flushed
    things to disk!

Link::Map links [module variable]
    A map of links to insert into the output

std::map<std::string, std::string> TOC [typedef]
    Type of the TableOfContents

```

| | |
|---|-------------------|
| TOC toc | [module variable] |
| The TOC used for looking up hrefs | |
| void parse_args(int argc, char * * argv) | [function] |
| Parses the command line arguments given to main() | |
| void write(std::ostream & out, int col, char * buf, int len, int buflen) | [function] |
| Writes some text to the output. It replaces chars that might be confused by HTML, such as < and >. All spaces are replaced with non-breaking spaces, and tabs are expanded to 8-col tabstops (hence the col argument) | |
| void write_lineno(std::ostream & out, int line) | [function] |
| Writes the line number to the output | |
| std::string decode(const std::string & str) | [function] |
| Undoes the %FF encoding | |
| void write_indent(std::ostream & out, char * buf, int & col, int buflen) | [function] |
| Writes whatever indent there is in the buf to the output using the special span. 'col' is modified to the first non-indent character. | |
| bool is_duplicate(Link * link, int len) | [function] |
| Returns true if the link is a duplicate. | |
| void read_links() | [function] |
| Reads the links file into the 'links' map. | |
| void dump_links() | [function] |
| Debugging method to dump all links to cout. | |
| void read_tocs() | [function] |
| Reads in the TOC files. The filenames are taken from the toc_filenames array and store in the 'toc' map. | |
| void link_file() | [function] |
| Reads the input file, inserts links, and writes the result to the output. It uses the 'links' line map to iterate through the file sequentially. | |

`void reset()` [function]
Clear all global vars

`PyObject * linkError` [global variable]

`PyObject * py_link(PyObject * self, PyObject * args)` [function]
The main python method, equivalent to the `main()` function

`[] PyMethodDef link_methods` [global variable]
The list of methods in this python module

`void initlink()` [function]
The initialisation method for this module

`{link_map.cc}` [module]

`Node` [struct]

`int start` [data member]

`int end` [data member]

`Type` [enum]

`START` [enumerator]

`END` [enumerator]

`Type type` [data member]

`int diff` [data member]

`Node bool operator<(const Node & other)` [member on function]

`std::set<Node> Line` [typedef]

`std::map<int, Line> LineMap` [typedef]

`void synopsis_macro_hook(const char * name, int line, int start, int end, int diff)` [function]
This function is a callback from the ucpp code to store macro expansions

`void synopsis_include_hook(const char * source_file, const char * target_file, int is_macro, int is_next)` [function]
This function is a callback from the ucpp code to store includes

```
void synopsis_define_hook( const char *           [function]
                          filename, int line, const char * name, int num_args,
                          const char * * args, int vaarg, const char * text)
```

This function is a callback from the ucpp code to store macro definitions

```
{linkstore.cc} [module]
```

```
const char * FS [module variable]
    The field separator
```

```
const char * RS [module variable]
    The record separator
```

```
[] const char * context_names [module variable]
```

```
void makedirs( const char * path) [function]
```

```
TypeStorer [class]
```

parents: parent class A class which acts as a Types Visitor to store the correct link to a given type

```
LinkStore * links [data member]
```

```
Ptree * node [data member]
```

```
LinkStore::Context context [data member]
```

```
TypeStorer TypeStorer( [member on function]
                       LinkStore * ls, Ptree * n, LinkStore::Context c)
```

Constructor

```
TypeStorer std::string describe( [member on function]
                                  Types::Type * type)
    Returns a suitable description for the given type
```

```
TypeStorer void visit_base( [member on function]
                             Types::Base * base)
```

```
TypeStorer void visit_named( [member on function]
                              Types::Named * named)
```

```
TypeStorer void visit_declared( [member on function]
                                  Types::Declared * declared)
```

```

TypeStorer void visit_modifier(      [member on function]
    Types::Modifier * mod)

TypeStorer void                      [member on function]
    visit_parameterized( Types::Parameterized *
    param)

std::ostream & operator<<( std::ostream & out,      [function]
    const LinkStore::encode & enc)

std::ostream & operator<<( std::ostream & out,      [function]
    const LinkStore::encode_name & enc)

{lookup.cc}                          [module]

ostream_ptr_iterator                  [class]
    This class is very similar to ostream_iterator, except that it
    works on pointers to types

std::ostream * out                    [data member]

const char * sep                      [data member]

ostream_ptr_iterator                  [member on function]
    ostream_ptr_iterator( std::ostream & o,
    const char * s)

ostream_ptr_iterator                  [member on function]
    const char * <const char * > & operator=(
    const const char * <const char * > & * value)

ostream_ptr_iterator                  [member on function]
    const const char * <const char * > &
    * <const const char * <const char * > &
    * > & operator*()

ostream_ptr_iterator                  [member on function]
    const const char * <const char * > &
    * <const const char * <const char * > &
    * > & < const const char * <const char * >
    & * <const const char * <const char * > &
    * > & > & operator++()

```

```
ostream_ptr_iterator [member on function]
    const const char *<const char *> &
    *<const const char *<const char *> &
    *> &< const const char *<const char *>
    & *<const const char *<const char *> &
    *> &> &< const const char *<const char
    *> & *<const const char *<const char
    *> & *> &< const const char *<const
    char *> & *<const const char *<const
    char *> & *> &> &> & operator++( int )
```

isType [class]

parents: parent class Predicate class that is true if the object passed to the constructor is a type, as opposed to a modifier or a parametrized, etc

bool m_value [data member]

isType isType(Types::Named * type) [member on function]

constructor. Visits the given type thereby setting the value

isType bool (bool)() [member on function]
bool operator, returns the value determined by visitation during construction

isType void visit_base(Types::Base *) [member on function]
Okay

isType void visit_unknown(Types::Unknown *) [member on function]
Okay

isType void visit_declared(Types::Declared * type) [member on function]
Okay if not a function declaration

isType void visit_dependent(Types::Dependent *) [member on function]
Okay if a template dependent arg

isType void visit_type(Types::Type *) [member on function]
Fallback: Not okay

| | |
|---|----------------------|
| FunctionHeuristic | [class] |
| std::vector< Types::Type *> v_Type | [typedef] |
| v_Type::iterator vi_Type | [typedef] |
| std::vector< AST::Parameter *> v_Param | [typedef] |
| v_Param::iterator vi_Param | [typedef] |
| v_Type m_args | [data member] |
| int cost | [data member] |
| FunctionHeuristic | [member on function] |
| FunctionHeuristic(const v_Type & v) Constructor - takes arguments to match functions against | |
| FunctionHeuristic int | [member on function] |
| operator()(AST::Function * func) Heuristic operator, returns 'cost' of given function - higher is worse, 1000 means no match | |
| FunctionHeuristic bool | [member on function] |
| hasEllipsis(v_Param * params) Find an ellipsis as the last arg | |
| FunctionHeuristic int | [member on function] |
| countDefault(v_Param * params) Returns the number of parameters with default values. Counts from the back and stops when it finds one without a default. | |
| FunctionHeuristic void | [member on function] |
| calcCost(Types::Type * arg_t, Types::Type * param_t) Calculate the cost of converting 'arg' into 'param'. The cost is accumulated on the 'cost' member variable. | |
| int ucpp_main(int argc, char ** argv) | [function] |
| bool verbose | [global variable] |
| bool syn_main_only | [global variable] |
| bool syn_extract_tails | [global variable] |

```

bool syn_use_gcc [global variable]
bool syn_fake_std [global variable]
bool syn_multi_files [global variable]
const char * syn_basename [global variable]
const char * syn_syntax_prefix [global variable]
const char * syn_xref_prefix [global variable]
const char * syn_file_syntax [global variable]
const char * syn_file_xref [global variable]
const char * syn_emulate_compiler [global variable]
std::vector<const char *> * syn_extra_filenames [global variable]
PyThreadState * pthread_save [global variable]
{occ.cc} [module]

void unexpected() [function]
    Override unexpected() to print a message before we abort

void getopts( PyObject * args, [function]
    std::vector<const char *> & cppflags,
    std::vector<const char *> & occlflags, PyObject * config,
    PyObject * extra_files)

void emulate_compiler( std::vector<const [function]
    char *> & args)
    Emulates the compiler in 'syn-emulate-compiler' by calling
    the Python module Synopsis.Parser.C++.emul to get a list of
    include paths and macros to add to the args vector.

char * RunPreprocessor( const char * [function]
    file, const std::vector<const char *> & flags)

void sighandler( int signo) [function]

void RunOpencxx( const char * src, const [function]
    char * file, const std::vector<const char *> &
    args, PyObject * ast, PyObject * types,
    PyObject * declarations, PyObject * files)

```



```

void do_parse( const char * src, const [function]
               std::vector<const char *> & cppargs, const
               std::vector<const char *> & occargs, PyObject *
               ast, PyObject * types, PyObject * declarations,
               PyObject * files)

PyObject * occParse( PyObject * self, [function]
                    PyObject * args)

PyObject * occUsage( PyObject * self, [function]
                    PyObject * )

[] PyMethodDef occ_methods [module variable]

void initocc() [function]

AST::Comment * make_Comment( [function]
                              AST::SourceFile * file, int line, Ptree * first, bool
                              suspect)

Leaf * make_Leaf( char * pos, int len) [function]

TypeResolver [class]
  parents: parent class Resolves the final type of any given Type.
  For example, it traverses typedefs and parameterized types, and
  resolves unknowns by looking them up.

TypeResolver TypeResolver( [member on function]
                           Builder * b)
  Constructor - needs a Builder to resolve unknowns with

TypeResolver Types::Type * [member on function]
             resolve( Types::Type * t)
             Resolves the given type object

TypeResolver AST::Scope * [member on function]
             scope( Types::Type * t)
             Tries to resolve the given type object to a Scope

TypeResolver void visit_unknown( [member on function]
                                 Types::Unknown * t)
             Looks up the unknown type for a fresh definition

TypeResolver void [member on function]
             visit_modifier( Types::Modifier * t)
             Recursively processes the aliased type

```

```

TypeResolver void visit_declared( Types::Declared * t) [member on function]
    Checks for typedefs and recursively processes them

TypeResolver void visit_parameterized( Types::Parameterized * t) [member on function]
    Processes the template type

Builder * m_builder [data member]

Types::Type * m_type [data member]
    < A reference to the builder object

void nullObj() [function]

ProgramFile [class]
    parents: parent class

ProgramFile ProgramFile( [member on function]
    std::ifstream & , char * filename)

ProgramFile ~ProgramFile() [member on function]

ProgramFromStdin [class]
    parents: parent class

ProgramFromStdin ProgramFromStdin() [member on function]

ProgramFromStdin ~ProgramFromStdin() [member on function]

ProgramFromStdin char Get() [member on function]

uint buf_size [data member]

ProgramString [class]
    parents: parent class

ProgramString ProgramString() [member on function]

ProgramString ~ProgramString() [member on function]

ProgramString void Clear() [member on function]

ProgramString uint Length() [member on function]

```

ProgramString **ProgramString** [member on function]
 & operator<<(const char *)

ProgramString **ProgramString** [member on function]
 & operator<<(const char)

uint **str_length** [data member]

ClassBodyWalker [class]

parents: parent class

ClassBodyWalker [member on function]
 ClassBodyWalker(Walker * w, PtreeArray *
 tlist)

ClassBodyWalker **Ptree *** [member on function]
 TranslateClassBody(Ptree * block, Ptree *
 bases, Class *)

ClassBodyWalker **void** [member on function]
 AppendNewMembers(Class * , PtreeArray & , bool
 &)

ClassBodyWalker **Ptree *** [member on function]
 TranslateTypespecifier(Ptree * tspec)

ClassBodyWalker **Ptree *** [member on function]
 TranslateTypedef(Ptree * def)

ClassBodyWalker **Ptree *** [member on function]
 TranslateMetaclassDecl(Ptree * decl)

ClassBodyWalker **Ptree *** [member on function]
 TranslateDeclarators(Ptree * decls)

ClassBodyWalker **Ptree *** [member on function]
 TranslateAssignInitializer(PtreeDeclarator *
 decl, Ptree * init)

ClassBodyWalker **Ptree *** [member on function]
 TranslateInitializeArgs(PtreeDeclarator *
 decl, Ptree * init)

ClassBodyWalker **Ptree *** [member on function]
 TranslateDeclarator(bool record,
 PtreeDeclarator * decl)

```

ClassBodyWalker Ptree *          [member on function]
    TranslateDeclarator( bool record,
    PtreeDeclarator * decl, bool append_body)

ClassBodyWalker Ptree *          [member on function]
    TranslateFunctionImplementation( Ptree * impl)

PtreeArray * tspec_list          [data member]

ClassWalker                                [class]
    parents: parent class

ClassWalker ClassWalker( Parser      [member on function]
    * p)

ClassWalker ClassWalker( Parser      [member on function]
    * p, Environment * e)

ClassWalker ClassWalker(              [member on function]
    Environment * e)

ClassWalker ClassWalker( Walker      [member on function]
    * w)

ClassWalker bool IsClassWalker()      [member on function]

ClassWalker void                    [member on function]
    InsertBeforeStatement( Ptree * )

ClassWalker void                    [member on function]
    AppendAfterStatement( Ptree * )

ClassWalker void                    [member on function]
    InsertBeforeToplevel( Ptree * )

ClassWalker void                    [member on function]
    AppendAfterToplevel( Ptree * )

ClassWalker bool                    [member on function]
    InsertDeclaration( Ptree * , Class * , Ptree * ,
    void * )

ClassWalker void *                  [member on function]
    LookupClientData( Class * , Ptree * )

ClassWalker Ptree *                [member on function]
    GetInsertedPtree()

```

```

ClassWalker Ptree * [member on function]
    GetAppendedPtree()

ClassWalker Ptree * [member on function]
    TranslateMetaclassDecl( Ptree * decl)

ClassWalker Ptree * [member on function]
    TranslateClassSpec( Ptree * spec, Ptree *
        userkey, Ptree * class_def, Class * metaobject)

ClassWalker Ptree * [member on function]
    TranslateTemplateInstantiation( Ptree * spec,
        Ptree * userkey, Ptree * class_def, Class *
        metaobject)

ClassWalker Ptree * [member on function]
    ConstructClass( Class * metaobject)

ClassWalker Ptree * [member on function]
    ConstructAccessSpecifier( int access)

ClassWalker Ptree * [member on function]
    ConstructMember( void * )

ClassWalker Ptree * [member on function]
    TranslateStorageSpecifiers( Ptree * )

ClassWalker Ptree * [member on function]
    TranslateTemplateFunction( Ptree * temp_def,
        Ptree * impl)

ClassWalker Class * [member on function]
    MakeMetaobjectForCfunctions()

ClassWalker Ptree * [member on function]
    TranslateFunctionImplementation( Ptree * )

ClassWalker Ptree * [member on function]
    MakeMemberDeclarator( bool record, void * ,
        PtreeDeclarator * )

ClassWalker Ptree * [member on function]
    RecordArgsAndTranslateFbody( Class * , Ptree *
        args, Ptree * body)

ClassWalker Ptree * [member on function]
    TranslateFunctionBody( Ptree * )

```

| | |
|--|----------------------|
| ClassWalker Ptree * | [member on function] |
| TranslateBlock(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateArgDeclList(bool , Ptree * , Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateInitializeArgs(PtreeDeclarator * , Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateAssignInitializer(PtreeDeclarator * , Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateUserAccessSpec(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateAssign(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateInfix(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateUnary(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateArray(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslatePostfix(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateFuncall(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateDotMember(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateArrowMember(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateThis(Ptree *) | |
| ClassWalker Ptree * | [member on function] |
| TranslateVariable(Ptree *) | |

```

ClassWalker Ptree * [member on function]
    TranslateUserStatement( Ptree * )

ClassWalker Ptree * [member on function]
    TranslateStaticUserStatement( Ptree * )

ClassWalker Ptree * [member on function]
    TranslateNew2( Ptree * , Ptree * , Ptree * ,
    Ptree * , Ptree * , Ptree * , Ptree * )

ClassWalker Ptree * [member on function]
    TranslateDelete( Ptree * )

ClassWalker Class * [member on function]
    GetClassMetaobject( TypeInfo & )

ClassWalker PtreeArray * [member on function]
    RecordMembers( Ptree * , Ptree * , Class * )

ClassWalker void [member on function]
    RecordMemberDeclaration( Ptree * mem,
    PtreeArray * tspec_list)

ClassWalker Ptree * [member on function]
    TranslateStorageSpecifiers2( Ptree * rest)

ClassWalker Ptree * [member on function]
    CheckMemberEquiv( Ptree * , Ptree * )

ClassWalker Ptree * CheckEquiv( [member on function]
    Ptree * p, Ptree * q)

ClientDataLink [struct]
    parents: parent class

    ClientDataLink * next [data member]
    Class * metaobject [data member]
    Ptree * key [data member]
    void * data [data member]

PtreeArray before_statement [data member]
PtreeArray after_statement [data member]
PtreeArray before_toplevel [data member]

```

| | | |
|-------------------------------|--|----------------------|
| <code>PtreeArray</code> | <code>after_toplevel</code> | [data member] |
| <code>PtreeArray</code> | <code>inserted_declarations</code> | [data member] |
| <code>ClientDataLink *</code> | <code>client_data</code> | [data member] |
| Encoding | | [class] |
| <code>Encoding</code> | <code>Encoding()</code> | [member on function] |
| <code>Encoding</code> | <code>Encoding(Encoding & e)</code> | [member on function] |
| <code>Encoding</code> | <code>void do_init_static()</code> | [member on function] |
| <code>Encoding</code> | <code>void Clear()</code> | [member on function] |
| <code>Encoding</code> | <code>void Reset(Encoding &)</code> | [member on function] |
| <code>Encoding</code> | <code>char * Get()</code> | [member on function] |
| <code>Encoding</code> | <code>bool IsEmpty()</code> | [member on function] |
| <code>Encoding</code> | <code>void CvQualify(Ptree * , Ptree *)</code> | [member on function] |
| <code>Encoding</code> | <code>void SimpleConst()</code> | [member on function] |
| <code>Encoding</code> | <code>void GlobalScope()</code> | [member on function] |
| <code>Encoding</code> | <code>void SimpleName(Ptree *)</code> | [member on function] |
| <code>Encoding</code> | <code>void NoName()</code> | [member on function] |
| <code>Encoding</code> | <code>void Template(Ptree * , Encoding &)</code> | [member on function] |
| <code>Encoding</code> | <code>void Qualified(int)</code> | [member on function] |
| <code>Encoding</code> | <code>void Destructor(Ptree *)</code> | [member on function] |
| <code>Encoding</code> | <code>void PtrOperator(int)</code> | [member on function] |
| <code>Encoding</code> | <code>void PtrToMember(Encoding & , int)</code> | [member on function] |

| | |
|---|----------------------|
| Encoding void CastOperator(Encoding &) | [member on function] |
| Encoding void Array() | [member on function] |
| Encoding void Function(Encoding & args) | [member on function] |
| Encoding void Recursion(Encoding & e) | [member on function] |
| Encoding void StartFuncArgs() | [member on function] |
| Encoding void EndFuncArgs() | [member on function] |
| Encoding void Void() | [member on function] |
| Encoding void EllipsisArg() | [member on function] |
| Encoding void NoReturnType() | [member on function] |
| Encoding void ValueTempParam() | [member on function] |
| Encoding void Insert(unsigned char) | [member on function] |
| Encoding void Insert(char * , int) | [member on function] |
| Encoding void Append(unsigned char) | [member on function] |
| Encoding void Append(char * , int) | [member on function] |
| Encoding void Append(Encoding & e) | [member on function] |
| Encoding void AppendWithLen(char * , int) | [member on function] |
| Encoding void AppendWithLen(Encoding & e) | [member on function] |
| Encoding void Print(std::ostream & , char *) | [member on function] |
| Encoding char * GetBaseName(char * , int & , Environment * &) | [member on function] |

| | |
|--|----------------------|
| Encoding Ptree * MakePtree (unsigned char * & , Ptree *) | [member on function] |
| Encoding Ptree * MakeQname (unsigned char * &) | [member on function] |
| Encoding Ptree * MakeLeaf (unsigned char * &) | [member on function] |
| Encoding bool IsSimpleName (unsigned char *) | [member on function] |
| Encoding Ptree * NameToPtree (char * , int) | [member on function] |
| Encoding unsigned char * GetTemplateArguments (unsigned char * , int &) | [member on function] |
| Encoding Environment * ResolveTypedefName (Environment * , char * , int) | [member on function] |
| Encoding int GetBaseNameIfTemplate (unsigned char * , Environment * &) | [member on function] |
| [] unsigned char name | [data member] |
| int len | [data member] |
| Ptree * bool_t | [data member] |
| Ptree * char_t | [data member] |
| Ptree * int_t | [data member] |
| Ptree * short_t | [data member] |
| Ptree * long_t | [data member] |
| Ptree * float_t | [data member] |
| Ptree * double_t | [data member] |
| Ptree * void_t | [data member] |
| Ptree * signed_t | [data member] |

| | |
|---------------------------------|---------------|
| Ptree * unsigned_t | [data member] |
| Ptree * const_t | [data member] |
| Ptree * volatile_t | [data member] |
| Ptree * operator_name | [data member] |
| Ptree * new_operator | [data member] |
| Ptree * anew_operator | [data member] |
| Ptree * delete_operator | [data member] |
| Ptree * adelete_operator | [data member] |
| Ptree * star | [data member] |
| Ptree * ampersand | [data member] |
| Ptree * comma | [data member] |
| Ptree * dots | [data member] |
| Ptree * scope | [data member] |
| Ptree * tilder | [data member] |
| Ptree * left_paren | [data member] |
| Ptree * right_paren | [data member] |
| Ptree * left_bracket | [data member] |
| Ptree * right_bracket | [data member] |
| Ptree * left_angle | [data member] |
| Ptree * right_angle | [data member] |

Environment [class]

parents: parent class

Environment Environment(Walker * w) [member on function]

Environment Environment(Environment * e) [member on function]

| | |
|--|----------------------|
| Environment Environment(Environment * e, Walker * w) | [member on function] |
| Environment void do_init_static() | [member on function] |
| Environment bool IsEmpty() | [member on function] |
| Environment Environment * GetOuterEnvironment() | [member on function] |
| Environment Environment * GetBottom() | [member on function] |
| Environment void AddBaseclassEnv(Environment * e) | [member on function] |
| Environment Walker * GetWalker() | [member on function] |
| Environment void SetWalker(Walker * w) | [member on function] |
| Environment Class * LookupClassMetaobject(Ptree * name) | [member on function] |
| Environment bool LookupType(const char * name, int len, Bind * & t) | [member on function] |
| Environment bool Lookup(Ptree * name, bool & is_type_name, TypeInfo & t) | [member on function] |
| Environment bool Lookup(Ptree * name, TypeInfo & t) | [member on function] |
| Environment bool Lookup(Ptree * , Bind * &) | [member on function] |
| Environment bool LookupTop(Ptree * , Bind * &) | [member on function] |
| Environment bool LookupTop(const char * name, int len, Bind * & t) | [member on function] |
| Environment bool LookupAll(const char * name, int len, Bind * & t) | [member on function] |
| Environment bool RecordVariable(char * name, Class * c) | [member on function] |

```

Environment bool [member on function]
    RecordPointerVariable( char * name, Class * c )

Environment int AddEntry( char * [member on function]
    , int , Bind * )

Environment int AddDupEntry( [member on function]
    char * , int , Bind * )

Environment void [member on function]
    RecordNamespace( Ptree * )

Environment bool [member on function]
    LookupNamespace( char * , int )

Environment void [member on function]
    RecordTypedefName( Ptree * )

Environment void RecordEnumName( [member on function]
    Ptree * )

Environment void [member on function]
    RecordClassName( char * , Class * )

Environment void [member on function]
    RecordTemplateClass( Ptree * , Class * )

Environment Environment * [member on function]
    RecordTemplateFunction( Ptree * , Ptree * )

Environment Environment * [member on function]
    RecordDeclarator( Ptree * )

Environment Environment * [member on function]
    DontRecordDeclarator( Ptree * )

Environment void [member on function]
    RecordMetaclassName( Ptree * )

Environment Ptree * [member on function]
    LookupMetaclass( Ptree * )

Environment bool [member on function]
    RecordClasskeyword( char * , char * )

Environment Ptree * [member on function]
    LookupClasskeyword( Ptree * )

```

| | |
|--|----------------------|
| Environment void SetMetaobject(Class * m) | [member on function] |
| Environment Class * IsClassEnvironment() | [member on function] |
| Environment Class * LookupThis() | [member on function] |
| Environment Environment * IsMember(Ptree *) | [member on function] |
| Environment void Dump() | [member on function] |
| Environment void Dump(int) | [member on function] |
| Environment Ptree * GetLineNumber(Ptree * , int &) | [member on function] |
| Array | [class] |
| parents: parent class | |
| Array Array (int) | [member on function] |
| Array uint Number() | [member on function] |
| Array Environment * Ref(uint index) | [member on function] |
| Array void Append(Environment *) | [member on function] |
| uint num | [data member] |
| uint size | [data member] |
| Environment * * array | [data member] |
| Environment * next | [data member] |
| HashTable * htable | [data member] |
| Class * metaobject | [data member] |
| Walker * walker | [data member] |
| PtreeArray metaclasses | [data member] |

| | |
|---|----------------------|
| <code>PtreeArray * classkeywords</code> | [data member] |
| <code>Array baseclasses</code> | [data member] |
| <code>HashTable * namespace_table</code> | [data member] |
| Bind | [class] |
| parents: parent class | |
| Kind | [enum] |
| isVarName | [enumerator] |
| isTypedefName | [enumerator] |
| isClassName | [enumerator] |
| isEnumName | [enumerator] |
| isTemplateClass | [enumerator] |
| isTemplateFunction | [enumerator] |
| <code>Bind Kind What()</code> | [member on function] |
| <code>Bind void GetType(TypeInfo & , Environment *)</code> | [member on function] |
| <code>Bind char * GetEncodedType()</code> | [member on function] |
| <code>Bind bool IsType()</code> | [member on function] |
| <code>Bind Class * ClassMetaobject()</code> | [member on function] |
| <code>Bind void SetClassMetaobject(Class *)</code> | [member on function] |
| BindVarName | [class] |
| parents: parent class | |
| <code>BindVarName BindVarName(char * t)</code> | [member on function] |
| <code>BindVarName Bind::Kind What()</code> | [member on function] |
| <code>BindVarName void GetType(TypeInfo & , Environment *)</code> | [member on function] |

| | | |
|------------------------------|--|----------------------|
| <code>BindVarName</code> | <code>char *</code> <code>GetEncodedType()</code> | [member on function] |
| <code>BindVarName</code> | <code>bool</code> <code>IsType()</code> | [member on function] |
| | <code>char * type</code> | [data member] |
| BindTypedefName | | [class] |
| | parents: parent class | |
| <code>BindTypedefName</code> | <code>BindTypedefName(char * t)</code> | [member on function] |
| <code>BindTypedefName</code> | Bind::Kind <code>What()</code> | [member on function] |
| <code>BindTypedefName</code> | <code>void</code> <code>GetType(TypeInfo & , Environment *)</code> | [member on function] |
| <code>BindTypedefName</code> | <code>char *</code> <code>GetEncodedType()</code> | [member on function] |
| | <code>char * type</code> | [data member] |
| BindClassName | | [class] |
| | parents: parent class | |
| <code>BindClassName</code> | <code>BindClassName(Class * c)</code> | [member on function] |
| <code>BindClassName</code> | Bind::Kind <code>What()</code> | [member on function] |
| <code>BindClassName</code> | <code>void</code> <code>GetType(TypeInfo & , Environment *)</code> | [member on function] |
| <code>BindClassName</code> | Class * <code>ClassMetaobject()</code> | [member on function] |
| <code>BindClassName</code> | <code>void</code> <code>SetClassMetaobject(Class *)</code> | [member on function] |
| | <code>Class * metaobject</code> | [data member] |
| BindEnumName | | [class] |
| | parents: parent class | |


```

BindEnumName BindEnumName( char      [member on function]
    * , Ptree * )

BindEnumName Bind::Kind What()      [member on function]

BindEnumName void GetType(          [member on function]
    TypeInfo & , Environment * )

BindEnumName Ptree *              [member on function]
    GetSpecification()

char * type                          [data member]

Ptree * specification                [data member]

BindTemplateClass                    [class]
parents: parent class

BindTemplateClass                    [member on function]
    BindTemplateClass( Class * c )

BindTemplateClass Bind::Kind        [member on function]
    What()

BindTemplateClass void GetType(      [member on function]
    TypeInfo & , Environment * )

BindTemplateClass Class *          [member on function]
    ClassMetaobject()

BindTemplateClass void              [member on function]
    SetClassMetaobject( Class * )

Class * metaobject                  [data member]

BindTemplateFunction                [class]
parents: parent class

BindTemplateFunction                  [member on function]
    BindTemplateFunction( Ptree * d )

BindTemplateFunction                  [member on function]
    Bind::Kind What()

BindTemplateFunction void           [member on function]
    GetType( TypeInfo & , Environment * )

```

```

    BindTemplateFunction bool           [member on function]
        IsType()

    Ptree * decl                         [data member]

void * HashValue                        [typedef]

HashTable                               [class]
    parents: parent class

    HashTable HashTable()                 [member on function]

    HashTable HashTable( int )           [member on function]

    HashTable void MakeTable()          [member on function]

    HashTable bool IsEmpty()             [member on function]

    HashTable void Dump(                 [member on function]
        std::ostream & )

    HashTable int AddEntry( char *      [member on function]
        key, HashValue value, int * index)

    HashTable int AddEntry( bool ,     [member on function]
        char * key, int len, HashValue value, int *
        index)

    HashTable int AddEntry( char *      [member on function]
        key, int len, HashValue value, int * index)

    HashTable int AddDupEntry( char *   [member on function]
        key, int len, HashValue value, int * index)

    HashTable bool Lookup( char *       [member on function]
        key, HashValue * value)

    HashTable bool Lookup( char *       [member on function]
        key, int len, HashValue * value)

    HashTable bool LookupEntries(       [member on function]
        char * key, int len, HashValue * value, int &
        nth)

    HashTable HashValue Peek( int       [member on function]
        index)

```

| | |
|---|----------------------|
| HashTable bool RemoveEntry(char * key) | [member on function] |
| HashTable bool RemoveEntry(char * key, int len) | [member on function] |
| HashTable void ReplaceValue(int index, HashValue value) | [member on function] |
| HashTable char * KeyString(char * key) | [member on function] |
| HashTable char * KeyString(char * key, int len) | [member on function] |
| HashTable bool Lookup2(char * key, HashValue * val, int * index) | [member on function] |
| HashTable bool Lookup2(char * key, int len, HashValue * val, int * index) | [member on function] |
| HashTable uint NextPrimeNumber(uint number) | [member on function] |
| HashTable bool GrowTable(int increment) | [member on function] |
| HashTable unsigned int StringToInt(char *) | [member on function] |
| HashTable unsigned int StringToInt(char * , int) | [member on function] |
| HashTable int HashFunc(unsigned int p, int n) | [member on function] |
| HashTableEntry * entries | [data member] |
| int Size | [data member] |
| int Prime2 | [data member] |
| BigHashTable | [class] |
| parents: parent class | |
| BigHashTable BigHashTable() | [member on function] |

| | |
|---|----------------------|
| Member | [class] |
| parents: parent class | |
| Member Member() | [member on function] |
| Member Member(const Member &) | [member on function] |
| Member Member(Class * , Ptree *) | [member on function] |
| Member void Set(Class * , Ptree * , int) | [member on function] |
| Member void Signature(TypeInfo & t) | [member on function] |
| Member Ptree * Name() | [member on function] |
| Member Ptree * Comments() | [member on function] |
| Member int Nth() | [member on function] |
| Member Class * Supplier() | [member on function] |
| Member bool IsConstructor() | [member on function] |
| Member bool IsDestructor() | [member on function] |
| Member bool IsFunction() | [member on function] |
| Member bool IsPublic() | [member on function] |
| Member bool IsProtected() | [member on function] |
| Member bool IsPrivate() | [member on function] |
| Member bool IsStatic() | [member on function] |
| Member bool IsMutable() | [member on function] |
| Member bool IsInline() | [member on function] |
| Member bool IsVirtual() | [member on function] |
| Member bool IsPureVirtual() | [member on function] |
| Member Ptree * GetUserMemberModifier() | [member on function] |

| | |
|---|----------------------|
| Member Ptree * GetUserAccessSpecifier() | [member on function] |
| Member bool GetUserArgumentModifiers(PtreeArray & result) | [member on function] |
| Member void Remove() | [member on function] |
| Member void SetName(Ptree *) | [member on function] |
| Member void SetQualifiedName(Ptree *) | [member on function] |
| Member Ptree * NewName() | [member on function] |
| Member Ptree * ArgumentList() | [member on function] |
| Member void SetArgumentList(Ptree *) | [member on function] |
| Member Ptree * NewArgumentList() | [member on function] |
| Member Ptree * MemberInitializers() | [member on function] |
| Member void SetMemberInitializers(Ptree *) | [member on function] |
| Member Ptree * NewMemberInitializers() | [member on function] |
| Member Ptree * FunctionBody() | [member on function] |
| Member void SetFunctionBody(Ptree *) | [member on function] |
| Member Ptree * NewFunctionBody() | [member on function] |
| Member Ptree * Arguments() | [member on function] |
| Member void Copy(Member * , void *) | [member on function] |
| Member bool IsInlineFuncImpl() | [member on function] |
| Member void SetName(Ptree * , Ptree *) | [member on function] |

| | | |
|--------|---|----------------------|
| Member | Ptree * ArgumentList(Ptree * decl) | [member on function] |
| Member | Ptree * Arguments(Ptree * , int) | [member on function] |
| Member | Ptree * MemberInitializers(Ptree * decl) | [member on function] |
| Member | char * Name(int &) | [member on function] |
| Member | bool Find() | [member on function] |
| Member | bool IsFunctionImplementation() | [member on function] |
| | Ptree * implementation | [data member] |
| | Ptree * original_decl | [data member] |
| | bool removed | [data member] |
| | Ptree * new_name | [data member] |
| | Ptree * new_args | [data member] |
| | Ptree * new_init | [data member] |
| | Ptree * new_body | [data member] |
| | bool arg_name_filled | [data member] |
| | Class * metaobject | [data member] |
| | Ptree * declarator | [data member] |
| | int nth | [data member] |
| | MemberFunction | [class] |
| | parents: parent class | |
| | MemberFunction MemberFunction(Class * , Ptree * , Ptree *) | [member on function] |
| | MemberList | [class] |
| | parents: parent class | |
| | Mem | [struct] |

| | |
|---|----------------------|
| <code>Class * supplying</code> | [data member] |
| <code>Ptree * definition</code> | [data member] |
| <code>Ptree * declarator</code> | [data member] |
| <code>char * name</code> | [data member] |
| <code>char * signature</code> | [data member] |
| <code>bool is_constructor</code> | [data member] |
| <code>bool is_destructor</code> | [data member] |
| <code>bool is_virtual</code> | [data member] |
| <code>bool is_static</code> | [data member] |
| <code>bool is_mutable</code> | [data member] |
| <code>bool is_inline</code> | [data member] |
| <code>int access</code> | [data member] |
| <code>Ptree * user_access</code> | [data member] |
| <code>Ptree * user_mod</code> | [data member] |
| <code>MemberList MemberList()</code> | [member on function] |
| <code>MemberList void Make(Class *)</code> | [member on function] |
| <code>MemberList Mem * Ref(int)</code> | [member on function] |
| <code>MemberList int Number()</code> | [member on function] |
| <code>MemberList Mem * Lookup(char * , char *)</code> | [member on function] |
| <code>MemberList int Lookup(char * , int , char *)</code> | [member on function] |
| <code>MemberList int Lookup(Environment * , Ptree * , int)</code> | [member on function] |
| <code>MemberList int Lookup(Environment * , char * , int)</code> | [member on function] |

| | |
|---|----------------------|
| MemberList void AppendThisClass(Class *) | [member on function] |
| MemberList void Append(Ptree * , Ptree * , int , Ptree *) | [member on function] |
| MemberList void AppendBaseClass(Environment * , Ptree *) | [member on function] |
| MemberList void CheckHeader(Ptree * , Mem *) | [member on function] |
| Class * this_class | [data member] |
| int num | [data member] |
| int size | [data member] |
| Mem * array | [data member] |
| ChangedMemberList parents: parent class | [class] |
| Cmem | [struct] |
| Ptree * declarator | [data member] |
| bool removed | [data member] |
| Ptree * name | [data member] |
| Ptree * args | [data member] |
| Ptree * init | [data member] |
| Ptree * body | [data member] |
| Ptree * def | [data member] |
| int access | [data member] |
| bool arg_name_filled | [data member] |
| ChangedMemberList ChangedMemberList() | [member on function] |
| ChangedMemberList void Append(Member * , int access) | [member on function] |

ChangedMemberList **void** Copy([member on function]
Member * *src*, Cmem * *dest*, int *access*)

ChangedMemberList **Cmem** * [member on function]
Lookup(Ptree * *decl*)

ChangedMemberList **Cmem** * [member on function]
Get(int)

ChangedMemberList **Cmem** * [member on function]
Ref(int)

int **num** [data member]

int **size** [data member]

Cmem * **array** [data member]

Metaclass [class]

parents: parent class

Metaclass Metaclass() [member on function]

Metaclass **bool** Initialize() [member on function]

Metaclass **char** * [member on function]
MetaclassName()

Metaclass **void** TranslateClass([member on function]
Environment *)

Metaclass **Ptree** * [member on function]
GetFinalizer()

Metaclass **void** [member on function]
CheckObsoleteness()

Metaclass **void** ProduceInitFile([member on function]
Ptree * *class_name*)

Metaclass **bool** [member on function]
IsBuiltinMetaclass(Ptree *)

Metaclass **void** [member on function]
InsertInitialize()

Metaclass **int** [member on function]
FindFirstNotInlinedVirtualFunction()

```

MetaClass void                                     [member on function]
    TranslateMemberFunction( Environment * env,
    Class::Member & m)

MetaClass void                                     [member on function]
    AppendHousekeepingCode( Environment * env,
    Ptree * class_name, Ptree * creator_name, Ptree
    * finalizer)

MetaClass void Load( Ptree *                       [member on function]
    metaclass_name)

MetaClass void Load( char *                         [member on function]
    metaclass_name, int len)

MetaClass void * LoadSoLib(                        [member on function]
    char * file_name)

MetaClass void * LookupSymbol(                    [member on function]
    void * handle, char * symbol)

MetaClass void do_init_static()                   [member on function]

Ptree * new_function_name                         [data member]

int first_not_inlined_vf                         [data member]

TemplateClass                                     [class]
    parents: parent class

TemplateClass void                               [member on function]
    InitializeInstance( Ptree * def, Ptree * margs)

TemplateClass bool Initialize()                   [member on function]

TemplateClass char *                             [member on function]
    MetaclassName()

TemplateClass Ptree *                            [member on function]
    TemplateDefinition()

TemplateClass Ptree *                            [member on function]
    TemplateArguments()

TemplateClass bool                               [member on function]
    AcceptTemplate()

```

```

TemplateClass Ptree * [member on function]
    TranslateInstantiation( Environment * , Ptree *
    )

TemplateClass Ptree * [member on function]
    GetClassInTemplate( Ptree * def )

Ptree * template_definition [data member]

ClassArray [class]
    parents: parent class

ClassArray ClassArray( int ) [member on function]

ClassArray uint Number() [member on function]

ClassArray Class * & [member on function]
    operator[] ( uint index )

ClassArray Class * & Ref( uint [member on function]
    index )

ClassArray void Append( Class * ) [member on function]

ClassArray void Clear() [member on function]

uint num [data member]

uint size [data member]

Class * * array [data member]

function_type opcxx_MetaclassCreator [typedef]

opcxx_ListOfMetaclass [class]

opcxx_ListOfMetaclass [member on function]
    opcxx_ListOfMetaclass( char * ,
    opcxx_MetaclassCreator , function_type ,
    function_type )

opcxx_ListOfMetaclass Class * [member on function]
    New( Ptree * , Ptree * , Ptree * )

opcxx_ListOfMetaclass Class * [member on function]
    New( char * , Ptree * , Ptree * )

```

| | | | |
|-------------------------------------|--|--|----------------------|
| <code>opcxx_ListOfMetaclass</code> | void | <code>FinalizeAll(std::ostream &)</code> | [member on function] |
| <code>opcxx_ListOfMetaclass</code> | bool | <code>AlreadyRecorded(char *)</code> | [member on function] |
| <code>opcxx_ListOfMetaclass</code> | bool | <code>AlreadyRecorded(Ptree *)</code> | [member on function] |
| <code>opcxx_ListOfMetaclass</code> | void | <code>PrintAllMetaClasses()</code> | [member on function] |
| <code>opcxx_ListOfMetaclass</code> | * next | | [data member] |
| <code>char *</code> | name | | [data member] |
| <code>opcxx_MetaClassCreator</code> | proc | | [data member] |
| <code>function_type</code> | finalizer | | [data member] |
| <code>opcxx_ListOfMetaclass</code> | * head | | [data member] |
| Parser | | | [class] |
| | parents: | parent class | |
| <code>Parser</code> | <code>Parser(Lex *)</code> | | [member on function] |
| <code>Parser</code> | bool <code>ErrorMessage(const char * , Ptree * , Ptree *)</code> | | [member on function] |
| <code>Parser</code> | void <code>WarningMessage(const char * , Ptree * , Ptree *)</code> | | [member on function] |
| <code>Parser</code> | int <code>NumOfErrors()</code> | | [member on function] |
| <code>Parser</code> | uint <code>LineNumber(char * pos, char * & fname, int & fname_len)</code> | | [member on function] |
| <code>Parser</code> | bool <code>rProgram(Ptree * &)</code> | | [member on function] |
| DeclKind | | | [enum] |
| | kDeclarator | | [enumerator] |
| | kArgDeclarator | | [enumerator] |
| | kCastDeclarator | | [enumerator] |

| | |
|---|----------------------|
| TemplateDeclKind | [enum] |
| tdk_unknown | [enumerator] |
| tdk_decl | [enumerator] |
| tdk_instantiation | [enumerator] |
| tdk_specialization | [enumerator] |
| num_tdks | [enumerator] |
| Parser bool SyntaxError() | [member on function] |
| Parser void ShowMessageHead(char *) | [member on function] |
| Parser bool rDefinition(Ptree * &) | [member on function] |
| Parser bool rNullDeclaration(Ptree * &) | [member on function] |
| Parser bool rTypedef(Ptree * &) | [member on function] |
| Parser bool rTypeSpecifier(Ptree * & , bool , Encoding &) | [member on function] |
| Parser bool isTypeSpecifier() | [member on function] |
| Parser bool rMetaclassDecl(Ptree * &) | [member on function] |
| Parser bool rMetaArguments(Ptree * &) | [member on function] |
| Parser bool rLinkageSpec(Ptree * &) | [member on function] |
| Parser bool rNamespaceSpec(Ptree * &) | [member on function] |
| Parser bool rUsing(Ptree * &) | [member on function] |
| Parser bool rLinkageBody(Ptree * &) | [member on function] |
| Parser bool rTemplateDecl(Ptree * &) | [member on function] |

```

Parser bool rTemplateDecl2(           [member on function]
    Ptree * & , TemplateDeclKind & kind)

Parser bool rTempArgList( Ptree       [member on function]
    * & )

Parser bool rTempArgDeclaration(     [member on function]
    Ptree * & )

Parser bool rExternTemplateDecl(     [member on function]
    Ptree * & )

Parser bool rDeclaration( Ptree      [member on function]
    * & )

Parser bool                               [member on function]
    rIntegralDeclaration( Ptree * & , Encoding & ,
    Ptree * , Ptree * , Ptree * )

Parser bool rConstDeclaration(       [member on function]
    Ptree * & , Encoding & , Ptree * , Ptree * )

Parser bool rOtherDeclaration(       [member on function]
    Ptree * & , Encoding & , Ptree * , Ptree * ,
    Ptree * )

Parser bool isConstructorDecl()      [member on function]

Parser bool isPtrToMember( int )     [member on function]

Parser bool optMemberSpec( Ptree     [member on function]
    * & )

Parser bool optStorageSpec(         [member on function]
    Ptree * & )

Parser bool optCvQualify( Ptree     [member on function]
    * & )

Parser bool                               [member on function]
    optIntegralTypeOrClassSpec( Ptree * & ,
    Encoding & )

Parser bool rConstructorDecl(       [member on function]
    Ptree * & , Encoding & )

Parser bool optThrowDecl( Ptree     [member on function]
    * & )

```

```
Parser bool rDeclarators( Ptree      [member on function]
    * & , Encoding & , bool , bool )

Parser bool rDeclaratorWithInit(      [member on function]
    Ptree * & , Encoding & , bool , bool )

Parser bool rDeclarator( Ptree *      [member on function]
    & , DeclKind , bool , Encoding & , Encoding & ,
    bool , bool )

Parser bool rDeclarator2( Ptree      [member on function]
    * & , DeclKind , bool , Encoding & , Encoding & ,
    bool , bool , Ptree * * )

Parser bool optPtrOperator(           [member on function]
    Ptree * & , Encoding & )

Parser bool rMemberInitializers(     [member on function]
    Ptree * & )

Parser bool rMemberInit( Ptree *     [member on function]
    & )

Parser bool rName( Ptree * & ,       [member on function]
    Encoding & )

Parser bool rOperatorName( Ptree     [member on function]
    * & , Encoding & )

Parser bool rCastOperatorName(       [member on function]
    Ptree * & , Encoding & )

Parser bool rPtrToMember( Ptree     [member on function]
    * & , Encoding & )

Parser bool rTemplateArgs( Ptree     [member on function]
    * & , Encoding & )

Parser bool rArgDeclListOrInit(      [member on function]
    Ptree * & , bool & , Encoding & , bool )

Parser bool rArgDeclList( Ptree     [member on function]
    * & , Encoding & )

Parser bool rArgDeclaration(         [member on function]
    Ptree * & , Encoding & )
```

| | |
|--|----------------------|
| Parser bool rFunctionArguments(Ptree * &) | [member on function] |
| Parser bool rInitializeExpr(Ptree * &) | [member on function] |
| Parser bool rEnumSpec(Ptree * & , Encoding &) | [member on function] |
| Parser bool rEnumBody(Ptree * &) | [member on function] |
| Parser bool rClassSpec(Ptree * & , Encoding &) | [member on function] |
| Parser bool rBaseSpecifiers(Ptree * &) | [member on function] |
| Parser bool rClassBody(Ptree * &) | [member on function] |
| Parser bool rClassMember(Ptree * &) | [member on function] |
| Parser bool rAccessDecl(Ptree * &) | [member on function] |
| Parser bool rUserAccessSpec(Ptree * &) | [member on function] |
| Parser bool rCommaExpression(Ptree * &) | [member on function] |
| Parser bool rExpression(Ptree * &) | [member on function] |
| Parser bool rConditionalExpr(Ptree * &) | [member on function] |
| Parser bool rLogicalOrExpr(Ptree * & , bool) | [member on function] |
| Parser bool rLogicalAndExpr(Ptree * & , bool) | [member on function] |
| Parser bool rInclusiveOrExpr(Ptree * & , bool) | [member on function] |

| | |
|---|----------------------|
| Parser bool rExclusiveOrExpr(Ptree * & , bool) | [member on function] |
| Parser bool rAndExpr(Ptree * & , bool) | [member on function] |
| Parser bool rEqualityExpr(Ptree * & , bool) | [member on function] |
| Parser bool rRelationalExpr(Ptree * & , bool) | [member on function] |
| Parser bool rShiftExpr(Ptree * &) | [member on function] |
| Parser bool rAdditiveExpr(Ptree * &) | [member on function] |
| Parser bool rMultiplyExpr(Ptree * &) | [member on function] |
| Parser bool rPmExpr(Ptree * &) | [member on function] |
| Parser bool rCastExpr(Ptree * &) | [member on function] |
| Parser bool rTypeName(Ptree * &) | [member on function] |
| Parser bool rTypeName(Ptree * & , Encoding &) | [member on function] |
| Parser bool rUnaryExpr(Ptree * &) | [member on function] |
| Parser bool rThrowExpr(Ptree * &) | [member on function] |
| Parser bool rSizeofExpr(Ptree * &) | [member on function] |
| Parser bool rTypeidExpr(Ptree * &) | [member on function] |
| Parser bool isAllocateExpr(int) | [member on function] |
| Parser bool rAllocateExpr(Ptree * &) | [member on function] |

| | |
|--|----------------------|
| Parser bool rUserdefKeyword(Ptree * &) | [member on function] |
| Parser bool rAllocateType(Ptree * &) | [member on function] |
| Parser bool rNewDeclarator(Ptree * & , Encoding &) | [member on function] |
| Parser bool rAllocateInitializer(Ptree * &) | [member on function] |
| Parser bool rPostfixExpr(Ptree * &) | [member on function] |
| Parser bool rPrimaryExpr(Ptree * &) | [member on function] |
| Parser bool rUserdefStatement(Ptree * &) | [member on function] |
| Parser bool rVarName(Ptree * &) | [member on function] |
| Parser bool rVarNameCore(Ptree * & , Encoding &) | [member on function] |
| Parser bool isTemplateArgs() | [member on function] |
| Parser bool rCondition(Ptree * &) | [member on function] |
| Parser bool rFunctionBody(Ptree * &) | [member on function] |
| Parser bool rCompoundStatement(Ptree * &) | [member on function] |
| Parser bool rStatement(Ptree * &) | [member on function] |
| Parser bool rIfStatement(Ptree * &) | [member on function] |
| Parser bool rSwitchStatement(Ptree * &) | [member on function] |
| Parser bool rWhileStatement(Ptree * &) | [member on function] |

| | |
|---|----------------------|
| Parser bool rDoStatement(Ptree * &) | [member on function] |
| Parser bool rForStatement(Ptree * &) | [member on function] |
| Parser bool rTryStatement(Ptree * &) | [member on function] |
| Parser bool rExprStatement(Ptree * &) | [member on function] |
| Parser bool rDeclarationStatement(Ptree * &) | [member on function] |
| Parser bool rIntegralDeclStatement(Ptree * & , Encoding & , Ptree * , Ptree * , Ptree *) | [member on function] |
| Parser bool rOtherDeclStatement(Ptree * & , Encoding & , Ptree * , Ptree *) | [member on function] |
| Parser bool MaybeTypeNameOrClassTemplate(Token &) | [member on function] |
| Parser void SkipTo(int token) | [member on function] |
| Parser bool moreVarName() | [member on function] |
| Lex * lex | [data member] |
| int nerrors | [data member] |
| Ptree * comments | [data member] |
| Ptree | [class] |
| parents: parent class | |
| Ptree bool IsLeaf() | [member on function] |
| Ptree bool Eq(char) | [member on function] |
| Ptree bool Eq(char *) | [member on function] |
| Ptree bool Eq(char * , int) | [member on function] |
| Ptree bool Eq(Ptree * p) | [member on function] |

| | |
|---|----------------------|
| <code>Ptree void Display()</code> | [member on function] |
| <code>Ptree void Display2(std::ostream &)</code> | [member on function] |
| <code>Ptree void Print(std::ostream & , int , int)</code> | [member on function] |
| <code>Ptree int Write(std::ostream &)</code> | [member on function] |
| <code>Ptree int Write(std::ostream & , int)</code> | [member on function] |
| <code>Ptree void PrintIndent(std::ostream & , int)</code> | [member on function] |
| <code>Ptree char * ToString()</code> | [member on function] |
| <code>Ptree void WritePS(ProgramString &)</code> | [member on function] |
| <code>Ptree char * GetPosition()</code> | [member on function] |
| <code>Ptree int GetLength()</code> | [member on function] |
| <code>Ptree Ptree * Car()</code> | [member on function] |
| <code>Ptree Ptree * Cdr()</code> | [member on function] |
| <code>Ptree Ptree * Cadr()</code> | [member on function] |
| <code>Ptree Ptree * Caddr()</code> | [member on function] |
| <code>Ptree Ptree * Ca_ar()</code> | [member on function] |
| <code>Ptree void SetCar(Ptree * p)</code> | [member on function] |
| <code>Ptree void SetCdr(Ptree * p)</code> | [member on function] |
| <code>Ptree char * LeftMost()</code> | [member on function] |
| <code>Ptree char * RightMost()</code> | [member on function] |
| <code>Ptree int What()</code> | [member on function] |
| <code>Ptree bool IsA(int)</code> | [member on function] |
| <code>Ptree bool IsA(int , int)</code> | [member on function] |

| | |
|---|----------------------|
| Ptree bool IsA(int , int , int) | [member on function] |
| Ptree Ptree * Translate(Walker *) | [member on function] |
| Ptree void Typeof(Walker * , TypeInfo &) | [member on function] |
| Ptree char * GetEncodedType() | [member on function] |
| Ptree char * GetEncodedName() | [member on function] |
| Ptree Ptree * Last() | [member on function] |
| Ptree Ptree * First() | [member on function] |
| Ptree Ptree * Rest() | [member on function] |
| Ptree Ptree * Second() | [member on function] |
| Ptree Ptree * Third() | [member on function] |
| Ptree Ptree * Nth(int n) | [member on function] |
| Ptree int Length() | [member on function] |
| Ptree Ptree * ListTail(int n) | [member on function] |
| Ptree bool Eq(Ptree * , char) | [member on function] |
| Ptree bool Eq(Ptree * , char *) | [member on function] |
| Ptree bool Eq(Ptree * , char * , int) | [member on function] |
| Ptree bool Eq(Ptree * , Ptree *) | [member on function] |
| Ptree bool Equiv(Ptree * , Ptree *) | [member on function] |
| Ptree bool Equal(Ptree * , Ptree *) | [member on function] |
| Ptree Ptree * Last(Ptree *) | [member on function] |
| Ptree Ptree * First(Ptree *) | [member on function] |
| Ptree Ptree * Rest(Ptree *) | [member on function] |

```

Ptree Ptree * Second( Ptree * )      [member on function]
Ptree Ptree * Third( Ptree * )      [member on function]
Ptree Ptree * Nth( Ptree * , int    [member on function]
    )
Ptree int Length( Ptree * )          [member on function]
Ptree Ptree * ListTail( Ptree *     [member on function]
    , int )
Ptree Ptree * Cons( Ptree * ,       [member on function]
    Ptree * )
Ptree Ptree * List()                 [member on function]
Ptree Ptree * List( Ptree * )        [member on function]
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * , Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * , Ptree * )
Ptree Ptree * List( Ptree * ,       [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * , Ptree * , Ptree * )
Ptree Ptree * CopyList( Ptree *     [member on function]
    )
Ptree Ptree * Append( Ptree * ,     [member on function]
    Ptree * )

```

```

Ptree Ptree * ReplaceAll( Ptree [member on function]
    * , Ptree * , Ptree * )

Ptree Ptree * Subst( Ptree * , [member on function]
    Ptree * , Ptree * )

Ptree Ptree * Subst( Ptree * , [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * )

Ptree Ptree * Subst( Ptree * , [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * ,
    Ptree * )

Ptree Ptree * ShallowSubst( [member on function]
    Ptree * , Ptree * , Ptree * )

Ptree Ptree * ShallowSubst( [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * )

Ptree Ptree * ShallowSubst( [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * ,
    Ptree * , Ptree * )

Ptree Ptree * ShallowSubst( [member on function]
    Ptree * , Ptree * , Ptree * , Ptree * , Ptree * ,
    Ptree * , Ptree * , Ptree * , Ptree * )

Ptree Ptree * SubstSublist( [member on function]
    Ptree * , Ptree * , Ptree * )

Ptree Ptree * Snoc( Ptree * , [member on function]
    Ptree * )

Ptree Ptree * Nconc( Ptree * , [member on function]
    Ptree * )

Ptree Ptree * Nconc( Ptree * , [member on function]
    Ptree * , Ptree * )

Ptree bool Match( Ptree * , char [member on function]
    * , ... )

Ptree Ptree * Make( const char * [member on function]
    pat, ... )

Ptree Ptree * MakeStatement( [member on function]
    const char * pat, ... )

```

| | |
|---|----------------------|
| <code>Ptree Ptree * GenSym()</code> | [member on function] |
| <code>Ptree Ptree * qMake(char *)</code> | [member on function] |
| <code>Ptree Ptree * qMakeStatement(char *)</code> | [member on function] |
| <code>Ptree bool Reify(unsigned int &)</code> | [member on function] |
| <code>Ptree bool Reify(char * &)</code> | [member on function] |
| <code>Ptree char * IntegerToString(sint , int &)</code> | [member on function] |
| <code>Ptree char * MatchPat(Ptree * , char *)</code> | [member on function] |
| <code>Ptree char * MatchList(Ptree * , char *)</code> | [member on function] |
| <code>Ptree char * MatchWord(Ptree * , char *)</code> | [member on function] |
| <code>bool show_encoded</code> | [data member] |
| <code>'0027</code> | [union] |
| <code>'0028</code> | [struct] |
| <code>Ptree * child</code> | [data member] |
| <code>Ptree * next</code> | [data member] |
| <code>'0028 nonleaf</code> | [data member] |
| <code>'0029</code> | [struct] |
| <code>char * position</code> | [data member] |
| <code>int length</code> | [data member] |
| <code>'0029 leaf</code> | [data member] |
| <code>'0027 data</code> | [data member] |
| <code>std::ostream & operator<<(std::ostream & s, Ptree * p)</code> | [function] |

PtreeIter [class]

parents: parent class

PtreeIter PtreeIter(Ptree * p) [member on function]

PtreeIter Ptree * operator>() [member on function]

PtreeIter Ptree * Pop() [member on function]

PtreeIter bool Next(Ptree * &) [member on function]

PtreeIter void Reset(Ptree * p) [member on function]

PtreeIter Ptree * operator*() [member on function]

PtreeIter Ptree * operator++() [member on function]

PtreeIter Ptree * operator++(
int) [member on function]

PtreeIter Ptree * This() [member on function]

PtreeIter bool Empty() [member on function]

Ptree * ptree [data member]

PtreeArray [class]

parents: parent class

PtreeArray PtreeArray(int) [member on function]

PtreeArray uint Number() [member on function]

PtreeArray Ptree * &
operator[] (uint index) [member on function]PtreeArray Ptree * & Ref(uint
index) [member on function]

PtreeArray void Append(Ptree *) [member on function]

PtreeArray void Clear() [member on function]

PtreeArray Ptree * All() [member on function]

uint num [data member]

| | |
|--|----------------------|
| <code>uint size</code> | [data member] |
| <code>Ptree * * array</code> | [data member] |
| <code>[] Ptree * default_buf</code> | [data member] |
| PtreeHead | [class] |
| parents: parent class | |
| <code>PtreeHead PtreeHead()</code> | [member on function] |
| <code>PtreeHead Ptree * (Ptree*)()</code> | [member on function] |
| <code>PtreeHead PtreeHead & operator+(Ptree *)</code> | [member on function] |
| <code>PtreeHead PtreeHead & operator+(const char *)</code> | [member on function] |
| <code>PtreeHead PtreeHead & operator+(char *)</code> | [member on function] |
| <code>PtreeHead PtreeHead & operator+(char)</code> | [member on function] |
| <code>PtreeHead PtreeHead & operator+(int)</code> | [member on function] |
| <code>PtreeHead Ptree * Append(Ptree * , Ptree *)</code> | [member on function] |
| <code>PtreeHead Ptree * Append(Ptree * , char * , int)</code> | [member on function] |
| <code>Ptree * ptree</code> | [data member] |
| Leaf | [class] |
| parents: parent class | |
| <code>Leaf Leaf(char * , int)</code> | [member on function] |
| <code>Leaf Leaf(Token &)</code> | [member on function] |
| <code>Leaf bool IsLeaf()</code> | [member on function] |
| <code>Leaf void Print(std::ostream & , int , int)</code> | [member on function] |

Leaf **int** Write(std::ostream & ,
int) [member on function]

Leaf **void** WritePS(ProgramString
&) [member on function]

CommentedLeaf [class]

parents: parent class

CommentedLeaf CommentedLeaf(
Token & tk, Ptree * c) [member on function]

CommentedLeaf CommentedLeaf(
char * p, int l, Ptree * c) [member on function]

CommentedLeaf **Ptree ***
GetComments() [member on function]

CommentedLeaf **void** SetComments(
Ptree * c) [member on function]

Ptree * comments [data member]

LeafName [class]

parents: parent class

LeafName LeafName(Token &) [member on function]

LeafName **Ptree *** Translate(
Walker *) [member on function]

LeafName **void** Typeof(Walker * ,
TypeInfo &) [member on function]

DupLeaf [class]

parents: parent class

DupLeaf DupLeaf(char * , int) [member on function]

DupLeaf DupLeaf(char * , int ,
char * , int) [member on function]

DupLeaf **void** Print(std::ostream
& , int , int) [member on function]

LeafReserved [class]

parents: parent class

| | | |
|--------------------|--------------------------------------|----------------------|
| LeafReserved | LeafReserved(Token & t) | [member on function] |
| LeafReserved | LeafReserved(char * str, int len) | [member on function] |
| LeafThis | | [class] |
| | parents: parent class | |
| LeafThis | LeafThis(Token & t) | [member on function] |
| LeafThis | int What() | [member on function] |
| LeafThis | Ptree * Translate(Walker *) | [member on function] |
| LeafThis | void Typeof(Walker * , TypeInfo &) | [member on function] |
| LeafAUTO | | [class] |
| | parents: parent class | |
| LeafAUTO | LeafAUTO(Token & t) | [member on function] |
| LeafAUTO | LeafAUTO(char * str, int len) | [member on function] |
| LeafAUTO | int What() | [member on function] |
| LeafBOOLEAN | | [class] |
| | parents: parent class | |
| LeafBOOLEAN | LeafBOOLEAN(Token & t) | [member on function] |
| LeafBOOLEAN | LeafBOOLEAN(char * str, int len) | [member on function] |
| LeafBOOLEAN | int What() | [member on function] |
| LeafCHAR | | [class] |
| | parents: parent class | |
| LeafCHAR | LeafCHAR(Token & t) | [member on function] |
| LeafCHAR | LeafCHAR(char * str, int len) | [member on function] |

| | |
|--|----------------------|
| <code>LeafCHAR int What()</code> | [member on function] |
| LeafCONST | [class] |
| parents: parent class | |
| <code>LeafCONST LeafCONST(Token & t)</code> | [member on function] |
| <code>LeafCONST LeafCONST(char * str, int len)</code> | [member on function] |
| <code>LeafCONST int What()</code> | [member on function] |
| LeafDOUBLE | [class] |
| parents: parent class | |
| <code>LeafDOUBLE LeafDOUBLE(Token & t)</code> | [member on function] |
| <code>LeafDOUBLE LeafDOUBLE(char * str, int len)</code> | [member on function] |
| <code>LeafDOUBLE int What()</code> | [member on function] |
| LeafEXTERN | [class] |
| parents: parent class | |
| <code>LeafEXTERN LeafEXTERN(Token & t)</code> | [member on function] |
| <code>LeafEXTERN LeafEXTERN(char * str, int len)</code> | [member on function] |
| <code>LeafEXTERN int What()</code> | [member on function] |
| LeafFLOAT | [class] |
| parents: parent class | |
| <code>LeafFLOAT LeafFLOAT(Token & t)</code> | [member on function] |
| <code>LeafFLOAT LeafFLOAT(char * str, int len)</code> | [member on function] |
| <code>LeafFLOAT int What()</code> | [member on function] |
| LeafFRIEND | [class] |
| parents: parent class | |

| | | |
|--------------------|---------------------------------------|----------------------|
| LeafFRIEND | LeafFRIEND(Token & t) | [member on function] |
| LeafFRIEND | LeafFRIEND(char * str, int len) | [member on function] |
| LeafFRIEND | int What() | [member on function] |
| LeafINLINE | | [class] |
| | parents: parent class | |
| LeafINLINE | LeafINLINE(Token & t) | [member on function] |
| LeafINLINE | LeafINLINE(char * str, int len) | [member on function] |
| LeafINLINE | int What() | [member on function] |
| LeafINT | | [class] |
| | parents: parent class | |
| LeafINT | LeafINT(Token & t) | [member on function] |
| LeafINT | LeafINT(char * str, int len) | [member on function] |
| LeafINT | int What() | [member on function] |
| LeafLONG | | [class] |
| | parents: parent class | |
| LeafLONG | LeafLONG(Token & t) | [member on function] |
| LeafLONG | LeafLONG(char * str, int len) | [member on function] |
| LeafLONG | int What() | [member on function] |
| LeafMUTABLE | | [class] |
| | parents: parent class | |
| LeafMUTABLE | LeafMUTABLE(Token & t) | [member on function] |
| LeafMUTABLE | LeafMUTABLE(char * str, int len) | [member on function] |

| | |
|---|----------------------|
| <code>LeafMUTABLE int What()</code> | [member on function] |
| LeafNAMESPACE | [class] |
| parents: parent class | |
| <code>LeafNAMESPACE LeafNAMESPACE(Token & t)</code> | [member on function] |
| <code>LeafNAMESPACE LeafNAMESPACE(char * str, int len)</code> | [member on function] |
| <code>LeafNAMESPACE int What()</code> | [member on function] |
| LeafPRIVATE | [class] |
| parents: parent class | |
| <code>LeafPRIVATE LeafPRIVATE(Token & t)</code> | [member on function] |
| <code>LeafPRIVATE LeafPRIVATE(char * str, int len)</code> | [member on function] |
| <code>LeafPRIVATE int What()</code> | [member on function] |
| LeafPROTECTED | [class] |
| parents: parent class | |
| <code>LeafPROTECTED LeafPROTECTED(Token & t)</code> | [member on function] |
| <code>LeafPROTECTED LeafPROTECTED(char * str, int len)</code> | [member on function] |
| <code>LeafPROTECTED int What()</code> | [member on function] |
| LeafPUBLIC | [class] |
| parents: parent class | |
| <code>LeafPUBLIC LeafPUBLIC(Token & t)</code> | [member on function] |
| <code>LeafPUBLIC LeafPUBLIC(char * str, int len)</code> | [member on function] |
| <code>LeafPUBLIC int What()</code> | [member on function] |
| LeafREGISTER | [class] |
| parents: parent class | |

| | | |
|---------------------|-------------------------------------|----------------------|
| LeafREGISTER | LeafREGISTER(Token & t) | [member on function] |
| LeafREGISTER | LeafREGISTER(char * str, int len) | [member on function] |
| LeafREGISTER | int What() | [member on function] |
| LeafSHORT | | [class] |
| | parents: parent class | |
| LeafSHORT | LeafSHORT(Token & t) | [member on function] |
| LeafSHORT | LeafSHORT(char * str, int len) | [member on function] |
| LeafSHORT | int What() | [member on function] |
| LeafSIGNED | | [class] |
| | parents: parent class | |
| LeafSIGNED | LeafSIGNED(Token & t) | [member on function] |
| LeafSIGNED | LeafSIGNED(char * str, int len) | [member on function] |
| LeafSIGNED | int What() | [member on function] |
| LeafSTATIC | | [class] |
| | parents: parent class | |
| LeafSTATIC | LeafSTATIC(Token & t) | [member on function] |
| LeafSTATIC | LeafSTATIC(char * str, int len) | [member on function] |
| LeafSTATIC | int What() | [member on function] |
| LeafUNSIGNED | | [class] |
| | parents: parent class | |
| LeafUNSIGNED | LeafUNSIGNED(Token & t) | [member on function] |
| LeafUNSIGNED | LeafUNSIGNED(char * str, int len) | [member on function] |

| | |
|--|----------------------|
| <code>LeafUNSIGNED int What()</code> | [member on function] |
| LeafUSING | [class] |
| parents: parent class | |
| <code>LeafUSING LeafUSING(Token & t)</code> | [member on function] |
| <code>LeafUSING LeafUSING(char * str, int len)</code> | [member on function] |
| <code>LeafUSING int What()</code> | [member on function] |
| LeafVIRTUAL | [class] |
| parents: parent class | |
| <code>LeafVIRTUAL LeafVIRTUAL(Token & t)</code> | [member on function] |
| <code>LeafVIRTUAL LeafVIRTUAL(char * str, int len)</code> | [member on function] |
| <code>LeafVIRTUAL int What()</code> | [member on function] |
| LeafVOID | [class] |
| parents: parent class | |
| <code>LeafVOID LeafVOID(Token & t)</code> | [member on function] |
| <code>LeafVOID LeafVOID(char * str, int len)</code> | [member on function] |
| <code>LeafVOID int What()</code> | [member on function] |
| LeafVOLATILE | [class] |
| parents: parent class | |
| <code>LeafVOLATILE LeafVOLATILE(Token & t)</code> | [member on function] |
| <code>LeafVOLATILE LeafVOLATILE(char * str, int len)</code> | [member on function] |
| <code>LeafVOLATILE int What()</code> | [member on function] |
| LeafUserKeyword2 | [class] |
| parents: parent class | |

| | |
|---|----------------------|
| LeafUserKeyword2 | [member on function] |
| LeafUserKeyword2(Token & t) | |
| LeafUserKeyword2 | [member on function] |
| LeafUserKeyword2(char * str, int len) | |
| LeafUserKeyword2 int What() | [member on function] |
| NonLeaf | [class] |
| parents: parent class | |
| NonLeaf NonLeaf(Ptree * , Ptree *) | [member on function] |
| NonLeaf bool IsLeaf() | [member on function] |
| NonLeaf void Print(std::ostream & , int , int) | [member on function] |
| NonLeaf int Write(std::ostream & , int) | [member on function] |
| NonLeaf void PrintWithEncodeds(std::ostream & , int , int) | [member on function] |
| NonLeaf void WritePS(ProgramString &) | [member on function] |
| NonLeaf bool TooDeep(std::ostream & , int) | [member on function] |
| PtreeBrace | [class] |
| parents: parent class | |
| PtreeBrace PtreeBrace(Ptree * p, Ptree * q) | [member on function] |
| PtreeBrace PtreeBrace(Ptree * ob, Ptree * body, Ptree * cb) | [member on function] |
| PtreeBrace void Print(std::ostream & , int , int) | [member on function] |
| PtreeBrace int Write(std::ostream & , int) | [member on function] |
| PtreeBrace Ptree * Translate(Walker *) | [member on function] |

PtreeBlock [class]

parents: parent class

PtreeBlock PtreeBlock(Ptree * [member on function]
 p, Ptree * *q*)

PtreeBlock PtreeBlock(Ptree * [member on function]
 ob, Ptree * *bdy*, Ptree * *cb*)

PtreeBlock **Ptree** * Translate([member on function]
 Walker *)

PtreeClassBody [class]

parents: parent class

PtreeClassBody PtreeClassBody([member on function]
 Ptree * *p*, Ptree * *q*)

PtreeClassBody PtreeClassBody([member on function]
 Ptree * *ob*, Ptree * *bdy*, Ptree * *cb*)

PtreeClassBody **Ptree** * [member on function]
 Translate(Walker *)

PtreeTypedef [class]

parents: parent class

PtreeTypedef PtreeTypedef([member on function]
 Ptree * *p*)

PtreeTypedef PtreeTypedef([member on function]
 Ptree * *p*, Ptree * *q*)

PtreeTypedef **int** What() [member on function]

PtreeTypedef **Ptree** * Translate([member on function]
 Walker *)

PtreeTemplateDecl [class]

parents: parent class

PtreeTemplateDecl [member on function]
 PtreeTemplateDecl(Ptree * *p*, Ptree * *q*)

PtreeTemplateDecl [member on function]
 PtreeTemplateDecl(Ptree * *p*)

| | | |
|--|--|----------------------|
| <code>PtreeTemplateDecl</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeTemplateDecl</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeTemplateInstantiation</code> | | [class] |
| parents: parent class | | |
| <code>PtreeTemplateInstantiation</code> | <code>PtreeTemplateInstantiation(Ptree * p)</code> | [member on function] |
| <code>PtreeTemplateInstantiation</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeTemplateInstantiation</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeExternTemplate</code> | | [class] |
| parents: parent class | | |
| <code>PtreeExternTemplate</code> | <code>PtreeExternTemplate(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeExternTemplate</code> | <code>PtreeExternTemplate(Ptree * p)</code> | [member on function] |
| <code>PtreeExternTemplate</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeExternTemplate</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeMetaclassDecl</code> | | [class] |
| parents: parent class | | |
| <code>PtreeMetaclassDecl</code> | <code>PtreeMetaclassDecl(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeMetaclassDecl</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeMetaclassDecl</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeLinkageSpec</code> | | [class] |
| parents: parent class | | |
| <code>PtreeLinkageSpec</code> | <code>PtreeLinkageSpec(Ptree * p, Ptree * q)</code> | [member on function] |

| | |
|---|----------------------|
| <code>PtreeLinkageSpec int What()</code> | [member on function] |
| <code>PtreeLinkageSpec Ptree * Translate(Walker *)</code> | [member on function] |
| PtreeNamespaceSpec | [class] |
| parents: parent class | |
| <code>PtreeNamespaceSpec PtreeNamespaceSpec(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeNamespaceSpec int What()</code> | [member on function] |
| <code>PtreeNamespaceSpec Ptree * Translate(Walker *)</code> | [member on function] |
| <code>PtreeNamespaceSpec Ptree * GetComments()</code> | [member on function] |
| <code>PtreeNamespaceSpec void SetComments(Ptree * c)</code> | [member on function] |
| <code>Ptree * comments</code> | [data member] |
| PtreeUsing | [class] |
| parents: parent class | |
| <code>PtreeUsing PtreeUsing(Ptree * p)</code> | [member on function] |
| <code>PtreeUsing int What()</code> | [member on function] |
| <code>PtreeUsing Ptree * Translate(Walker *)</code> | [member on function] |
| PtreeDeclaration | [class] |
| parents: parent class | |
| <code>PtreeDeclaration PtreeDeclaration(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeDeclaration int What()</code> | [member on function] |
| <code>PtreeDeclaration Ptree * Translate(Walker *)</code> | [member on function] |
| <code>PtreeDeclaration Ptree * GetComments()</code> | [member on function] |

```

PtreeDeclaration void                                [member on function]
    SetComments( Ptree * c )

Ptree * comments                                     [data member]

PtreeDeclarator                                     [class]
parents: parent class

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( Ptree * , Encoding & ,
    Encoding & , Ptree * )

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( Encoding & , Encoding & ,
    Ptree * )

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( Ptree * , Ptree * , Encoding &
    , Encoding & , Ptree * )

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( Ptree * , Encoding & )

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( Encoding & )

PtreeDeclarator                                       [member on function]
    PtreeDeclarator( PtreeDeclarator * , Ptree * ,
    Ptree * )

PtreeDeclarator int What()                            [member on function]

PtreeDeclarator char *                               [member on function]
    GetEncodedType()

PtreeDeclarator char *                               [member on function]
    GetEncodedName()

PtreeDeclarator void                                 [member on function]
    SetEncodedType( char * t )

PtreeDeclarator Ptree * Name()                       [member on function]

PtreeDeclarator void Print(                          [member on function]
    std::ostream & , int , int )

PtreeDeclarator Ptree *                             [member on function]
    GetComments()

```

| | | |
|------------------------------|--|----------------------|
| PtreeDeclarator | void | [member on function] |
| | SetComments(Ptree * c) | |
| char * type | | [data member] |
| char * name | | [data member] |
| Ptree * declared_name | | [data member] |
| Ptree * comments | | [data member] |
| PtreeName | | [class] |
| parents: parent class | | |
| PtreeName | PtreeName (Ptree * , Encoding &) | [member on function] |
| PtreeName | int What() | [member on function] |
| PtreeName | char * GetEncodedName() | [member on function] |
| PtreeName | void Print(std::ostream & , int , int) | [member on function] |
| PtreeName | Ptree * Translate(Walker *) | [member on function] |
| PtreeName | void Typeof(Walker * , TypeInfo &) | [member on function] |
| char * name | | [data member] |
| PtreeFstyleCastExpr | | [class] |
| parents: parent class | | |
| PtreeFstyleCastExpr | PtreeFstyleCastExpr (Encoding & , Ptree * , Ptree *) | [member on function] |
| PtreeFstyleCastExpr | PtreeFstyleCastExpr (char * , Ptree * , Ptree *) | [member on function] |
| PtreeFstyleCastExpr | int What() | [member on function] |
| PtreeFstyleCastExpr | char * GetEncodedType() | [member on function] |

```

PtreeFstyleCastExpr void Print(      [member on function]
    std::ostream & , int , int )

PtreeFstyleCastExpr Ptree *         [member on function]
    Translate( Walker * )

PtreeFstyleCastExpr void            [member on function]
    Typeof( Walker * , TypeInfo & )

char * type                          [data member]

PtreeClassSpec                      [class]
parents: parent class

PtreeClassSpec PtreeClassSpec(      [member on function]
    Ptree * , Ptree * , Ptree * )

PtreeClassSpec PtreeClassSpec(      [member on function]
    Ptree * , Ptree * , Ptree * , char * )

PtreeClassSpec int What()           [member on function]

PtreeClassSpec Ptree *             [member on function]
    Translate( Walker * )

PtreeClassSpec char *              [member on function]
    GetEncodedName()

PtreeClassSpec Ptree *             [member on function]
    GetComments()

char * encoded_name                 [data member]

Ptree * comments                    [data member]

PtreeEnumSpec                      [class]
parents: parent class

PtreeEnumSpec PtreeEnumSpec(        [member on function]
    Ptree * )

PtreeEnumSpec int What()           [member on function]

PtreeEnumSpec Ptree *             [member on function]
    Translate( Walker * )

PtreeEnumSpec char *              [member on function]
    GetEncodedName()

```



```

        char * encoded_name                                [data member]

PtreeAccessSpec                                        [class]
    parents: parent class

        PtreeAccessSpec                                  [member on function]
            PtreeAccessSpec( Ptree * , Ptree * )

        PtreeAccessSpec int What()                      [member on function]

        PtreeAccessSpec Ptree *                         [member on function]
            Translate( Walker * )

PtreeAccessDecl                                       [class]
    parents: parent class

        PtreeAccessDecl                                  [member on function]
            PtreeAccessDecl( Ptree * , Ptree * )

        PtreeAccessDecl int What()                    [member on function]

        PtreeAccessDecl Ptree *                        [member on function]
            Translate( Walker * )

PtreeUserAccessSpec                                    [class]
    parents: parent class

        PtreeUserAccessSpec                              [member on function]
            PtreeUserAccessSpec( Ptree * , Ptree * )

        PtreeUserAccessSpec int What()                 [member on function]

        PtreeUserAccessSpec Ptree *                   [member on function]
            Translate( Walker * )

PtreeUserdefKeyword                                   [class]
    parents: parent class

        PtreeUserdefKeyword                              [member on function]
            PtreeUserdefKeyword( Ptree * , Ptree * )

        PtreeUserdefKeyword int What()                [member on function]

PtreeIfStatement                                     [class]
    parents: parent class

        PtreeIfStatement                                  [member on function]
            PtreeIfStatement( Ptree * p, Ptree * q )

```

```

PtreeIfStatement int What()           [member on function]
PtreeIfStatement Ptree *
    Translate( Walker * )             [member on function]

PtreeSwitchStatement                 [class]
parents: parent class

PtreeSwitchStatement                 [member on function]
    PtreeSwitchStatement( Ptree * p, Ptree * q)

PtreeSwitchStatement int What()      [member on function]
PtreeSwitchStatement Ptree *
    Translate( Walker * )             [member on function]

PtreeWhileStatement                 [class]
parents: parent class

PtreeWhileStatement                 [member on function]
    PtreeWhileStatement( Ptree * p, Ptree * q)

PtreeWhileStatement int What()       [member on function]
PtreeWhileStatement Ptree *
    Translate( Walker * )             [member on function]

PtreeDoStatement                   [class]
parents: parent class

PtreeDoStatement                    [member on function]
    PtreeDoStatement( Ptree * p, Ptree * q)

PtreeDoStatement int What()          [member on function]
PtreeDoStatement Ptree *
    Translate( Walker * )             [member on function]

PtreeForStatement                  [class]
parents: parent class

PtreeForStatement                    [member on function]
    PtreeForStatement( Ptree * p, Ptree * q)

PtreeForStatement int What()         [member on function]
PtreeForStatement Ptree *
    Translate( Walker * )             [member on function]

```

PtreeTryStatement [class]

parents: parent class

PtreeTryStatement [member on function]
 PtreeTryStatement(Ptree * p, Ptree * q)

PtreeTryStatement int What() [member on function]

PtreeTryStatement Ptree * [member on function]
 Translate(Walker *)

PtreeBreakStatement [class]

parents: parent class

PtreeBreakStatement [member on function]
 PtreeBreakStatement(Ptree * p, Ptree * q)

PtreeBreakStatement int What() [member on function]

PtreeBreakStatement Ptree * [member on function]
 Translate(Walker *)

PtreeContinueStatement [class]

parents: parent class

PtreeContinueStatement [member on function]
 PtreeContinueStatement(Ptree * p, Ptree * q)

PtreeContinueStatement int [member on function]
 What()

PtreeContinueStatement Ptree [member on function]
 * Translate(Walker *)

PtreeReturnStatement [class]

parents: parent class

PtreeReturnStatement [member on function]
 PtreeReturnStatement(Ptree * p, Ptree * q)

PtreeReturnStatement int What() [member on function]

PtreeReturnStatement Ptree * [member on function]
 Translate(Walker *)

PtreeGotoStatement [class]

parents: parent class

```

PtreeGotoStatement                                [member on function]
    PtreeGotoStatement( Ptree * p, Ptree * q)

PtreeGotoStatement int What()                    [member on function]

PtreeGotoStatement Ptree *                       [member on function]
    Translate( Walker * )

PtreeCaseStatement                                [class]
    parents: parent class

PtreeCaseStatement                                [member on function]
    PtreeCaseStatement( Ptree * p, Ptree * q)

PtreeCaseStatement int What()                    [member on function]

PtreeCaseStatement Ptree *                       [member on function]
    Translate( Walker * )

PtreeDefaultStatement                            [class]
    parents: parent class

PtreeDefaultStatement                            [member on function]
    PtreeDefaultStatement( Ptree * p, Ptree * q)

PtreeDefaultStatement int What()                 [member on function]

PtreeDefaultStatement Ptree *                   [member on function]
    Translate( Walker * )

PtreeLabelStatement                              [class]
    parents: parent class

PtreeLabelStatement                              [member on function]
    PtreeLabelStatement( Ptree * p, Ptree * q)

PtreeLabelStatement int What()                   [member on function]

PtreeLabelStatement Ptree *                     [member on function]
    Translate( Walker * )

PtreeExprStatement                                [class]
    parents: parent class

PtreeExprStatement                              [member on function]
    PtreeExprStatement( Ptree * p, Ptree * q)

```

| | | |
|---------------------------------|--|----------------------|
| <code>PtreeExprStatement</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeExprStatement</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| PtreeCommaExpr | | [class] |
| parents: parent class | | |
| <code>PtreeCommaExpr</code> | <code>PtreeCommaExpr(</code> <code>Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeCommaExpr</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeCommaExpr</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeCommaExpr</code> | <code>void</code> <code>Typeof(</code> <code>Walker * , TypeInfo &)</code> | [member on function] |
| PtreeAssignExpr | | [class] |
| parents: parent class | | |
| <code>PtreeAssignExpr</code> | <code>PtreeAssignExpr(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeAssignExpr</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeAssignExpr</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeAssignExpr</code> | <code>void</code> <code>Typeof(</code> <code>Walker * , TypeInfo &)</code> | [member on function] |
| PtreeCondExpr | | [class] |
| parents: parent class | | |
| <code>PtreeCondExpr</code> | <code>PtreeCondExpr(</code> <code>Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeCondExpr</code> | <code>int</code> <code>What()</code> | [member on function] |
| <code>PtreeCondExpr</code> | <code>Ptree *</code> <code>Translate(Walker *)</code> | [member on function] |
| <code>PtreeCondExpr</code> | <code>void</code> <code>Typeof(</code> <code>Walker * , TypeInfo &)</code> | [member on function] |

PtreeInfixExpr [class]

parents: parent class

PtreeInfixExpr PtreeInfixExpr([member on function]
 Ptree * p, Ptree * q)

PtreeInfixExpr int What() [member on function]

PtreeInfixExpr Ptree * [member on function]
 Translate(Walker *)

PtreeInfixExpr void Typeof([member on function]
 Walker * , TypeInfo &)

PtreePmExpr [class]

parents: parent class

PtreePmExpr PtreePmExpr(Ptree [member on function]
 * p, Ptree * q)

PtreePmExpr int What() [member on function]

PtreePmExpr Ptree * Translate([member on function]
 Walker *)

PtreePmExpr void Typeof(Walker [member on function]
 * , TypeInfo &)

PtreeCastExpr [class]

parents: parent class

PtreeCastExpr PtreeCastExpr([member on function]
 Ptree * p, Ptree * q)

PtreeCastExpr int What() [member on function]

PtreeCastExpr Ptree * [member on function]
 Translate(Walker *)

PtreeCastExpr void Typeof([member on function]
 Walker * , TypeInfo &)

PtreeUnaryExpr [class]

parents: parent class

PtreeUnaryExpr PtreeUnaryExpr([member on function]
 Ptree * p, Ptree * q)

| | |
|---|----------------------|
| <code>PtreeUnaryExpr int What()</code> | [member on function] |
| <code>PtreeUnaryExpr Ptree * Translate(Walker *)</code> | [member on function] |
| <code>PtreeUnaryExpr void Typeof(Walker * , TypeInfo &)</code> | [member on function] |
| PtreeThrowExpr | [class] |
| parents: parent class | |
| <code>PtreeThrowExpr PtreeThrowExpr(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeThrowExpr int What()</code> | [member on function] |
| <code>PtreeThrowExpr Ptree * Translate(Walker *)</code> | [member on function] |
| <code>PtreeThrowExpr void Typeof(Walker * , TypeInfo &)</code> | [member on function] |
| PtreeSizeofExpr | [class] |
| parents: parent class | |
| <code>PtreeSizeofExpr PtreeSizeofExpr(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeSizeofExpr int What()</code> | [member on function] |
| <code>PtreeSizeofExpr Ptree * Translate(Walker *)</code> | [member on function] |
| <code>PtreeSizeofExpr void Typeof(Walker * , TypeInfo &)</code> | [member on function] |
| PtreeTypeidExpr | [class] |
| parents: parent class | |
| <code>PtreeTypeidExpr PtreeTypeidExpr(Ptree * p, Ptree * q)</code> | [member on function] |
| <code>PtreeTypeidExpr int What()</code> | [member on function] |
| <code>PtreeTypeidExpr Ptree * Translate(Walker *)</code> | [member on function] |

```

PtreeTypeidExpr void Typeof(           [member on function]
    Walker * , TypeInfo & )

PtreeTypeofExpr                               [class]
parents: parent class

PtreeTypeofExpr           [member on function]
    PtreeTypeofExpr( Ptree * p, Ptree * q)

PtreeTypeofExpr int What()           [member on function]

PtreeTypeofExpr Ptree *           [member on function]
    Translate( Walker * )

PtreeTypeofExpr void Typeof(           [member on function]
    Walker * , TypeInfo & )

PtreeNewExpr                               [class]
parents: parent class

PtreeNewExpr PtreeNewExpr(           [member on function]
    Ptree * p, Ptree * q)

PtreeNewExpr int What()           [member on function]

PtreeNewExpr Ptree * Translate( [member on function]
    Walker * )

PtreeNewExpr void Typeof(           [member on function]
    Walker * , TypeInfo & )

PtreeDeleteExpr                               [class]
parents: parent class

PtreeDeleteExpr           [member on function]
    PtreeDeleteExpr( Ptree * p, Ptree * q)

PtreeDeleteExpr int What()           [member on function]

PtreeDeleteExpr Ptree *           [member on function]
    Translate( Walker * )

PtreeDeleteExpr void Typeof(           [member on function]
    Walker * , TypeInfo & )

PtreeArrayExpr                               [class]
parents: parent class

```


`PtreeArrayExpr PtreeArrayExpr(Ptree * p, Ptree * q)` [member on function]

`PtreeArrayExpr int What()` [member on function]

`PtreeArrayExpr Ptree * Translate(Walker *)` [member on function]

`PtreeArrayExpr void Typeof(Walker * , TypeInfo &)` [member on function]

PtreeFuncallExpr [class]

parents: parent class

`PtreeFuncallExpr PtreeFuncallExpr(Ptree * p, Ptree * q)` [member on function]

`PtreeFuncallExpr int What()` [member on function]

`PtreeFuncallExpr Ptree * Translate(Walker *)` [member on function]

`PtreeFuncallExpr void Typeof(Walker * , TypeInfo &)` [member on function]

PtreePostfixExpr [class]

parents: parent class

`PtreePostfixExpr PtreePostfixExpr(Ptree * p, Ptree * q)` [member on function]

`PtreePostfixExpr int What()` [member on function]

`PtreePostfixExpr Ptree * Translate(Walker *)` [member on function]

`PtreePostfixExpr void Typeof(Walker * , TypeInfo &)` [member on function]

PtreeUserStatementExpr [class]

parents: parent class

`PtreeUserStatementExpr PtreeUserStatementExpr(Ptree * p, Ptree * q)` [member on function]

`PtreeUserStatementExpr int What()` [member on function]

```

PtreeUserStatementExpr Ptree      [member on function]
    * Translate( Walker * )

PtreeUserStatementExpr void      [member on function]
    Typeof( Walker * , TypeInfo & )

PtreeDotMemberExpr                [class]
parents: parent class

PtreeDotMemberExpr      [member on function]
    PtreeDotMemberExpr( Ptree * p, Ptree * q )

PtreeDotMemberExpr int What()    [member on function]

PtreeDotMemberExpr Ptree *      [member on function]
    Translate( Walker * )

PtreeDotMemberExpr void Typeof(  [member on function]
    Walker * , TypeInfo & )

PtreeArrowMemberExpr            [class]
parents: parent class

PtreeArrowMemberExpr     [member on function]
    PtreeArrowMemberExpr( Ptree * p, Ptree * q )

PtreeArrowMemberExpr int What()  [member on function]

PtreeArrowMemberExpr Ptree *     [member on function]
    Translate( Walker * )

PtreeArrowMemberExpr void       [member on function]
    Typeof( Walker * , TypeInfo & )

PtreeParenExpr                 [class]
parents: parent class

PtreeParenExpr PtreeParenExpr(  [member on function]
    Ptree * p, Ptree * q )

PtreeParenExpr int What()       [member on function]

PtreeParenExpr Ptree *         [member on function]
    Translate( Walker * )

PtreeParenExpr void Typeof(     [member on function]
    Walker * , TypeInfo & )

```

PtreeStaticUserStatementExpr [class]

parents: parent class

PtreeStaticUserStatementExpr [member on function]
PtreeStaticUserStatementExpr(**Ptree** * *p*, **Ptree** * *q*)

PtreeStaticUserStatementExpr [member on function]
int **What**()

PtreeStaticUserStatementExpr [member on function]
Ptree * **Translate**(**Walker** *)

PtreeStaticUserStatementExpr [member on function]
void **Typeof**(**Walker** * , **TypeInfo** &)

QuoteClass [class]

parents: parent class

QuoteClass **bool** **Initialize**() [member on function]

QuoteClass **char** * [member on function]
MetaclassName()

QuoteClass **Ptree** * [member on function]
TranslateMemberCall(**Environment** * , **Ptree** * , **Ptree** *)

QuoteClass **Ptree** * [member on function]
ProcessBackQuote(**Environment** * , **char** * , **Ptree** * , **Ptree** *)

Token [class]

Token **bool** **Eq**(**char** *c*) [member on function]

char * **ptr** [data member]

int **len** [data member]

int **kind** [data member]

Lex [class]

parents: parent class

Lex **Lex**(**Program** *) [member on function]

| | |
|---|----------------------|
| Lex int GetToken(Token &) | [member on function] |
| Lex int LookAhead(int) | [member on function] |
| Lex int LookAhead(int , Token &) | [member on function] |
| Lex char * Save() | [member on function] |
| Lex void Restore(char *) | [member on function] |
| Lex void GetOnlyClosingBracket(Token &) | [member on function] |
| Lex Ptree * GetComments() | [member on function] |
| Lex Ptree * GetComments2() | [member on function] |
| Lex uint LineNumber(char * , char * & , int &) | [member on function] |
| Lex bool RecordKeyword(char * , int) | [member on function] |
| Lex bool Reify(Ptree * , unsigned int &) | [member on function] |
| Lex bool Reify(Ptree * t, char * &) | [member on function] |
| TokenFifo | [class] |
| TokenFifo TokenFifo(Lex *) | [member on function] |
| TokenFifo ~TokenFifo() | [member on function] |
| TokenFifo void Clear() | [member on function] |
| TokenFifo void Push(int , char * , int) | [member on function] |
| TokenFifo int Pop(char * & , int &) | [member on function] |
| TokenFifo int Peek(int) | [member on function] |
| TokenFifo int Peek(int , char * & , int &) | [member on function] |

| | | |
|------------------------|---|----------------------|
| <code>TokenFifo</code> | <code>int Peek2(int)</code> | [member on function] |
| <code>Lex</code> | <code>* lex</code> | [data member] |
| <code>int</code> | <code>head</code> | [data member] |
| <code>int</code> | <code>tail</code> | [data member] |
| <code>int</code> | <code>size</code> | [data member] |
| <code>Slot</code> | | [struct] |
| <code>int</code> | <code>token</code> | [data member] |
| <code>char</code> | <code>* pos</code> | [data member] |
| <code>int</code> | <code>len</code> | [data member] |
| <code>Slot</code> | <code>* ring</code> | [data member] |
| <code>Lex</code> | <code>uint Tokenp()</code> | [member on function] |
| <code>Lex</code> | <code>int TokenLen()</code> | [member on function] |
| <code>Lex</code> | <code>char * TokenPosition()</code> | [member on function] |
| <code>Lex</code> | <code>char Ref(uint i)</code> | [member on function] |
| <code>Lex</code> | <code>void Rewind(char *)</code> | [member on function] |
| <code>Lex</code> | <code>int ReadToken(char * & , int &)</code> | [member on function] |
| <code>Lex</code> | <code>void SkipAttributeToken()</code> | [member on function] |
| <code>Lex</code> | <code>int SkipExtensionToken(char * & , int &)</code> | [member on function] |
| <code>Lex</code> | <code>char GetNextNonWhiteChar()</code> | [member on function] |
| <code>Lex</code> | <code>int ReadLine()</code> | [member on function] |
| <code>Lex</code> | <code>bool ReadCharConst(uint top)</code> | [member on function] |
| <code>Lex</code> | <code>bool ReadStrConst(uint top)</code> | [member on function] |
| <code>Lex</code> | <code>int ReadNumber(char c , uint top)</code> | [member on function] |

| | |
|--|----------------------|
| Lex int ReadFloat(uint <i>top</i>) | [member on function] |
| Lex bool ReadLineDirective() | [member on function] |
| Lex int ReadIdentifier(uint <i>top</i>) | [member on function] |
| Lex int Screening(char * <i>identifier</i> , int <i>len</i>) | [member on function] |
| Lex int ReadSeparator(char <i>c</i> , uint <i>top</i>) | [member on function] |
| Lex int SingleCharOp(unsigned char <i>c</i>) | [member on function] |
| Lex int ReadComment(char <i>c</i> , uint <i>top</i>) | [member on function] |
| Program * file | [data member] |
| TokenFifo fifo | [data member] |
| uint tokenp | [data member] |
| int token_len | [data member] |
| int last_token | [data member] |
| HashTable * user_keywords | [data member] |
| Ptree * comments | [data member] |
| bool is_blank (char <i>c</i>) | [function] |
| bool is_letter (char <i>c</i>) | [function] |
| bool is_digit (char <i>c</i>) | [function] |
| bool is_xletter (char <i>c</i>) | [function] |
| bool is_eletter (char <i>c</i>) | [function] |
| bool is_hexdigit (char <i>c</i>) | [function] |
| bool is_int_suffix (char <i>c</i>) | [function] |
| bool is_float_suffix (char <i>c</i>) | [function] |
| TypeInfoId | [enum] |

| | |
|---|----------------------|
| UndefType | [enumerator] |
| BuiltInType | [enumerator] |
| ClassType | [enumerator] |
| EnumType | [enumerator] |
| TemplateType | [enumerator] |
| PointerType | [enumerator] |
| ReferenceType | [enumerator] |
| PointerToMemberType | [enumerator] |
| ArrayType | [enumerator] |
| FunctionType | [enumerator] |
| bool BOOL | [typedef] |
| int sint | [typedef] |
| unsigned int uint | [typedef] |
| gc LightObject | [typedef] |
| gc_cleanup Object | [typedef] |
| Walker | [class] |
| parents: parent class | |
| Walker Walker(Parser *) | [member on function] |
| Walker Walker(Parser * , Environment *) | [member on function] |
| Walker Walker(Environment *) | [member on function] |
| Walker Walker(Walker *) | [member on function] |
| Walker Ptree * Translate(Ptree *) | [member on function] |
| Walker void Typeof(Ptree * , TypeInfo &) | [member on function] |

```

Walker bool IsClassWalker()           [member on function]

Walker Ptree * TranslatePtree(       [member on function]
    Ptree * )

Walker void TypeofPtree( Ptree *     [member on function]
    , TypeInfo & )

Walker Ptree *                       [member on function]
    TranslateTypedef( Ptree * )

Walker Ptree *                       [member on function]
    TranslateTemplateDecl( Ptree * )

Walker Ptree *                       [member on function]
    TranslateTemplateInstantiation( Ptree * )

Walker Ptree *                       [member on function]
    TranslateTemplateInstantiation( Ptree * , Ptree *
    , Ptree * , Class * )

Walker Class *                       [member on function]
    MakeTemplateInstantiationMetaobject( Ptree *
    full_class_spec, Ptree * userkey, Ptree *
    class_spec)

Walker Ptree *                       [member on function]
    TranslateExternTemplate( Ptree * )

Walker Ptree *                       [member on function]
    TranslateTemplateClass( Ptree * , Ptree * )

Walker Class *                       [member on function]
    MakeTemplateClassMetaobject( Ptree * , Ptree *
    , Ptree * )

Walker Ptree *                       [member on function]
    TranslateTemplateFunction( Ptree * , Ptree * )

Walker Ptree *                       [member on function]
    TranslateMetaclassDecl( Ptree * )

Walker Ptree *                       [member on function]
    TranslateLinkageSpec( Ptree * )

Walker Ptree *                       [member on function]
    TranslateNamespaceSpec( Ptree * )

```


Walker **Ptree *** TranslateUsing([member on function]
Ptree *)

Walker **Ptree *** TranslateDeclaration(Ptree *) [member on function]

Walker **Ptree *** TranslateStorageSpecifiers(Ptree *) [member on function]

Walker **Ptree *** TranslateDeclarators(Ptree *) [member on function]

Walker **Ptree *** TranslateDeclarator(bool , PtreeDeclarator *) [member on function]

Walker **bool** GetArgDeclList([member on function]
PtreeDeclarator * , Ptree * &)

Walker **Ptree *** TranslateArgDeclList(bool , Ptree * , Ptree *) [member on function]

Walker **Ptree *** TranslateArgDeclList2(bool , Environment * ,
bool , bool , int , Ptree *) [member on function]

Walker **Ptree *** FillArgumentName(Ptree * , Ptree * , int
arg_name) [member on function]

Walker **Ptree *** TranslateInitializeArgs(PtreeDeclarator * ,
Ptree *) [member on function]

Walker **Ptree *** TranslateAssignInitializer(PtreeDeclarator * ,
Ptree *) [member on function]

Walker **Ptree *** TranslateFunctionImplementation(Ptree *) [member on function]

Walker **Ptree *** RecordArgsAndTranslateFbody(Class * , Ptree *
args , Ptree * *body*) [member on function]

Walker **Ptree *** TranslateFunctionBody(Ptree *) [member on function]

Walker **Ptree *** TranslateBrace([member on function]
Ptree *)

Walker **Ptree *** TranslateBlock([member on function]
Ptree *)

Walker **Ptree *** [member on function]
TranslateClassBody(Ptree * , Ptree * , Class *
)

Walker **Ptree *** [member on function]
TranslateClassSpec(Ptree *)

Walker **Class *** [member on function]
MakeClassMetaobject(Ptree * , Ptree * , Ptree
*)

Walker **Ptree *** [member on function]
TranslateClassSpec(Ptree * , Ptree * , Ptree *
, Class *)

Walker **Ptree *** [member on function]
TranslateEnumSpec(Ptree *)

Walker **Ptree *** [member on function]
TranslateAccessSpec(Ptree *)

Walker **Ptree *** [member on function]
TranslateAccessDecl(Ptree *)

Walker **Ptree *** [member on function]
TranslateUserAccessSpec(Ptree *)

Walker **Ptree *** TranslateIf([member on function]
Ptree *)

Walker **Ptree *** TranslateSwitch([member on function]
Ptree *)

Walker **Ptree *** TranslateWhile([member on function]
Ptree *)

Walker **Ptree *** TranslateDo([member on function]
Ptree *)

Walker **Ptree *** TranslateFor([member on function]
Ptree *)

Walker **Ptree *** TranslateTry([member on function]
Ptree *)

Walker **Ptree *** TranslateBreak([member on function]
Ptree *)

Walker **Ptree *** TranslateContinue([member on function]
Ptree *)

Walker **Ptree *** TranslateReturn([member on function]
Ptree *)

Walker **Ptree *** TranslateGoto([member on function]
Ptree *)

Walker **Ptree *** TranslateCase([member on function]
Ptree *)

Walker **Ptree *** TranslateDefault([member on function]
Ptree *)

Walker **Ptree *** TranslateLabel([member on function]
Ptree *)

Walker **Ptree *** TranslateExprStatement([member on function]
Ptree *)

Walker **Ptree *** TranslateTypespecifier([member on function]
Ptree *)

Walker **Ptree *** TranslateComma([member on function]
Ptree *)

Walker **Ptree *** TranslateAssign([member on function]
Ptree *)

Walker **Ptree *** TranslateCond([member on function]
Ptree *)

Walker **Ptree *** TranslateInfix([member on function]
Ptree *)

Walker **Ptree *** TranslatePm([member on function]
Ptree *)

Walker **Ptree *** TranslateCast([member on function]
Ptree *)

| | | |
|--------|---|----------------------|
| Walker | Ptree * TranslateUnary(Ptree *) | [member on function] |
| Walker | Ptree * TranslateThrow(Ptree *) | [member on function] |
| Walker | Ptree * TranslateSizeof(Ptree *) | [member on function] |
| Walker | Ptree * TranslateTypeid(Ptree *) | [member on function] |
| Walker | Ptree * TranslateTypeof(Ptree *) | [member on function] |
| Walker | Ptree * TranslateNew(Ptree *) | [member on function] |
| Walker | Ptree * TranslateNew2(Ptree * , Ptree * , Ptree * , Ptree * , Ptree * , Ptree * , Ptree *) | [member on function] |
| Walker | Ptree * TranslateNew3(Ptree * type) | [member on function] |
| Walker | Ptree * TranslateDelete(Ptree *) | [member on function] |
| Walker | Ptree * TranslateThis(Ptree *) | [member on function] |
| Walker | Ptree * TranslateVariable(Ptree *) | [member on function] |
| Walker | Ptree * TranslateFstyleCast(Ptree *) | [member on function] |
| Walker | Ptree * TranslateArray(Ptree *) | [member on function] |
| Walker | Ptree * TranslateFuncall(Ptree *) | [member on function] |
| Walker | Ptree * TranslatePostfix(Ptree *) | [member on function] |
| Walker | Ptree * TranslateUserStatement(Ptree *) | [member on function] |

| | | |
|--------|---|----------------------|
| Walker | Ptree * TranslateDotMember(Ptree *) | [member on function] |
| Walker | Ptree * TranslateArrowMember(Ptree *) | [member on function] |
| Walker | Ptree * TranslateParen(Ptree *) | [member on function] |
| Walker | Ptree * TranslateStaticUserStatement(Ptree *) | [member on function] |
| Walker | void TypeofComma(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofAssign(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofCond(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofInfix(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofPm(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofCast(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofUnary(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofThrow(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofSizeof(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofTypeid(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofTypeof(Ptree * , TypeInfo &) | [member on function] |
| Walker | void TypeofNew(Ptree * , TypeInfo &) | [member on function] |

| | |
|--|----------------------|
| Walker void TypeofDelete(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofThis(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofVariable(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofFstyleCast(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofArray(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofFuncall(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofPostfix(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofUserStatement(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofDotMember(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofArrowMember(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofParen(Ptree * , TypeInfo &) | [member on function] |
| Walker void TypeofStaticUserStatement(Ptree * , TypeInfo &) | [member on function] |
| NameScope | [struct] |
| Environment * env | [data member] |
| Walker * walker | [data member] |
| Walker void NewScope() | [member on function] |
| Walker void NewScope(Class *) | [member on function] |
| Walker Environment * ExitScope() | [member on function] |

Walker **void** RecordBaseclassEnv([member on function]
Ptree *)

Walker **NameScope** ChangeScope([member on function]
Environment *)

Walker **void** RestoreScope([member on function]
NameScope &)

Walker **Ptree *** [member on function]
TranslateDeclarators(Ptree * , bool)

Walker **Class *** LookupMetaClass([member on function]
Ptree * , Ptree * , Ptree * , bool)

Walker **Class *** [member on function]
LookupBaseMetaClass(Ptree * , Ptree * , bool)

Walker **Ptree *** [member on function]
TranslateNewDeclarator(Ptree * decl)

Walker **Ptree *** [member on function]
TranslateNewDeclarator2(Ptree * decl)

Walker **Ptree *** [member on function]
TranslateArguments(Ptree *)

Walker **Ptree *** [member on function]
GetClassOrEnumSpec(Ptree *)

Walker **Ptree *** [member on function]
GetClassTemplateSpec(Ptree *)

Walker **Ptree *** [member on function]
StripCvFromIntegralType(Ptree *)

Walker **void** [member on function]
SetDeclaratorComments(Ptree * , Ptree *)

Walker **Ptree *** FindLeftLeaf([member on function]
Ptree * node, Ptree * & parent)

Walker **void** SetLeafComments([member on function]
Ptree * , Ptree *)

Walker **Ptree *** NthDeclarator([member on function]
Ptree * , int &)

```

Walker Ptree * FindDeclarator(      [member on function]
    Ptree * , char * , int , char * , int & ,
    Environment * )

Walker bool MatchedDeclarator(      [member on function]
    Ptree * , char * , int , char * , Environment * )

Walker bool WhichDeclarator(        [member on function]
    Ptree * , Ptree * , int & , Environment * )

Walker void ErrorMessage( char *    [member on function]
    , Ptree * , Ptree * )

Walker void WarningMessage( char    [member on function]
    * , Ptree * , Ptree * )

Walker void                               [member on function]
    InaccurateErrorMessage( char * , Ptree * ,
    Ptree * )

Walker void                               [member on function]
    InaccurateWarningMessage( char * , Ptree * ,
    Ptree * )

Walker void                               [member on function]
    ChangeDefaultMetaclass( char * )

Walker Parser * GetParser()          [member on function]

Environment * env                        [data member]

Parser * parser                          [data member]

char * argument_name                    [data member]

Parser * default_parser                 [data member]

char * default_metaclass                [data member]

```


3.2 The IDL Module

| | |
|--|-------------------------------|
| omni | [module] |
| strip (<i>filename</i>) | [function] |
| strip_filename (<i>filename</i>) | [function] |
| This is aliased as strip if -b used and basename set | |
| usage () | [function] |
| --parseArgs (<i>args</i> , <i>config_obj</i>) | [function] |
| parse (<i>file</i> , <i>extra_files</i> , <i>args</i> , <i>config_obj</i>) | [function] |
| TypeTranslator | [class] |
| parents: parent class maps idltype objects to Synopsis.Type objects in a Type.Dictionary | |
| __init__ (<i>self</i> , <i>types</i>) | [operation on TypeTranslator] |
| internalize (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| add (<i>self</i> , <i>name</i> , <i>type</i>) | [operation on TypeTranslator] |
| get (<i>self</i> , <i>name</i>) | [operation on TypeTranslator] |
| visitBaseType (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| visitStringType (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| visitWStringType (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| visitSequenceType (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| visitDeclaredType (<i>self</i> , <i>idltype</i>) | [operation on TypeTranslator] |
| ASTTranslator | [class] |
| parents: parent class | |
| __init__ (<i>self</i> , <i>declarations</i> , <i>types</i> , <i>mainfile_only</i>) | [operation on ASTTranslator] |

| | |
|--|------------------------------|
| scope (<i>self</i>) | [operation on ASTTranslator] |
| addDeclaration (<i>self</i> , <i>declaration</i>) | [operation on ASTTranslator] |
| addType (<i>self</i> , <i>name</i> , <i>type</i>) | [operation on ASTTranslator] |
| getType (<i>self</i> , <i>name</i>) | [operation on ASTTranslator] |
| visitAST (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitModule (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitInterface (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitForward (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitConst (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitTypedef (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitMember (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitStruct (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitException (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitUnionCase (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitUnion (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitEnumerator (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitEnum (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitAttribute (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitParameter (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |
| visitOperation (<i>self</i> , <i>node</i>) | [operation on ASTTranslator] |

3.3 The Python Module

exparse [module]

Simple code to extract class & function docstrings from a module.

This code is used as an example in the library reference manual in the section on using the parser module. Refer to the manual for a thorough discussion of the operation of this code.

The code has been extended by Stephen Davies for the Synopsis project. It now also recognises parameter names and values, and baseclasses. Names are now returned in order also.

findModulePath(*module*) [function]

Return the path for the given module

format(*tree*, *depth*) [function]

Format the given tree up to the given depth. Numbers are replaced with their symbol or token names.

stringify(*tree*) [function]

Convert the given tree to a string

get_docs(*file*) [function]

Retrieve information from the parse tree of a source file.

file Name of the file to read Python source code from.

filter_names(*list*) [function]

map_second(*list*) [function]

map_rest(*list*) [function]

get_names_only(*list*) [function]

match(*pattern*, *data*, *vars*) [function]

Match 'data' to 'pattern', with variable extraction.

pattern Pattern to match against, possibly containing variables.

data Data to be checked and against which variables are extracted.

vars Dictionary of variables which have already been found. If not provided, an empty dictionary is created.

The 'pattern' value may contain variables of the form ['varname'] which are allowed to match anything. The value that is matched is returned as part of a dictionary which maps 'varname' to the matched value. 'varname' is not required to be a string object, but using strings makes patterns and the code which uses them more readable.

This function returns two values: a boolean indicating whether a match was found and a dictionary mapping variable names to their associated values.

dmatch(*pattern*, *data*, *vars*) [function]
 Debugging match

SuiteInfoBase [class]

__init__(*self*, *tree*, *env*) [operation on SuiteInfoBase]

_extract_info(*self*, *tree*) [operation on SuiteInfoBase]

_addImport(*self*, *names*) [operation on SuiteInfoBase]

_addFromImport(*self*, *module*,
names) [operation on SuiteInfoBase]

get_docstring(*self*) [operation on SuiteInfoBase]

get_name(*self*) [operation on SuiteInfoBase]

get_class_names(*self*) [operation on SuiteInfoBase]

get_class_info(*self*, *name*) [operation on SuiteInfoBase]

__getitem__(*self*, *name*) [operation on SuiteInfoBase]

SuiteFuncInfo [class]

get_function_names(*self*) [operation on SuiteFuncInfo]

get_function_info(*self*, *name*) [operation on SuiteFuncInfo]

FunctionInfo [class]

parents: parent class parent class

__init__(*self*, *tree*, *env*) [operation on FunctionInfo]

get_params(*self*) [operation on FunctionInfo]

get_param_defaults(*self*) [operation on FunctionInfo]

ClassInfo [class]

parents: parent class

__init__(*self*, *tree*, *env*) [operation on ClassInfo]

get_method_names(*self*) [operation on ClassInfo]

get_method_info(*self*, *name*) [operation on ClassInfo]

get_base_names(*self*) [operation on `ClassInfo`]

ModuleInfo [class]

parents: parent class parent class

__init__(*self*, *tree*, *name*) [operation on `ModuleInfo`]

python [module]

Main module for the Python parser. Parsing python is achieved by using the code in the Python distribution that is an example for parsing python by using the built-in parser. This parser returns a parse tree which we can traverse and translate into Synopsis' AST. The exparse module contains the enhanced example code (it has many more features than the simple example did), and this module translates the resulting intermediate AST objects into Synopsis.Core.AST objects.

addDeclaration(*decl*) [function]

Adds the given declaration to the current top scope and to the SourceFile for this file.

push(*scope*) [function]

Pushes the given scope onto the top of the stack

pop() [function]

Pops the scope stack by one level

scopeName(*name*) [function]

Scopes the given name. If the given name is a list then it is returned verbatim, else it is concatenated with the (scoped) name of the current scope

process_ModuleInfo(*mi*) [function]

Processes a ModuleInfo object. The comments are extracted, and any functions and comments recursively processed.

add_params(*func*, *fi*) [function]

Adds the parameters of 'fi' to the AST.Function 'func'.

process_FunctionInfo(*fi*) [function]

Process a FunctionInfo object. An AST.Function object is created and inserted into the current scope.

process_MethodInfo(*fi*) [function]

Process a MethodInfo object. An AST.Operation object is created and inserted into the current scope.

| | |
|--|------------|
| process_ClassInfo(<i>ci</i>) | [function] |
| Process a ClassInfo object. An AST.Class object is created and inserted into the current scope. The inheritance of the class is also parsed, and nested classes and methods recursively processed. | |
| usage() | [function] |
| Prints a usage message | |
| --parseArgs(<i>args</i>, <i>config_obj</i>) | [function] |
| Parses the command line arguments and the config object | |
| get_synopsis(<i>file</i>) | [function] |
| Returns the docstring from the top of an open file | |
| parse(<i>file</i>, <i>extra_files</i>, <i>parser_args</i>, <i>config_obj</i>) | [function] |
| Entry point for the Python parser | |

4 The Linker Module

The linker performs some first modifications to the AST/Type dictionary as to resolve cross references (even across languages !), strip off unwanted scopes, parse the comments that are associated with the declarations for cross references, etc.

AccessRestrictor [module]

AccessRestrictor [class]

parents: parent class This class processes declarations, and removes those that need greater access than the maximum passed to the constructor

`__init__(self)` [operation on AccessRestrictor]

`execute(self, ast)` [operation on AccessRestrictor]

`push(self)` [operation on AccessRestrictor]

`pop(self, decl)` [operation on AccessRestrictor]

`add(self, decl)` [operation on AccessRestrictor]

`currscope(self)` [operation on AccessRestrictor]

`visitDeclaration(self, decl)` [operation on AccessRestrictor]

`visitScope(self, scope)` [operation on AccessRestrictor]

Comments [module]

Comment Processor

CommentProcessor [class]

parents: parent class Base class for comment processors.

This is an AST visitor, and by default all declarations call process() with the current declaration. Subclasses may override just the process method.

`processAll(self, declarations)` [operation on CommentProcessor]

`process(self, decl)` [operation on CommentProcessor]

Process comments for the given declaration

`visitDeclaration(self, decl)` [operation on CommentProcessor]

SSDComments [class]

parents: parent class A class that selects only //. comments.

__init__(*self*) [operation on SSDComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on SSDComments]
Calls processComment on all comments

processComment(*self*, *comment*) [operation on SSDComments]
Replaces the text in the comment. It calls strip_star() first to remove all multi-line star comments, then follows with parse_ssd().

strip_star(*self*, *str*) [operation on SSDComments]
Strips all star-format comments from the string

parse_ssd(*self*, *str*) [operation on SSDComments]
Filters str and returns just the lines that start with //.

JavaComments [class]

parents: parent class A class that formats java /** style comments

__init__(*self*) [operation on JavaComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on JavaComments]
Calls processComment on all comments

processComment(*self*, *comment*) [operation on JavaComments]
Finds comments in the java format. The format is /** ... */, and it has to cater for all four line forms: "/** ... ", " * ...", " */" and the one-line "/** ... */".

SSComments [class]

parents: parent class A class that selects only // comments.

__init__(*self*) [operation on SSComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on SSComments]
Calls processComment on all comments

processComment(*self*, *comment*) [operation on SSComments]
Replaces the text in the comment. It calls strip_star() first to remove all multi-line star comments, then follows with parse_ss().

strip_star(*self*, *str*) [operation on SSComments]
Strips all star-format comments from the string

parse_ss(*self*, *str*) [operation on SSComments]
Filters *str* and returns just the lines that start with //

QtComments [class]

parents: parent class A class that finds Qt style comments. These have two styles: `///...` and `/*!...*/`. The first means "brief comment" and there must only be one. The second type is the detailed comment.

__init__(*self*) [operation on QtComments]
Compiles the regular expressions

process(*self*, *decl*) [operation on QtComments]
Calls `processComment` on all comments

processComment(*self*, *comment*) [operation on QtComments]
Matches either brief or detailed comments

Transformer [class]

parents: parent class A class that creates a new AST from an old one. This is a helper base for more specialized classes that manipulate the AST based on the comments in the nodes

__init__(*self*) [operation on Transformer]
Constructor

processAll(*self*, *declarations*) [operation on Transformer]
Overrides the default `processAll()` to setup the stack

push(*self*) [operation on Transformer]
Pushes the current scope onto the stack and starts a new one

pop(*self*, *decl*) [operation on Transformer]
Pops the current scope from the stack, and appends the given declaration to it

add(*self*, *decl*) [operation on Transformer]
Adds the given *decl* to the current scope

currscope(*self*) [operation on Transformer]
Returns the current scope: a list of declarations

Dummies [class]

parents: parent class A class that deals with dummy declarations and their comments. This class just removes them.

visitDeclaration(*self*, *decl*) [operation on Dummies]
Checks for dummy declarations

visitScope(*self*, *scope*) [operation on Dummies]
Visits all children of the scope in a new scope. The value of `currscope()` at the end of the list is used to replace `scope`'s list of declarations - hence you can remove (or insert) declarations from the list. Such as dummy declarations :)

visitEnum(*self*, *enum*) [operation on Dummies]
Does the same as `visitScope`, but for the `enum`'s list of enumerators

visitEnumerator(*self*, *enumerator*) [operation on Dummies]
Removes dummy enumerators

Previous [class]
parents: parent class A class that maps comments that begin with '<' to the previous declaration

processAll(*self*, *declarations*) [operation on Previous]
decorates `processAll()` to initialise `last` and `laststack`

push(*self*) [operation on Previous]
decorates `push()` to also push 'last' onto 'laststack'

pop(*self*, *decl*) [operation on Previous]
decorates `pop()` to also pop 'last' from 'laststack'

visitScope(*self*, *scope*) [operation on Previous]
overrides `visitScope()` to set 'last' after each declaration

checkPrevious(*self*, *decl*) [operation on Previous]
Checks a `decl` to see if the comment should be moved. If the comment begins with a less-than sign, then it is moved to the 'last' declaration

removeSuspect(*self*, *decl*) [operation on Previous]
Removes any suspect comments from the declaration

visitDeclaration(*self*, *decl*) [operation on Previous]
Calls `checkPrevious` on the declaration and removes dummies

visitEnum(*self*, *enum*) [operation on Previous]
Does the same as `visitScope` but for `enum` and enumerators

visitEnumerator(*self*, *enumerator*) [operation on Previous]
 Checks previous comment and removes dummies

Grouper [class]
 parents: parent class A class that detects grouping tags and moves the enclosed nodes into a subnode (a 'Group')

__init__(*self*) [operation on Grouper]

visitDeclaration(*self*, *decl*) [operation on Grouper]
 Checks for grouping tags. If an opening tag is found in the middle of a comment, a new Group is generated, the preceding comments are associated with it, and is pushed onto the scope stack as well as the groups stack.

visitScope(*self*, *scope*) [operation on Grouper]
 Visits all children of the scope in a new scope. The value of currscope() at the end of the list is used to replace scope's list of declarations - hence you can remove (or insert) declarations from the list. Such as dummy declarations :)

visitEnum(*self*, *enum*) [operation on Grouper]
 Does the same as visitScope, but for the enum's list of enumerators

visitEnumerator(*self*, *enumerator*) [operation on Grouper]
 Removes dummy enumerators

Summarizer [class]
 parents: parent class Splits comments into summary/detail parts.

__init__(*self*) [operation on Summarizer]

process(*self*, *decl*) [operation on Summarizer]
 Combine and summarize the comments of this declaration.

JavaTags [class]
 parents: parent class Extracts javadoc-style @tags from the end of comments.

__init__(*self*) [operation on JavaTags]

process(*self*, *decl*) [operation on JavaTags]
 Extract tags from each comment of the given decl

Comments [class]
 parents: parent class

__init__(*self*) [operation on Comments]
 Constructor, parses the config object

execute(*self*, *ast*) [operation on Comments]

EmptyNS [module]

EmptyNS [class]

parents: parent class parent class A class that removes empty namespaces

__init__(*self*) [operation on EmptyNS]
 Overrides the default processAll() to setup the stack

execute(*self*, *ast*) [operation on EmptyNS]

push(*self*) [operation on EmptyNS]
 Pushes the current scope onto the stack and starts a new one

pop(*self*, *decl*) [operation on EmptyNS]
 Pops the current scope from the stack, and appends the given declaration to it

pop_only(*self*) [operation on EmptyNS]
 Only pops, doesn't append to scope

add(*self*, *decl*) [operation on EmptyNS]
 Adds the given decl to the current scope

currscope(*self*) [operation on EmptyNS]
 Returns the current scope: a list of declarations

visitDeclaration(*self*, *decl*) [operation on EmptyNS]
 Adds declaration to scope

visitGroup(*self*, *group*) [operation on EmptyNS]
 Overrides recursive behaviour to just add the group

visitEnum(*self*, *enum*) [operation on EmptyNS]
 Overrides recursive behaviour to just add the enum

visitModule(*self*, *module*) [operation on EmptyNS]
 Visits all children of the module, and if there are no declarations after that removes the module

_count_not_forwards(*self*, *decls*) [operation on EmptyNS]
 Returns the number of declarations not instances of AST.Forward

| | |
|---|-------------------------------|
| LanguageMapper | [module] |
| LanguageMapper | [class] |
| parents: parent class | |
| execute(<i>self</i>, <i>ast</i>) | [operation on LanguageMapper] |
| Linker | [module] |
| usage() | [function] |
| __parseArgs(<i>args</i>, <i>config_obj</i>) | [function] |
| mapTypes(<i>m</i>) | [function] |
| resolve(<i>args</i>, <i>ast</i>, <i>config_obj</i>) | [function] |
| Config | [class] |
| Central configuration repository for Linker. | |
| __init__(<i>self</i>) | [operation on Config] |
| Constructor - initialise objects to None. | |
| use_config(<i>self</i>, <i>obj</i>) | [operation on Config] |
| Extracts useful attributes from 'obj' and stores them. The object itself is also stored as config.obj | |
| _config_verbose(<i>self</i>, <i>verbose</i>) | [operation on Config] |
| Configures from the given verbose boolean | |
| _config_strip(<i>self</i>, <i>strip</i>) | [operation on Config] |
| Configures from the given list of strip | |
| _config_mapper_list(<i>self</i>, <i>mapper_list</i>) | [operation on Config] |
| Configures from the given list of mapper_list | |
| _config_operations(<i>self</i>, <i>operations</i>) | [operation on Config] |
| Configures from the given list of operations | |
| _config_max_access(<i>self</i>, <i>access</i>) | [operation on Config] |
| _config_map_declaration_names(<i>self</i>, <i>names</i>) | [operation on Config] |
| _config_map_declaration_type(<i>self</i>, <i>typename</i>) | [operation on Config] |

`_config_comment_processors(self, processors)` [operation on `Config`]

Operation [class]

The base class for Linker operations. The linker executes a number of operations, depending on the config option 'operations'. Each operation may use other config options

`execute(self)` [operation on `Operation`]
Executes this operation

CXX2IDL [class]

this function maps a C++ external reference to an IDL interface if the name either starts with 'POA_' or ends in '_ptr'

`__init__(self)` [operation on `CXX2IDL`]

`map(self, unknown)` [operation on `CXX2IDL`]

Mapper [class]

parents: parent class allow user to supply a mapping functor that is applied to Unknown types. This is useful for linking to externally defined types, such as when cross- referencing modules written in different languages

`__init__(self, mappers)` [operation on `Mapper`]

`visitUnknown(self, unknown)` [operation on `Mapper`]

NameMapper [module]

NameMapper [class]

parents: parent class parent class This class adds a prefix to all declaration and type names.

`visitDeclaration(self, decl)` [operation on `NameMapper`]
Changes the name of this declaration and its associated type

`visitGroup(self, node)` [operation on `NameMapper`]
Recursively visits declarations under this group/scope/etc

`execute(self, ast)` [operation on `NameMapper`]

Stripper [module]

`filterName(name, prefixes)` [function]

Stripper [class]

parents: parent class parent class Strip common prefix from the declaration's name. Keep a list of root nodes, such that children whose parent scopes are not accepted but which themselves are correct can be maintained as new root nodes.

`__init__(self)` [operation on Stripper]

`execute(self, ast)` [operation on Stripper]

`_stripName(self, name)` [operation on Stripper]

`stripDeclarations(self, declarations)` [operation on Stripper]

`stripTypes(self, types)` [operation on Stripper]

`declarations(self)` [operation on Stripper]

`strip(self, declaration)` [operation on Stripper]

test whether the declaration matches one of the prefixes, strip it off, and return success. Success means that the declaration matches the prefix set and thus should not be removed from the AST.

`visitScope(self, scope)` [operation on Stripper]

`visitClass(self, clas)` [operation on Stripper]

`visitDeclaration(self, decl)` [operation on Stripper]

`visitEnumerator(self, enumerator)` [operation on Stripper]

`visitEnum(self, enum)` [operation on Stripper]

`visitFunction(self, function)` [operation on Stripper]

`visitOperation(self, operation)` [operation on Stripper]

`visitMetaModule(self, module)` [operation on Stripper]

Unduplicator [module]**Unduplicator** [class]

parents: parent class parent class Visitor that removes duplicate declarations

`__init__(self)` [operation on Unduplicator]

`execute(self, ast)` [operation on Unduplicator]

| | |
|--|-----------------------------|
| lookup (<i>self</i> , <i>name</i>) | [operation on Unduplicator] |
| look whether the current scope already contains a declaration with the given name | |
| append (<i>self</i> , <i>declaration</i>) | [operation on Unduplicator] |
| append declaration to the current scope | |
| push (<i>self</i> , <i>scope</i>) | [operation on Unduplicator] |
| push new scope on the stack | |
| pop (<i>self</i>) | [operation on Unduplicator] |
| restore the previous scope | |
| top (<i>self</i>) | [operation on Unduplicator] |
| top_dict (<i>self</i>) | [operation on Unduplicator] |
| linkType (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| Returns the same or new proxy type | |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitTemplate (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitArray (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Unduplicator] |
| visitSourceFile (<i>self</i> , <i>file</i>) | [operation on Unduplicator] |
| Resolves any duplicates in the list of declarations from this file | |
| visitModule (<i>self</i> , <i>module</i>) | [operation on Unduplicator] |
| merge_comments (<i>self</i> , <i>dest</i> , <i>src</i>) | [operation on Unduplicator] |
| Merges the src comments into dest. Merge is just an append, unless src already exists inside dest! | |
| visitMetaModule (<i>self</i> , <i>module</i>) | [operation on Unduplicator] |

| | |
|--|-----------------------------|
| addDeclaration (<i>self</i> , <i>decl</i>) | [operation on Unduplicator] |
| Adds a declaration to the current (top) scope. If there is already a Forward declaration, then this replaces it unless this is also a Forward. | |
| visitNamed (<i>self</i> , <i>decl</i>) | [operation on Unduplicator] |
| visitFunction (<i>self</i> , <i>func</i>) | [operation on Unduplicator] |
| visitVariable (<i>self</i> , <i>var</i>) | [operation on Unduplicator] |
| visitTypedef (<i>self</i> , <i>tdef</i>) | [operation on Unduplicator] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on Unduplicator] |
| visitInheritance (<i>self</i> , <i>parent</i>) | [operation on Unduplicator] |
| visitParameter (<i>self</i> , <i>param</i>) | [operation on Unduplicator] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on Unduplicator] |

XRefCompiler [module]

do_compile(*input_files*, *output_file*, *no_locals*) [function]

XRefCompiler [class]

parents: parent class This class compiles a set of text-based xref files from the C++ parser into a cPickled data structure with a name index.

The format of the data structure is:

```
(data, index) = cPickle.load()
data = dict<scoped targetname, target_data> index = dict<name, list<scoped
targetname>> target_data = (definitions = list<target_info>, func calls =
list<target_info>, references = list<target_info>) target_info = (filename,
int(line number), scoped context name) </pre> The scoped targetnames in the
index are guaranteed to exist in the data dictionary.
```

__init__(*self*) [operation on XRefCompiler]

execute(*self*, *ast*) [operation on XRefCompiler]

5 The Formatter Module

The backbone of synopsis is an Abstract Syntax Tree, which can be traversed using the Visitor Pattern. One of the main uses of Visitors is to provide different Formatters, which traverse the AST to generate formatted output. This module contains a range of formatters for various purposes, ranging from online browsing, over graph generation, to generating hardcopies.

5.1 The ASCII Module

| | |
|---|-------------------------------|
| usage() | [function] |
| Print usage to stdout | |
| --parseArgs(<i>args</i>) | [function] |
| print_types(<i>types</i>) | [function] |
| format(<i>args</i>, <i>ast</i>, <i>config_obj</i>) | [function] |
| ASCIIFormatter | [class] |
| parents: parent class parent class | |
| outputs as ascii. This is to test for features still missing. The output should be compatible with the input... | |
| __init__(<i>self</i>, <i>os</i>) | [operation on ASCIIFormatter] |
| indent(<i>self</i>) | [operation on ASCIIFormatter] |
| incr(<i>self</i>) | [operation on ASCIIFormatter] |
| decr(<i>self</i>) | [operation on ASCIIFormatter] |
| scope(<i>self</i>) | [operation on ASCIIFormatter] |
| set_scope(<i>self</i>, <i>name</i>) | [operation on ASCIIFormatter] |
| enterScope(<i>self</i>, <i>name</i>) | [operation on ASCIIFormatter] |
| leaveScope(<i>self</i>) | [operation on ASCIIFormatter] |
| write(<i>self</i>, <i>text</i>) | [operation on ASCIIFormatter] |
| formatType(<i>self</i>, <i>type</i>, <i>id_holder</i>) | [operation on ASCIIFormatter] |
| visitBaseType(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |
| visitDependent(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |
| visitUnknown(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |
| visitDeclared(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |
| visitModifier(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |
| visitParametrized(<i>self</i>, <i>type</i>) | [operation on ASCIIFormatter] |

| | |
|--|-------------------------------|
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitTemplate (<i>self</i> , <i>type</i>) | [operation on ASCIIFormatter] |
| visitDeclaration (<i>self</i> , <i>decl</i>) | [operation on ASCIIFormatter] |
| visitMacro (<i>self</i> , <i>macro</i>) | [operation on ASCIIFormatter] |
| writeComments (<i>self</i> , <i>comments</i>) | [operation on ASCIIFormatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on ASCIIFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on ASCIIFormatter] |
| visitMetaModule (<i>self</i> , <i>module</i>) | [operation on ASCIIFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on ASCIIFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on ASCIIFormatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on ASCIIFormatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on ASCIIFormatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on ASCIIFormatter] |
| visitVariable (<i>self</i> , <i>var</i>) | [operation on ASCIIFormatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on ASCIIFormatter] |
| visitEnumerator (<i>self</i> , <i>enumer</i>) | [operation on ASCIIFormatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on ASCIIFormatter] |

5.2 The HTML Module

ASTFormatter [module]

AST Formatting classes.

This module contains classes for formatting parts of a scope page (class, module, etc with methods, variables etc. The actual formatting of the declarations is delegated to multiple strategies for each part of the page, and are defined in the FormatStrategy module.

Part [class]

parents: parent class parent class Base class for formatting a Part of a Scope Page.

This class contains functionality for modularly formatting an AST node and its children for display. It is typically used to construct Heading, Summary and Detail formatters. Strategy objects are added according to configuration, and this base class then checks which format methods each strategy implements. For each AST declaration visited, the Part asks all Strategies which implement the appropriate format method to generate output for that declaration. The final writing of the formatted html is delegated to the writeSectionStart, writeSectionEnd, and writeSectionItem methods, which must be implemented in a subclass.

__init__(*self*, *page*) [operation on Part]

page(*self*) [operation on Part]

filename(*self*) [operation on Part]

os(*self*) [operation on Part]

scope(*self*) [operation on Part]

write(*self*, *text*) [operation on Part]

_init_formatters(*self*, *config_option*,
type_msg) [operation on Part]

Loads strategies from config file

addFormatter(*self*, *formatterClass*) [operation on Part]

Adds the given formatter Class. An object is instantiated from the class passing self to the constructor. Stores the object, and stores which format methods it overrides

type_ref(*self*) [operation on Part]

| | |
|---|---|
| type_label (<i>self</i>) | [operation on Part] |
| declarator (<i>self</i>) | [operation on Part] |
| parameter (<i>self</i>) | [operation on Part] |
| reference (<i>self</i> , <i>name</i> , <i>label</i> , **keys) | [operation on Part] Returns a reference to the given name. The name is a scoped name, and the optional label is an alternative name to use as the link text. The name is looked up in the TOC so the link may not be local. The optional keys are appended as attributes to the A tag. |
| label (<i>self</i> , <i>name</i> , <i>label</i>) | [operation on Part] Create a label for the given name. The label is an anchor so it can be referenced by other links. The name of the label is derived by looking up the name in the TOC and using the link in the TOC entry. The optional label is an alternative name to use as the displayed name. If the name is not found in the TOC then the name is not anchored and just label is returned (or name if no label is given). |
| formatDeclaration (<i>self</i> , <i>decl</i> , <i>method</i>) | [operation on Part] Format decl using named method of each formatter. Each formatter returns two strings - type and name. All the types are joined and all the names are joined separately. The consolidated type and name strings are then passed to writeSectionItem. |
| process (<i>self</i> , <i>decl</i>) | [operation on Part] Formats the given decl, creating the output for this Part of the page. This method is implemented in various subclasses in different ways, for example Summary and Detail iterate through the children of 'decl' section by section, whereas Heading only formats decl itself. |
| visitDeclaration (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitForward (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitGroup (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitScope (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitModule (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitMetaModule (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitClass (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitTypedef (<i>self</i> , <i>decl</i>) | [operation on Part] |

| | |
|---|---|
| visitEnum (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitVariable (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitConst (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitFunction (<i>self</i> , <i>decl</i>) | [operation on Part] |
| visitOperation (<i>self</i> , <i>decl</i>) | [operation on Part] |
| formatType (<i>self</i> , <i>typeObj</i> , <i>id_holder</i>) | [operation on Part] Returns a reference string for the given type object |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's name |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's link |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be a reference to the type's name |
| visitDependent (<i>self</i> , <i>type</i>) | [operation on Part] Sets the label to be the type's name (which has no proper scope) |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Part] Adds modifiers to the formatted label of the modifier's alias |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Part] Adds the parameters to the template name in angle brackets |
| visitTemplate (<i>self</i> , <i>type</i>) | [operation on Part] Labs the template with the parameters |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Part] Labels the function type with return type, name and parameters |
| write_start (<i>self</i>) | [operation on Part] Abstract method to start the output, eg table headings |
| writeSectionStart (<i>self</i> , <i>heading</i>) | [operation on Part] Abstract method to start a section of declaration types |
| writeSectionEnd (<i>self</i> , <i>heading</i>) | [operation on Part] Abstract method to end a section of declaration types |

writeSectionItem(*self*, *text*) [operation on Part]
 Abstract method to write the output of one formatted declaration

write_end(*self*) [operation on Part]
 Abstract method to end the output, eg close the table

Heading [class]

parents: parent class Heading page part. Displays a header for the page – its strategies are only passed the object that the page is for; ie a Class or Module

__init__(*self*, *page*) [operation on Heading]

_init_default_formatters(*self*) [operation on Heading]

writeSectionItem(*self*, *text*) [operation on Heading]
 Writes text and follows with a horizontal rule

process(*self*, *decl*) [operation on Heading]
 Process this Part by formatting only the given decl

Summary [class]

parents: parent class Formatting summary visitor. This formatter displays a summary for each declaration, with links to the details if there is one. All of this is controlled by the ASTFormatters.

__init__(*self*, *page*) [operation on Summary]

_init_default_formatters(*self*) [operation on Summary]

set_link_detail(*self*, *boolean*) [operation on Summary]
 Sets link_detail flag to given value.

label(*self*, *ref*, *label*) [operation on Summary]
 Override to check link_detail flag. If it's set, returns a reference instead - which will be to the detailed info

writeSectionStart(*self*, *heading*) [operation on Summary]
 Starts a table entity. The heading is placed in a row in a td with the class 'heading'.

writeSectionEnd(*self*, *heading*) [operation on Summary]
 Closes the table entity and adds a break.

writeSectionItem(*self*, *text*) [operation on Summary]
 Adds a table row

process(*self*, *decl*) [operation on Summary]
 Print out the summaries from the given decl

Detail [class]

parents: parent class

__init__(*self*, *page*) [operation on Detail]

_init_default_formatters(*self*) [operation on Detail]

writeSectionStart(*self*, *heading*) [operation on Detail]
 Creates a table with one row. The row has a td of class 'heading' containing the heading string

writeSectionItem(*self*, *text*) [operation on Detail]
 Writes text and follows with a horizontal rule

process(*self*, *decl*) [operation on Detail]
 Print out the details for the children of the given decl

Inheritance [class]

parents: parent class

__init__(*self*, *page*) [operation on Inheritance]

_init_default_formatters(*self*) [operation on Inheritance]

process(*self*, *decl*) [operation on Inheritance]
 Walk the hierarchy to find inherited members to print.

_process_class(*self*, *clas*, *names*) [operation on Inheritance]
 Prints info for the given class, and calls `_process_superclasses` after

_short_name(*self*, *decl*) [operation on Inheritance]

_process_superclasses(*self*, *clas*, *names*) [operation on Inheritance]
 Iterates through the superclasses of `clas` and calls `_process_clas` for each

writeSectionStart(*self*, *heading*) [operation on Inheritance]
 Creates a table with one row. The row has a td of class 'heading' containing the heading string

writeSectionItem(*self*, *text*) [operation on Inheritance]
 Adds a table row

writeSectionEnd(*self*, *heading*) [operation on Inheritance]
 Closes the table entity and adds a break.

CommentFormatter [module]

CommentParser, CommentFormatter and derivatives.

CommentFormatter [class]

A class that takes a Declaration and formats its comments into a string.

__init__(*self*) [operation on CommentFormatter]

format(*self*, *page*, *decl*) [operation on CommentFormatter]
 Formats the first comment of the given AST.Declaration. Note that the Linker.Comments.Summarizer CommentProcessor is supposed to have combined all comments first in the Linker stage.

format_summary(*self*, *page*, *decl*) [operation on CommentFormatter]

Formats the summary of the first comment of the given AST.Declaration. Note that the Linker.Comments.Summarizer CommentProcessor is supposed to have combined all comments first in the Linker stage.

CommentFormatterStrategy [class]

Interface class that takes a comment and formats its summary and/or detail strings.

format(*self*, *page*, *decl*, *text*) [operation on CommentFormatterStrategy]

Format the given comment

format_summary(*self*, *page*, *decl*, *summary*) [operation on CommentFormatterStrategy]

Format the given comment summary

QuoteHTML [class]

parents: parent class A formatter that quotes HTML characters like the angle brackets and the ampersand. Formats both text and summary.

format(*self*, *page*, *decl*, *text*) [operation on QuoteHTML]
 Replace angle brackets with HTML codes

format_summary(*self*, *page*, *decl*, *text*) [operation on QuoteHTML]

Replace angle brackets with HTML codes

JavadocFormatter [class]

parents: parent class A formatter that formats comments similar to Javadoc @tags

__init__(*self*) [operation on JavadocFormatter]
Create regex objects for regexps

extract(*self*, *regexp*, *str*) [operation on JavadocFormatter]
Extracts all matches of the regexp from the text. The MatchObjects are returned in a list

format(*self*, *page*, *decl*, *text*) [operation on JavadocFormatter]
Format any @tags in the text, and any @tags stored by the JavaTags CommentProcessor in the Linker stage.

format_inline_see(*self*, *page*, *decl*, *text*) [operation on JavadocFormatter]
Formats inline tags in the text

format_params(*self*, *param_tags*) [operation on JavadocFormatter]
Formats a list of (param, description) tags

format_attrs(*self*, *attr_tags*) [operation on JavadocFormatter]
Formats a list of (attr, description) tags

format_return(*self*, *return_tag*) [operation on JavadocFormatter]
Formats a since description string

format_see(*self*, *page*, *see_tags*, *decl*) [operation on JavadocFormatter]
Formats a list of (ref,description) tags

find_link(*self*, *page*, *ref*, *decl*) [operation on JavadocFormatter]
Given a "reference" and a declaration, returns a HTML link. Various methods are tried to resolve the reference. First the parameters are taken off, then we try to split the ref using '.' or ':.'. The params are added back, and then we try to match this scoped name against the current scope. If that fails, then we recursively try enclosing scopes.

_find_link_at(*self*, *ref*, *scope*) [operation on JavadocFormatter]

find_method_entry(*self*, *name*, *scope*) [operation on JavadocFormatter]

Tries to find a TOC entry for a method adjacent to decl. The enclosing scope is found using the types dictionary, and the realname()'s of all the functions compared to ref.

QtDocFormatter [class]

parents: parent class A formatter that uses Qt-style doc tags.

__init__(*self*) [operation on QtDocFormatter]

parseText(*self*, *str*, *decl*) [operation on QtDocFormatter]

format_see(*self*, *see_tags*, *decl*) [operation on QtDocFormatter]
 Formats a list of (ref,description) tags

SectionFormatter [class]

parents: parent class A test formatter

__init__(*self*) [operation on SectionFormatter]

format(*self*, *page*, *decl*, *text*) [operation on SectionFormatter]

core [module]

Core module for the HTML Formatter.

This module is the first to be loaded, and it creates the global 'core.config' object before creating any pages. It also handles the command line parsing for this module, and coordinates the actual output generation.

sort(*list*) [function]
 Utility func to sort and return the given list

old_reference(*name*, *scope*, *label*, *keys*)** [function]
 Utility method to insert a reference to a name.

compile_glob(*globstr*) [function]
 Returns a compiled regular expression for the given glob string. A glob string is something like "*.?pp" which gets translated into "^.*\..pp\$".

usage() [function]
 Print usage to stdout

__parseArgs(*args*, *config_obj*) [function]

format(*args*, *ast*, *config_obj*) [function]

configure_for_gui(*ast*, *config_obj*) [function]

Config [class]

Central configuration repository for HTML formatter. This class holds references to the current formatters, tocs, etc.

__init__(*self*) [operation on Config]
 Constructor - initialise objects to None.

fillDefaults(*self*) [operation on Config]
 Fill in empty options with defaults

use_config(*self*, *obj*) [operation on Config]
 Extracts useful attributes from 'obj' and stores them. The object itself is also stored as config.obj

_config_pages(*self*, *pages*) [operation on Config]
 Configures from the given list of pages

_config_sorter(*self*, *sorter*) [operation on Config]

_config_datadir(*self*, *datadir*) [operation on Config]

_config_output_dir(*self*, *output_dir*) [operation on Config]

_config_stylesheet(*self*, *stylesheet*) [operation on Config]

_config_stylesheet_file(*self*,
stylesheet_file) [operation on Config]

_config_comment_formatters(*self*,
comment_formatters) [operation on Config]

_config_toc_in(*self*, *toc_in*) [operation on Config]

_config_default_toc(*self*, *page*) [operation on Config]

_config_toc_out(*self*, *toc_out*) [operation on Config]

_config_tree_formatter(*self*, *tree_class*) [operation on Config]

_config_file_layout(*self*, *layout*) [operation on Config]

_config_base_dir(*self*, *dir*) [operation on Config]

_config_start_dir(*self*, *dir*) [operation on Config]

`_config_structs_as_classes(self, yesno)` [operation on `Config`]

`_config_exclude_globs(self, globs)` [operation on `Config`]

`_config_page_format(self, page_format)` [operation on `Config`]

`set_contents_page(self, page)` [operation on `Config`]

Call this method to set the contents page. First come first served – whatever module the user puts first in the list that sets this is it. This is the frame in the top-left if you use the default frameset.

`set_index_page(self, page)` [operation on `Config`]

Call this method to set the index page. First come first served – whatever module the user puts first in the list that sets this is it. This is the frame on the left if you use the default frameset.

`set_main_page(self, page)` [operation on `Config`]

Call this method to set the main index.html page. First come first served – whatever module the user puts first in the list that sets this is it.

`set_using_module_index(self)` [operation on `Config`]

Sets the `using_module_index` flag. This will cause the an intermediate level of links intended to go in the left frame.

Struct [class]

Dummy class. Initialise with keyword args.

`__init__(self, **keys)` [operation on `Struct`]

DeclStyle [class]

This class just maintains a mapping from declaration to display style. The style is an enumeration, possible values being: `SUMMARY` (only display a summary for this declaration), `DETAIL` (summary and detailed info), `INLINE` (summary and detailed info, where detailed info is an inline version of the declaration even if it's a class, etc.)

Upon initiation, an instance of this class installs itself in the config object as `"decl_style"`.

`__init__(self)` [operation on `DeclStyle`]

`style_of(self, decl)` [operation on `DeclStyle`]

Returns the style of the given decl

PageManager [class]

This class manages and coordinates the various pages. The user adds pages by passing their class object to the `addPage` method. Pages should be derived from `Page.Page`, and their constructors may want to call the `addRootPage` method of the `PageManager` object to register a name and link that is listed along with other root or top-level pages.

`__init__(self)` [operation on PageManager]

`getPage(self, name)` [operation on PageManager]
Returns the Page with the given name

`globalScope(self)` [operation on PageManager]
Return the global scope

`calculateStart(self, root, namespace)` [operation on PageManager]
Calculates the start scope using the 'namespace' config var

`addPage(self, pageClass)` [operation on PageManager]
Add a page of the given class. An instance is created and stored, and its `root()` method is called and the name,link tuple stored if None isn't returned.

`addRootPage(self, file, label, target, visibility)` [operation on PageManager]
Adds a named link to the list of root pages. Called from the constructors of Page objects. The root pages are displayed at the top of every page, depending on their visibility (higher = more visible).

`formatHeader(self, origin, visibility)` [operation on PageManager]
Formats the list of root pages to HTML. The origin specifies the generated page itself (which shouldn't be linked), such that the relative links can be generated. Only root pages of 'visibility' or above are included.

`process(self, root)` [operation on PageManager]
Create all pages from the start Scope, derived from the root Scope

`_loadPages(self)` [operation on PageManager]
Loads the page objects from the `config.pages` list. Each element is either a string or a tuple of two strings. One string means load the named module and look for a 'htmlPageClass' attribute in it. A tuple of two strings means load the module from the first string, and look for an attribute using the second string.

register_filename(*self*, *filename*, *page*, *scope*) [operation on PageManager]

Registers a file for later production. The first page to register the filename gets to keep it.

filename_info(*self*, *filename*) [operation on PageManager]

Returns information about a registered file, as a (page,scope) pair. Will return None if the filename isn't registered.

DirBrowse [module]

DirBrowse [class]

parents: parent class A page that shows the entire contents of directories, in a form similar to LXR.

__init__(*self*, *manager*) [operation on DirBrowse]

filename(*self*) [operation on DirBrowse]

since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on DirBrowse]

since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

filename_for_dir(*self*, *dir*) [operation on DirBrowse]

Returns the output filename for the given input directory

register(*self*) [operation on DirBrowse]

Registers a page for each file in the hierarchy

register_filenames(*self*, *start*) [operation on DirBrowse]

Registers a page for every directory

process(*self*, *start*) [operation on DirBrowse]

Recursively visit each directory below the base path given in the config.

process_dir(*self*, *path*) [operation on DirBrowse]

Process a directory, producing an output page for it

doxygen [module]

Doxygen emulation mode classes.

DOScopeSorter [class]

parents: parent class

| | |
|--|-----------------------------------|
| <code>_section_of(self, decl)</code> | [operation on DScopeSorter] |
| <code>sort_section_names(self)</code> Sort by a known order.. | [operation on DScopeSorter] |
| DOSummaryAST parents: parent class | [class] |
| <code>formatEnum(self, decl)</code> (enum, enum { list of enumerator names }) | [operation on DOSummaryAST] |
| DOSummaryCommenter parents: parent class Adds summary comments to all declarations | [class] |
| <code>formatDeclaration(self, decl)</code> | [operation on DOSummaryCommenter] |
| DODetailAST parents: parent class | [class] |
| <code>formatFunction(self, decl)</code> | [operation on DODetailAST] |
| PreDivFormatter parents: parent class | [class] |
| <code>formatDeclaration(self, decl)</code> | [operation on PreDivFormatter] |
| PostDivFormatter parents: parent class | [class] |
| <code>formatDeclaration(self, decl)</code> | [operation on PostDivFormatter] |
| PreSummaryDiv parents: parent class | [class] |
| <code>formatDeclaration(self, decl)</code> | [operation on PreSummaryDiv] |
| PostSummaryDiv parents: parent class | [class] |
| <code>formatDeclaration(self, decl)</code> | [operation on PostSummaryDiv] |
| DOSummary parents: parent class | [class] |
| <code>old_init_formatters(self)</code> | [operation on DOSummary] |

write_start(*self*) [operation on DOSummary]

writeSectionStart(*self*, *heading*) [operation on DOSummary]

writeSectionEnd(*self*, *heading*) [operation on DOSummary]

write_end(*self*) [operation on DOSummary]
Closes the table entity and adds a break.

writeSectionItem_Foo(*self*, *type*, *name*) [operation on DOSummary]
Adds a table row with one or two data elements. If *type* is None then there is only one td with a colspan of 2.

DODetail [class]
parents: parent class

old_init_formatters(*self*) [operation on DODetail]

writeSectionEnd(*self*, *heading*) [operation on DODetail]

writeSectionItem_Foo(*self*, *text*) [operation on DODetail]
Joins *text1* and *text2*

FileDetails [module]

FileDetails [class]
parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

__init__(*self*, *manager*) [operation on FileDetails]

filename(*self*) [operation on FileDetails]
since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on FileDetails]
since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

register_filenames(*self*, *start*) [operation on FileDetails]
Registers a page for each file indexed

process(*self*, *start*) [operation on FileDetails]
Creates a page for each file using `process_scope`

process_scope(*self*, *filename*, *file*) [operation on FileDetails]
 Creates a page for the given file. The page is just an index, containing a list of declarations.

FileIndexer [module]

FileIndexer [class]

parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

__init__(*self*, *manager*) [operation on FileIndexer]

filename(*self*) [operation on FileIndexer]
 since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on FileIndexer]
 since FileTree generates a whole file hierarchy, this method returns the current title, which may change over the lifetime of this object

register_filenames(*self*, *start*) [operation on FileIndexer]
 Registers a page for each file indexed

process(*self*, *start*) [operation on FileIndexer]
 Creates a page for each file using process_scope

process_scope(*self*, *filename*, *file*) [operation on FileIndexer]
 Creates a page for the given file. The page is just an index, containing a list of declarations.

FileLayout [module]

FileLayout module. This is the base class for the FileLayout interface. The default implementation stores everything in the same directory.

FileLayout [class]

parents: parent class Base class for naming files. You may derive from this class and reimplement any methods you like. The default implementation stores everything in the same directory (specified by -o).

__init__(*self*) [operation on FileLayout]

copyFile(*self*, *orig_name*, *new_name*) [operation on FileLayout]
 Copies file if newer. The file named by orig_name is compared to new_name, and if newer or new_name doesn't exist, it is copied.

- `_checkMain(self, filename)`** [operation on FileLayout]
 Checks whether the given filename is the main index page or not. If it is, then it returns the filename from `nameOfIndex()`, else it returns it unchanged
- `nameOfScope(self, scope)`** [operation on FileLayout]
 Return the filename of a scoped name (class or module). The default implementation is to join the names with '-' and append ".html" Additionally, special characters are Util.quoted in URL-style
- `_stripFilename(self, file)`** [operation on FileLayout]
- `nameOfFileIndex(self, file)`** [operation on FileLayout]
 Return the filename for the index of an input file. The `base_dir` config option is used. Default implementation is to join the path with '-', prepend "_file-" and append ".html"
- `nameOfFileSource(self, file)`** [operation on FileLayout]
 Return the filename for the source of an input file. The `base_dir` config option is used. Default implementation is to join the path with '-', prepend "_source-" and append ".html"
- `nameOfFileDetails(self, file)`** [operation on FileLayout]
 Return the filename for the details of an input file. The `base_dir` config option is used. Default implementation is to join the path with '-', prepend "_filedetail-" and append ".html"
- `nameOfIndex(self)`** [operation on FileLayout]
 Return the name of the main index file. Default is `index.html`
- `nameOfSpecial(self, name)`** [operation on FileLayout]
 Return the name of a special file (tree, etc). Default is `_name.html`
- `nameOfScopedSpecial(self, name, scope, ext)`** [operation on FileLayout]
 Return the name of a special type of scope file. Default is to join the scope with '-' and prepend '-' + name
- `nameOfModuleTree(self)`** [operation on FileLayout]
 Return the name of the module tree index. Default is `_modules.html`
- `nameOfModuleIndex(self, scope)`** [operation on FileLayout]
 Return the name of the index of the given module. Default is to join the name with '-', prepend "_module_" and append ".html"

link(*self*, *decl*) [operation on FileLayout]
 Create a link to the named declaration. This method may have to deal with the directory layout.

NestedFileLayout [class]
 parents: parent class generates a structured file system instead of a flat one

__init__(*self*) [operation on NestedFileLayout]

nameOfScope(*self*, *scope*) [operation on NestedFileLayout]
 Return the filename of a scoped name (class or module). One subdirectory per scope

nameOfFileIndex(*self*, *file*) [operation on NestedFileLayout]
 Return the filename for the index of an input file. The `base_dir` config option is used. Default implementation is to join the path with '-', prepend "_file-" and append ".html"

nameOfFileSource(*self*, *file*) [operation on NestedFileLayout]
 Return the filename for the source of an input file. The `base_dir` config option is used. Default implementation is to join the path with '-', prepend "_source-" and append ".html"

nameOfFileDetails(*self*, *file*) [operation on NestedFileLayout]
 Return the filename for the details of an input file. The `base_dir` config option is used. Returns the filename nested in the FileDetails directory and with .html appended.

nameOfIndex(*self*) [operation on NestedFileLayout]
 Return the name of the main index file.

nameOfSpecial(*self*, *name*) [operation on NestedFileLayout]
 Return the name of a special file (tree, etc).

nameOfScopedSpecial(*self*, *name*, *scope*, *ext*) [operation on NestedFileLayout]
 Return the name of a special type of scope file

nameOfModuleTree(*self*) [operation on NestedFileLayout]
 Return the name of the module tree index

nameOfModuleIndex(*self*, *scope*) [operation on NestedFileLayout]
 Return the name of the index of the given module

FileListing [module]**FileListing** [class]

parents: parent class A page that creates an index of files, and an index for each file. First the index of files is created, intended for the top-left frame. Second a page is created for each file, listing the major declarations for that file, eg: classes, global functions, namespaces, etc.

__init__(*self*, *manager*) [operation on FileListing]

filename(*self*) [operation on FileListing]
Returns the filename

title(*self*) [operation on FileListing]
Returns the title

register(*self*) [operation on FileListing]
Registers this page for the top-left frame

register_filenames(*self*, *start*) [operation on FileListing]
Registers a page for each file indexed

process(*self*, *start*) [operation on FileListing]
Creates the listing using the recursive processFileTreeNode method

_node_sorter(*self*, *a*, *b*) [operation on FileListing]
Compares file nodes a and b depending on whether they are leaves or not

processFileTreeNode(*self*, *node*) [operation on FileListing]
Creates a portion of the tree for the given file node. This method assumes that the file is already in progress, and just appends to it. This method is recursive, calling itself for each child of node (file or directory).

FileSource [module]**FileSource** [class]

parents: parent class A module for creating a page for each file with hyperlinked source

__init__(*self*, *manager*) [operation on FileSource]

filename(*self*) [operation on FileSource]
since FileSource generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

| | |
|---|---------------------------|
| title(<i>self</i>) | [operation on FileSource] |
| since FileSource generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object | |
| process(<i>self</i>, <i>start</i>) | [operation on FileSource] |
| Creates a page for every file | |
| register_filenames(<i>self</i>, <i>start</i>) | [operation on FileSource] |
| Registers a page for every source file | |
| process_node(<i>self</i>, <i>file</i>) | [operation on FileSource] |
| Creates a page for the given file | |

FileTreeJS [module]

FileTree [class]

parents: parent class

__init__(*self*, *manager*) [operation on FileTree]

filename(*self*) [operation on FileTree]
since FileTree generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on FileTree]
since FileTree generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object

process(*self*, *start*) [operation on FileTree]

_node_sorter(*self*, *a*, *b*) [operation on FileTree]

processFileTreeNode(*self*, *node*) [operation on FileTree]

processFileTreeNodePage(*self*, *node*) [operation on FileTree]

FileTree [module]

FileTree [class]

parents: parent class A page which wraps the three pages now used for creating file-related info in the doco. For backwards compatibility only. The three new pages which should be used are FileListing, FileIndexer and FileDetails.

__init__(*self*, *manager*) [operation on FileTree]

register(*self*) [operation on FileTree]

register_filenames(*self*, *start*) [operation on FileTree]

process(*self*, *start*) [operation on FileTree]

FormatStrategy [module]

AST Formatting Strategies.

This module contains all the builtin formatting strategies used to make the main scope pages. These strategies are used by Strategy.Summary/Detail to create the entries for each declaration - each has a list of strategies that it calls in turn to generate a HTML fragment for each declaration. So for example, a link to source code is added just by adding a Strategy (eg: FilePages) that outputs the link to the source. This can be done without changing the strategies that generate the declaration name, type or comments.

Strategy [class]

Generates HTML fragment for a declaration. Multiple strategies are combined to generate the output for a single declaration, allowing the user to customise the output by choosing a set of strategies. This follows the Strategy design pattern.

The key concept of this class is the `format*` methods. Any class derived from Strategy that overrides one of the format methods will have that method called by the Summary and Detail formatters when they visit that AST type. Summary and Detail maintain a list of Strategies, and a list for each AST type.

For example, when Strategy.Summary visits a Function object, it calls the `formatFunction` method on all Strategies registered with SummaryFormatter that implemented that method. Each of these format methods returns a string, which may contain a TD tag to create a new column.

An important point to note is that only Strategies which override a particular format method are called - if that format method is not overridden then it is not called for that declaration type.

__init__(*self*, *formatter*) [operation on Strategy]

Store formatter as `self.formatter`. The formatter is either a SummaryFormatter or DetailFormatter, and is used for things like `reference()` and `label()` calls. Local references to the formatter's `reference` and `label` methods are stored in `self` for more efficient use of them.

formatModifiers(*self*, *modifiers*) [operation on Strategy]

Returns a HTML string from the given list of string modifiers. The modifiers are enclosed in 'keyword' spans.

formatDeclaration(*self*, *decl*) [operation on Strategy]

formatForward(*self*, *decl*) [operation on Strategy]

| | |
|---|-------------------------|
| formatGroup (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatScope (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatModule (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatMetaModule (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatClass (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatTypedef (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatEnum (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatVariable (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatConst (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatFunction (<i>self</i> , <i>decl</i>) | [operation on Strategy] |
| formatOperation (<i>self</i> , <i>decl</i>) | [operation on Strategy] |

BaseAST [class]

parents: parent class Base class for SummaryAST and DetailAST.

The two classes SummaryAST and DetailAST are actually very similar in operation, and so most of their methods are defined here. Both of them print out the definition of the declarations, including type, parameters, etc. Some things such as exception specifications are only printed out in the detailed version.

| | |
|--|------------------------|
| formatParameters (<i>self</i> , <i>parameters</i>) | [operation on BaseAST] |
| Returns formatted string for the given parameter list | |
| formatDeclaration (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| The default is to return no type and just the declarations name for the name | |
| formatForward (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatGroup (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatScope (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| Scopes have their own pages, so return a reference to it | |
| formatModule (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatMetaModule (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |

| | |
|--|------------------------|
| formatClass (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatTypedef (<i>self</i> , <i>decl</i>) (typedef type, typedef name) | [operation on BaseAST] |
| formatEnumerator (<i>self</i> , <i>decl</i>) This is only called by formatEnum | [operation on BaseAST] |
| formatEnum (<i>self</i> , <i>decl</i>) (enum name, list of enumerator names) | [operation on BaseAST] |
| formatVariable (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatConst (<i>self</i> , <i>decl</i>) (const type, const name = const value) | [operation on BaseAST] |
| formatFunction (<i>self</i> , <i>decl</i>) (return type, func + params + formatFunctionExceptions) | [operation on BaseAST] |
| formatOperation (<i>self</i> , <i>decl</i>) | [operation on BaseAST] |
| formatParameter (<i>self</i> , <i>parameter</i>) Returns one string for the given parameter | [operation on BaseAST] |

SummaryAST [class]

parents: parent class Derives from BaseStrategy to provide summary-specific methods. Currently the only one is formatOperationExceptions

| | |
|---|---------------------------|
| formatOperationExceptions (<i>self</i> , <i>oper</i>) | [operation on SummaryAST] |
| Returns a reference to the detail if there are any exceptions. | |

Default [class]

parents: parent class A base AST strategy that calls formatDeclaration for all types

| | |
|---|------------------------|
| formatForward (<i>self</i> , <i>decl</i>) | [operation on Default] |
| formatGroup (<i>self</i> , <i>decl</i>) | [operation on Default] |
| formatScope (<i>self</i> , <i>decl</i>) | [operation on Default] |
| formatModule (<i>self</i> , <i>decl</i>) | [operation on Default] |
| formatMetaModule (<i>self</i> , <i>decl</i>) | [operation on Default] |

formatClass(*self*, *decl*) [operation on Default]

formatTypedef(*self*, *decl*) [operation on Default]

formatEnum(*self*, *decl*) [operation on Default]

formatVariable(*self*, *decl*) [operation on Default]

formatConst(*self*, *decl*) [operation on Default]

formatFunction(*self*, *decl*) [operation on Default]

formatOperation(*self*, *decl*) [operation on Default]

SummaryCommenter [class]

parents: parent class Adds summary comments to all declarations

formatDeclaration(*self*, *decl*) [operation on SummaryCommenter]

formatGroup(*self*, *decl*) [operation on SummaryCommenter]
 Override for group to use the div version of commenting, and no

 before

SourceLinker [class]

parents: parent class Adds a link to the decl on the file page to all declarations

formatDeclaration(*self*, *decl*) [operation on SourceLinker]

XRefLinker [class]

parents: parent class Adds an xref link to all declarations

__init__(*self*, *formatter*) [operation on XRefLinker]

formatDeclaration(*self*, *decl*) [operation on XRefLinker]

Heading [class]

parents: parent class Formats the top of a page - it is passed only the Declaration that the page is for (a Module or Class).

formatName(*self*, *scoped_name*) [operation on Heading]
 Formats a reference to each parent scope

formatNameInNamespace(*self*, *scoped_name*) [operation on Heading]
 Formats a reference to each parent scope, starting at the first non-module
 scope

formatNamespaceOfName(*self*, *scoped_name*) [operation on Heading]

Formats a reference to each parent scope and this one

formatModule(*self*, *module*) [operation on Heading]

Formats the module by linking to each parent scope in the name

formatMetaModule(*self*, *module*) [operation on Heading]

Calls formatModule

formatClass(*self*, *clas*) [operation on Heading]

Formats the class by linking to each parent scope in the name

formatParameter(*self*, *parameter*) [operation on Heading]

Returns one string for the given parameter

DetailAST [class]

parents: parent class Derives BaseAST to provide detail-specific AST formatting.

formatOperationExceptions(*self*, *oper*) [operation on DetailAST]

Prints out the full exception spec

formatEnum(*self*, *enum*) [operation on DetailAST]

formatEnumerator(*self*, *enumerator*) [operation on DetailAST]

DetailCommenter [class]

parents: parent class Adds summary comments to all declarations

formatDeclaration(*self*, *decl*) [operation on DetailCommenter]

ClassHierarchySimple [class]

parents: parent class Prints a simple text hierarchy for classes

formatInheritance(*self*, *inheritance*) [operation on ClassHierarchySimple]

formatClass(*self*, *clas*) [operation on ClassHierarchySimple]

ClassHierarchyGraph [class]

parents: parent class Prints a graphical hierarchy for classes, using the Dot formatter.

formatClass(*self*, *clas*) [operation on ClassHierarchyGraph]

Inheritance [class]
 parents: parent class Prints just the name of each declaration, with a link to its doc

formatDeclaration(*self*, *decl*, *label*) [operation on Inheritance]

formatFunction(*self*, *decl*) [operation on Inheritance]

formatOperation(*self*, *decl*) [operation on Inheritance]

FramesIndex [module]

FramesIndex [class]
 parents: parent class A class that creates an index with frames

__init__(*self*, *manager*) [operation on FramesIndex]

filename(*self*) [operation on FramesIndex]

title(*self*) [operation on FramesIndex]

register(*self*) [operation on FramesIndex]

process(*self*, *start*) [operation on FramesIndex]
 Creates a frames index file

InheritanceGraph [module]

find_common_name(*graph*) [function]

ToDecl [class]
 parents: parent class

__call__(*self*, *name*) [operation on ToDecl]

visitBaseType(*self*, *type*) [operation on ToDecl]

visitUnknown(*self*, *type*) [operation on ToDecl]

visitDeclared(*self*, *type*) [operation on ToDecl]

visitModifier(*self*, *type*) [operation on ToDecl]

visitArray(*self*, *type*) [operation on ToDecl]

visitTemplate(*self*, *type*) [operation on ToDecl]

visitParametrized(*self*, *type*) [operation on ToDecl]

visitFunctionType(*self*, *type*) [operation on ToDecl]

InheritanceGraph [class]

parents: parent class

__init__(*self*, *manager*) [operation on InheritanceGraph]

filename(*self*) [operation on InheritanceGraph]

title(*self*) [operation on InheritanceGraph]

register(*self*) [operation on InheritanceGraph]
Registers this page with the manager

consolidate(*self*, *graphs*) [operation on InheritanceGraph]
Consolidates small graphs into larger ones

process(*self*, *start*) [operation on InheritanceGraph]
Creates a file with the inheritance graph

InheritanceTree [module]

InheritanceTree [class]

parents: parent class

__init__(*self*, *manager*) [operation on InheritanceTree]

filename(*self*) [operation on InheritanceTree]

title(*self*) [operation on InheritanceTree]

register(*self*) [operation on InheritanceTree]

process(*self*, *start*) [operation on InheritanceTree]
Creates a file with the inheritance tree

processClassInheritance(*self*, *args*) [operation on InheritanceTree]

JSTree [module]

JSTree [class]

parents: parent class Page that makes Javascript trees. The trees have expanding and collapsing nodes. call `js_init()` with the button images and default open/close policy during process

| | |
|---|-----------------------|
| <code>__init__(self, manager)</code> | [operation on JSTree] |
| <code>getId(self)</code> | [operation on JSTree] |
| <code>js_init(self, open_img, close_img, leaf_img, base, default_open)</code> | [operation on JSTree] |
| Initialise the JSTree page. This method copies the files to the output directory and stores the values given. | |
| <code>start_file(self)</code> | [operation on JSTree] |
| Overrides start_file to add the javascript | |
| <code>formatImage(self, id, filename, alt_text)</code> | [operation on JSTree] |
| Returns the image element for the given image | |
| <code>writeLeaf(self, item_text)</code> | [operation on JSTree] |
| Write a leaf node to the output at the current tree level. | |
| <code>writeNodeStart(self, item_text)</code> | [operation on JSTree] |
| Write a non-leaf node to the output at the current tree level, and start a new level. | |
| <code>writeNodeEnd(self)</code> | [operation on JSTree] |
| Finish a non-leaf node, and close the current tree level. | |

ModuleIndexer [module]

ModuleIndexer [class]

parents: parent class A module for indexing AST.Modules. Each module gets its own page with a list of nested scope declarations with comments. It is intended to go in the left frame...

| | |
|---------------------------------------|------------------------------|
| <code>__init__(self, manager)</code> | [operation on ModuleIndexer] |
| <code>filename(self)</code> | [operation on ModuleIndexer] |
| <code>title(self)</code> | [operation on ModuleIndexer] |
| <code>register(self)</code> | [operation on ModuleIndexer] |
| Register first page as index page | |
| <code>process(self, start)</code> | [operation on ModuleIndexer] |
| Creates indexes for all modules | |

`_makePageHeading(self, ns)` [operation on `ModuleIndexer`]
 Creates a HTML fragment which becomes the name at the top of the index page. This may be overridden, but the default is (now) to make a linked fully scoped name, where each scope has a link to the relevant index.

`processNamespaceIndex(self, ns)` [operation on `ModuleIndexer`]
 Index one module

`ModuleListingJS` [module]

`ModuleListingJS` [class]

parents: parent class Create an index of all modules with JS. The JS allows the user to expand/collapse sections of the tree!

`__init__(self, manager)` [operation on `ModuleListingJS`]

`_init_page(self)` [operation on `ModuleListingJS`]
 Sets `_filename` and registers the page with the manager

`filename(self)` [operation on `ModuleListingJS`]

`title(self)` [operation on `ModuleListingJS`]

`process(self, start)` [operation on `ModuleListingJS`]
 Create a page with an index of all modules

`_child_filter(self, child)` [operation on `ModuleListingJS`]

`_link_href(self, ns)` [operation on `ModuleListingJS`]

`get_children(self, decl)` [operation on `ModuleListingJS`]

`indexModule(self, ns, rel_scope)` [operation on `ModuleListingJS`]
 Write a link for this module and recursively visit child modules.

`ModuleListing` [module]

`ModuleListing` [class]

parents: parent class Create an index of all modules with JS. The JS allows the user to expand/collapse sections of the tree!

`__init__(self, manager)` [operation on `ModuleListing`]

| | |
|--|---|
| filename(<i>self</i>) | [operation on ModuleListing] |
| title(<i>self</i>) | [operation on ModuleListing] |
| register(<i>self</i>) | [operation on ModuleListing] registers the page with the manager for the 'contents' (top left) frame |
| process(<i>self</i>, <i>start</i>) | [operation on ModuleListing] Create a page with an index of all modules |
| _child_filter(<i>self</i>, <i>child</i>) | [operation on ModuleListing] Returns true if the given child declaration is to be included |
| _link_href(<i>self</i>, <i>ns</i>) | [operation on ModuleListing] Returns the link to the given declaration |
| _get_children(<i>self</i>, <i>decl</i>) | [operation on ModuleListing] Returns the children of the given declaration |
| indexModule(<i>self</i>, <i>ns</i>, <i>rel_scope</i>) | [operation on ModuleListing] Write a link for this module and recursively visit child modules. |

NameIndex [module]

NameIndex [class]

parents: parent class Creates an index of all names on one page in alphabetical order

| | |
|--|--|
| __init__(<i>self</i>, <i>manager</i>) | [operation on NameIndex] |
| filename(<i>self</i>) | [operation on NameIndex] |
| title(<i>self</i>) | [operation on NameIndex] |
| register(<i>self</i>) | [operation on NameIndex] |
| process(<i>self</i>, <i>start</i>) | [operation on NameIndex] Creates the page. It is created as a list of tables, one for each letter. The tables have a number of columns, which is 2 by default. <code>_processItem</code> is called for each item in the dictionary. |
| _makeDict(<i>self</i>) | [operation on NameIndex] Returns a dictionary of items. The keys of the dictionary are the headings - the first letter of the name. The values are each a sorted list of items with that first letter. |

`_processItem(self, type)` [operation on `NameIndex`]
 Process the given name for output

Page [module]
 Page base class, contains base functionality and common interface for all Pages.

PageFormat [class]
 Default and base class for formatting a page layout. The PageFormat class basically defines the HTML used at the start and end of the page. The default creates an XHTML compliant header and footer with a proper title, and link to the stylesheet.

`__init__(self)` [operation on `PageFormat`]

`set_prefix(self, prefix)` [operation on `PageFormat`]
 Sets the prefix to use to correctly reference files in the document root directory.

`stylesheet(self)` [operation on `PageFormat`]
 Returns the relative filename of the stylesheet to use. The stylesheet specified in the user's config is copied into the output directory. If this page is not in the same directory, the url returned from this function will have the appropriate number of '..'s added.

`prefix(self)` [operation on `PageFormat`]
 Returns the prefix to use to correctly reference files in the document root directory. This will only ever not be "" if you are using the NestedFileLayout, in which case it will be "" or '../' or '../..' etc as appropriate.

`page_header(self, os, title, body, headextra)` [operation on `PageFormat`]
 Called to output the page header to the given output stream.

`page_footer(self, os, body)` [operation on `PageFormat`]
 Called to output the page footer to the given output stream.

TemplatePageFormat [class]
 parents: parent class PageFormat subclass that uses a template file to define the HTML header and footer for each page.

`__init__(self)` [operation on `TemplatePageFormat`]

`load_file(self)` [operation on `TemplatePageFormat`]
 Loads and parses the template file

write(*self*, *os*, *text*) [operation on `TemplatePageFormat`]
Writes the text to the output stream, replacing @PREFIX@ with the prefix for this file

page_header(*self*, *os*, *title*, *body*, *headextra*) [operation on `TemplatePageFormat`]
Formats the header using the template file

page_footer(*self*, *os*, *body*) [operation on `TemplatePageFormat`]
Formats the footer using the template file

Page [class]

Base class for a Page. The base class provides a common interface, and also handles common operations such as opening the file, and delegating the page formatting to a strategy class.

__init__(*self*, *manager*) [operation on `Page`]
Constructor, loads the formatting class.

filename(*self*) [operation on `Page`]
Polymorphic method returning the filename associated with the page

title(*self*) [operation on `Page`]
Polymorphic method returning the title associated with the page

os(*self*) [operation on `Page`]
Returns the output stream opened with `start_file`

write(*self*, *str*) [operation on `Page`]
Writes the given string to the currently opened file

register(*self*) [operation on `Page`]
Registers this Page class with the PageManager. This method is abstract - derived Pages should implement it to call the appropriate methods in PageManager if they need to. This method is called after construction.

register_filenames(*self*, *start*) [operation on `Page`]
Registers filenames for each file this Page will generate, given the starting Scope.

get_toc(*self*, *start*) [operation on `Page`]
Retrieves the TOC for this page. This method assumes that the page generates info for the the whole AST, which could be the ScopePages, the FilePages (source code) or the XRefPages (cross reference info). The default implementation returns None. Start is the declaration to start processing from, which could be the global namespace.

process(*self*, *start*) [operation on Page]
 Process the given Scope recursively. This is the method which is called to actually create the files, so you probably want to override it ;)

process_scope(*self*, *scope*) [operation on Page]
 Process just the given scope

open_file(*self*) [operation on Page]
 Returns a new output stream. This template method is for internal use only, but may be overridden in derived classes. The default joins `config.basename` and `self.filename()` and uses `Util.open()`

close_file(*self*) [operation on Page]
 Closes the internal output stream. This template method is for internal use only, but may be overridden in derived classes.

start_file(*self*, *body*, *headextra*) [operation on Page]
 Start a new file with given filename, title and body. This method opens a file for writing, and writes the html header crap at the top. You must specify a title, which is prepended with the project name. The body argument is optional, and it is preferred to use stylesheets for that sort of stuff. You may want to put an `onLoad` handler in it though in which case that's the place to do it. The opened file is stored and can be accessed using the `os()` method.

end_file(*self*, *body*) [operation on Page]
 Close the file using given close body tag. The default is just a close body tag, but if you specify `"` then nothing will be written (useful for a frames page)

reference(*self*, *name*, *scope*, *label*, *keys*)** [operation on Page]
 Returns a reference to the given name. The name is a scoped name, and the optional label is an alternative name to use as the link text. The name is looked up in the TOC so the link may not be local. The optional keys are appended as attributes to the A tag.

BufferPage [class]

parents: parent class A page that writes to a string buffer.

_take_control(*self*) [operation on BufferPage]

open_file(*self*) [operation on BufferPage]
 Returns a new StringIO

close_file(*self*) [operation on BufferPage]
Does nothing.

get_buffer(*self*) [operation on BufferPage]
Returns the page as a string, then deletes the internal buffer

RawFilePages [module]

RawFilePages [class]

parents: parent class A module for creating a page for each file with hyperlinked source

__init__(*self*, *manager*) [operation on RawFilePages]

filename(*self*) [operation on RawFilePages]
since RawFilePages generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on RawFilePages]
since RawFilePages generates a whole file hierarchy, this method returns the current title, which may change over the lifetime of this object

_get_files(*self*) [operation on RawFilePages]
Returns a list of (path, output_filename) for each file

process(*self*, *start*) [operation on RawFilePages]
Creates a page for every file

register_filenames(*self*, *start*) [operation on RawFilePages]
Registers a page for every file

**process_file(*self*, *original*,
 filename)** [operation on RawFilePages]
Creates a page for the given file

ScopePages [module]

ScopePages [class]

parents: parent class A module for creating a page for each Scope with summaries and details. This module is highly modular, using the classes from AST-Formatter to do the actual formatting. The classes to use may be controlled via the config script, resulting in a very configurable output.

__init__(*self*, *manager*) [operation on ScopePages]

| | |
|--|---------------------------|
| <code>_get_parts(self)</code> | [operation on ScopePages] |
| Loads the list of parts from config | |
| <code>get_toc(self, start)</code> | [operation on ScopePages] |
| Returns the TOC for the whole AST starting at start | |
| <code>filename(self)</code> | [operation on ScopePages] |
| since ScopePages generates a whole file hierarchy, this method returns the current filename, which may change over the lifetime of this object | |
| <code>title(self)</code> | [operation on ScopePages] |
| since ScopePages generates a while file hierarchy, this method returns the current title, which may change over the lifetime of this object | |
| <code>scope(self)</code> | [operation on ScopePages] |
| return the current scope processed by this object | |
| <code>process(self, start)</code> | [operation on ScopePages] |
| Creates a page for every Scope | |
| <code>register_filenames(self, start)</code> | [operation on ScopePages] |
| Registers a page for every Scope | |
| <code>process_scope(self, ns)</code> | [operation on ScopePages] |
| Creates a page for the given scope | |
| <code>end_file(self)</code> | [operation on ScopePages] |
| Overrides end_file to provide synopsis logo | |

ScopeSorter [module]
 Scope sorting class module. This module contains the class for sorting Scopes.

`compare_sections(a, b)` [function]

ScopeSorter [class]
 A class that takes a scope and sorts its children by type. To use it call set_scope, then access the sorted list by the other methods.

`__init__(self, scope)` [operation on ScopeSorter]
 Optional scope starts using that AST.Scope

`set_scope(self, scope)` [operation on ScopeSorter]
 Sort children of given scope

| | |
|--|----------------------------|
| <code>_add_decl(self, decl, name, section)</code> | [operation on ScopeSorter] |
| Adds the given decl with given name and section to the internal data | |
| <code>_section_of(self, decl)</code> | [operation on ScopeSorter] |
| <code>_sort_sections(self)</code> | [operation on ScopeSorter] |
| <code>sort_section_names(self)</code> | [operation on ScopeSorter] |
| Sorts sections names if they need it | |
| <code>_set_section_names(self, sections)</code> | [operation on ScopeSorter] |
| <code>_handle_group(self, group)</code> | [operation on ScopeSorter] |
| Handles a group | |
| <code>sort_sections(self)</code> | [operation on ScopeSorter] |
| Sorts the children of all sections, if they need it | |
| <code>child(self, name)</code> | [operation on ScopeSorter] |
| Returns the child with the given name. Throws KeyError if not found. | |
| <code>sections(self)</code> | [operation on ScopeSorter] |
| Returns a list of available section names | |
| <code>children(self, section)</code> | [operation on ScopeSorter] |
| Returns list of children in given section, or all children | |

Tags [module]
 HTML Tag generation utilities. You will probably find it easiest to import * from this module.

| | |
|---|------------|
| <code>k2a(keys)</code> | [function] |
| Convert a name/value dict to a string of attributes | |
| <code>rel(frm, to)</code> | [function] |
| Find link to to relative to frm | |
| <code>href(_ref, _label, **keys)</code> | [function] |
| Return a href to 'ref' with name 'label' and attributes | |
| <code>name(ref, label)</code> | [function] |
| Return a name anchor with given reference and label | |
| <code>span(clas, body)</code> | [function] |
| Wrap the body in a span of the given class | |

| | |
|--|------------|
| div (<i>clas</i> , <i>body</i>) | [function] |
| Wrap the body in a div of the given class | |
| entity (<i>_type</i> , <i>body</i> , <i>**keys</i>) | [function] |
| Wrap the body in a tag of given type and attributes | |
| solotag (<i>_type</i> , <i>**keys</i>) | [function] |
| Create a solo tag (no close tag) of given type and attributes | |
| desc (<i>text</i>) | [function] |
| Create a description div for the given text | |
| anglebrackets (<i>text</i>) | [function] |
| Replace angle brackets with HTML codes | |
| replace_spaces (<i>text</i>) | [function] |
| Replaces spaces in the given string with sequences. Does NOT replace spaces inside tags | |

TreeFormatterJS [module]

TreeFormatterJS [class]

parents: parent class Javascript trees. The trees have expanding and collapsing nodes. call `js_init()` with the button images and default open/close policy during process

| | |
|--|--------------------------------|
| __init__ (<i>self</i> , <i>page</i>) | [operation on TreeFormatterJS] |
| getId (<i>self</i>) | [operation on TreeFormatterJS] |
| js_init (<i>self</i> , <i>open_img</i> , <i>close_img</i> , <i>leaf_img</i> , <i>base</i> , <i>default_open</i>) | [operation on TreeFormatterJS] |
| Initialise the JSTree page. This method copies the files to the output directory and stores the values given. | |
| startTree (<i>self</i>) | [operation on TreeFormatterJS] |
| Writes the javascript | |
| formatImage (<i>self</i> , <i>id</i> , <i>filename</i> , <i>alt_text</i>) | [operation on TreeFormatterJS] |
| Returns the image element for the given image | |
| writeLeaf (<i>self</i> , <i>item_text</i>) | [operation on TreeFormatterJS] |
| Write a leaf node to the output at the current tree level. | |

writeNodeStart(*self*, *item_text*) [operation on `TreeFormatterJS`]

Write a non-leaf node to the output at the current tree level, and start a new level.

writeNodeEnd(*self*) [operation on `TreeFormatterJS`]

Finish a non-leaf node, and close the current tree level.

endTree(*self*) [operation on `TreeFormatterJS`]

Writes the end of the tree.

TreeFormatter [module]

Tree formatter interface.

This module contains the class which defines the interface for all tree views. A tree is a structure of leaves and nodes, where each leaf has one node, each node can have many child leaves or nodes, and there is one root node. Trees are used to describe the relationship between modules, and also between files. The user can select between different ways of representing trees, for example as simple nested lists or as complex javascript trees that the user can expand and collapse individual branches of. This module contains the tree interface which is common to all implementations.

TreeFormatter [class]

Interface for all tree implementations. This tree class provides default implementations of its methods for example and basic usage. The implementation provided outputs a nested tree using the UL and LI html elements.

__init__(*self*, *page*) [operation on `TreeFormatter`]

A tree is a strategy, so it must be passed the page instance to display to.

startTree(*self*) [operation on `TreeFormatter`]

Writes anything to the file that needs to be written at the start. For example a script section for the global scripts used by a javascript tree.

endTree(*self*) [operation on `TreeFormatter`]

Writes anything that needs to be written at the end.

writeLeaf(*self*, *text*) [operation on `TreeFormatter`]

Writes a leaf to the output. A leaf is a node with no children, for example a module (not package) or a file (not directory). The text is output verbatim in the appropriate html tags, which in the default instance is LI

writeNodeStart(*self*, *text*) [operation on `TreeFormatter`]

Starts a node with children. The text is written, and then a block of children is started. This method call must be followed by a corresponding `writeNodeEnd()` call. Individual leaves inside the block may be written out using the `writeLeaf()` method.

writeNodeEnd(*self*) [operation on **TreeFormatter**]
 Ends a node with children. This method just closes any tags opened by the corresponding writeNodeStart() method for a node.

XRefPages [module]

XRefLinker [class]

parents: parent class

__init__(*self*, *xref*) [operation on **XRefLinker**]

link(*self*, *name*) [operation on **XRefLinker**]

XRefPages [class]

parents: parent class A module for creating pages full of xref infos

__init__(*self*, *manager*) [operation on **XRefPages**]

get_toc(*self*, *start*) [operation on **XRefPages**]
 Returns the toc for XRefPages

filename(*self*) [operation on **XRefPages**]
 Returns the current filename, which may change over the lifetime of this object

title(*self*) [operation on **XRefPages**]
 Returns the current title, which may change over the lifetime of this object

process(*self*, *start*) [operation on **XRefPages**]
 Creates a page for every bunch of xref infos

register_filenames(*self*, *start*) [operation on **XRefPages**]
 Registers each page

process_link(*self*, *file*, *line*, *scope*) [operation on **XRefPages**]
 Outputs the info for one link

describe_decl(*self*, *decl*) [operation on **XRefPages**]
 Returns a description of the declaration. Detects constructors and destructors

process_name(*self*, *name*) [operation on **XRefPages**]
 Outputs the info for a given name

5.3 The DUMP Module

| | |
|---|-----------------------|
| usage() | [function] |
| Print usage to stdout | |
| --parseArgs(<i>args</i>) | [function] |
| format(<i>args, ast, config_obj</i>) | [function] |
| Dumper | [class] |
| __init__(<i>self</i>) | [operation on Dumper] |
| clear(<i>self</i>) | [operation on Dumper] |
| writeln(<i>self, text</i>) | [operation on Dumper] |
| lnwrite(<i>self, text</i>) | [operation on Dumper] |
| write(<i>self, text</i>) | [operation on Dumper] |
| indent(<i>self, str</i>) | [operation on Dumper] |
| outdent(<i>self</i>) | [operation on Dumper] |
| visit(<i>self, obj</i>) | [operation on Dumper] |
| visitNone(<i>self, obj</i>) | [operation on Dumper] |
| visitType(<i>self, obj</i>) | [operation on Dumper] |
| visitInt(<i>self, obj</i>) | [operation on Dumper] |
| visitLong(<i>self, obj</i>) | [operation on Dumper] |
| visitFloat(<i>self, obj</i>) | [operation on Dumper] |
| visitString(<i>self, obj</i>) | [operation on Dumper] |
| visitTuple(<i>self, obj</i>) | [operation on Dumper] |
| visitList(<i>self, obj</i>) | [operation on Dumper] |
| _dictKey(<i>self, key</i>) | [operation on Dumper] |
| visitDict(<i>self, dict, namefunc</i>) | [operation on Dumper] |
| _instAttr(<i>self, name</i>) | [operation on Dumper] |
| visitInstance(<i>self, obj</i>) | [operation on Dumper] |

5.4 The Dia Module

| | |
|---|-----------------------------|
| k2a (<i>keys</i>) | [function] |
| Convert a keys dict to a string of attributes | |
| quote (<i>str</i>) | [function] |
| Remove HTML chars from str | |
| usage () | [function] |
| Print usage to stdout | |
| __parseArgs (<i>args</i>) | [function] |
| format (<i>args, ast, config_obj</i>) | [function] |
| DiaFormatter | [class] |
| parents: parent class parent class Outputs a Dia file | |
| __init__ (<i>self, filename</i>) | [operation on DiaFormatter] |
| indent (<i>self</i>) | [operation on DiaFormatter] |
| incr (<i>self</i>) | [operation on DiaFormatter] |
| decr (<i>self</i>) | [operation on DiaFormatter] |
| write (<i>self, text</i>) | [operation on DiaFormatter] |
| scope (<i>self</i>) | [operation on DiaFormatter] |
| startTag (<i>self, tagname, **keys</i>) | [operation on DiaFormatter] |
| Starts a tag and indents, attributes using keyword arguments | |
| startTagDict (<i>self, tagname, attrs</i>) | [operation on DiaFormatter] |
| Starts a tag and indents, attributes using dictionary argument | |
| endTag (<i>self, tagname</i>) | [operation on DiaFormatter] |
| Un-indent and closes tag | |
| soloTag (<i>self, tagname, **keys</i>) | [operation on DiaFormatter] |
| Writes a solo tag with attributes from keyword arguments | |
| attribute (<i>self, name, type, value, allow_solo</i>) | [operation on DiaFormatter] |
| Writes an attribute with given name, type and value | |

| | |
|--|-----------------------------|
| output(<i>self</i>, <i>declarations</i>) | [operation on DiaFormatter] |
| Output declarations to file | |
| doDiagramData(<i>self</i>) | [operation on DiaFormatter] |
| Write the stock diagramdata stuff | |
| doInheritance(<i>self</i>, <i>inherit</i>) | [operation on DiaFormatter] |
| Create a generalization object for one inheritance | |
| createObjectID(<i>self</i>, <i>decl</i>) | [operation on DiaFormatter] |
| Creates a new object identifier, and remembers it with the given declaration | |
| getObjectID(<i>self</i>, <i>decl</i>) | [operation on DiaFormatter] |
| Returns the stored identifier for the given object | |
| formatType(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| Returns a string representation for the given type | |
| visitBaseType(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitUnknown(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitDeclared(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitModifier(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitParametrized(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitFunctionType(<i>self</i>, <i>type</i>) | [operation on DiaFormatter] |
| visitDeclaration(<i>self</i>, <i>decl</i>) | [operation on DiaFormatter] |
| Default is to not do anything with it | |
| visitModule(<i>self</i>, <i>decl</i>) | [operation on DiaFormatter] |
| Just traverse child declarations | |
| visitClass(<i>self</i>, <i>decl</i>) | [operation on DiaFormatter] |
| Creates a Class object for one class, with attributes and operations | |

5.5 The DocBook Module

usage() [function]
Print usage to stdout

--parseArgs(args) [function]

format(args, ast, config) [function]

Formatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

__init__(self, os) [operation on Formatter]

scope(self) [operation on Formatter]

push_scope(self, newscope) [operation on Formatter]

pop_scope(self) [operation on Formatter]

write(self, text) [operation on Formatter]
Write some text to the output stream, replacing 's with 's and indents.

start_entity(self, __type, **__params) [operation on Formatter]
Write the start of an entity, ending with a newline

end_entity(self, type) [operation on Formatter]
Write the end of an entity, starting with a newline

write_entity(self, __type, __body, **__params) [operation on Formatter]
Write a single entity on one line (though body may contain newlines)

entity(self, __type, __body, **__params) [operation on Formatter]
Return but do not write the text for an entity on one line

reference(self, ref, label) [operation on Formatter]
reference takes two strings, a reference (used to look up the symbol and generated the reference), and the label (used to actually write it)

label(self, ref) [operation on Formatter]

type_label(self) [operation on Formatter]

| | |
|--|--------------------------|
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on Formatter] |
| visitComment (<i>self</i> , <i>comment</i>) | [operation on Formatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on Formatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on Formatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on Formatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on Formatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on Formatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on Formatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on Formatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on Formatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on Formatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on Formatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on Formatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on Formatter] |

DocFormatter [class]

parents: parent class A specialized version that just caters for the needs of the Doc-Book manual's Config section. Only modules and classes are printed, and the docbook elements classsynopsis etc are not used.

| | |
|--|-----------------------------|
| visitComment (<i>self</i> , <i>comment</i>) | [operation on DocFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on DocFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on DocFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on DocFormatter] |

5.6 The Dot Module

| | |
|--|-------------------------------------|
| system (<i>command</i>) | [function] |
| Run the command. If the command fails, an exception SystemError is thrown. | |
| usage () | [function] |
| Print usage to stdout | |
| --parseArgs (<i>args</i> , <i>config_obj</i>) | [function] |
| _rel (<i>frm</i> , <i>to</i>) | [function] |
| Find link to to relative to frm | |
| _convert_map (<i>input</i> , <i>output</i>) | [function] |
| convert map generated from Dot to a html region map. input and output are (open) streams | |
| _format (<i>input</i> , <i>output</i> , <i>format</i>) | [function] |
| _format_png (<i>input</i> , <i>output</i>) | [function] |
| _format_html (<i>input</i> , <i>output</i>) | [function] |
| generate (active) image for html. input and output are file names. If output ends in '.html', its stem is used with an '.png' suffix for the actual image. | |
| format (<i>args</i> , <i>ast</i> , <i>config_obj</i>) | [function] |
| SystemError | [class] |
| Error thrown by the system() function. Attributes are 'retval', encoded as per os.wait(): low-byte is killing signal number, high-byte is return value of command. | |
| --init-- (<i>self</i> , <i>retval</i> , <i>command</i>) | [operation on SystemError] |
| --repr-- (<i>self</i>) | [operation on SystemError] |
| InheritanceFormatter | [class] |
| parents: parent class parent class A Formatter that generates an inheritance graph | |
| --init-- (<i>self</i> , <i>os</i> , <i>operations</i> , <i>attributes</i>) | [operation on InheritanceFormatter] |
| scope (<i>self</i>) | [operation on InheritanceFormatter] |
| write (<i>self</i> , <i>text</i>) | [operation on InheritanceFormatter] |

| | |
|---|-------------------------------------|
| <code>type_ref(self)</code> | [operation on InheritanceFormatter] |
| <code>type_label(self)</code> | [operation on InheritanceFormatter] |
| <code>parameter(self)</code> | [operation on InheritanceFormatter] |
| <code>formatType(self, typeObj)</code> | [operation on InheritanceFormatter] |
| Returns a reference string for the given type object | |
| <code>clearType(self)</code> | [operation on InheritanceFormatter] |
| <code>writeNode(self, ref, name, label, **attr)</code> | [operation on InheritanceFormatter] |
| helper method to generate output for a given node | |
| <code>writeEdge(self, parent, child, label, **attr)</code> | [operation on InheritanceFormatter] |
| <code>getClassName(self, node)</code> | [operation on InheritanceFormatter] |
| Returns the name of the given class node, relative to all its parents. This makes the graph simpler by making the names shorter | |
| <code>visitModifier(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitUnknown(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitBase(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitDependent(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitDeclared(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitParametrized(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitTemplate(self, type)</code> | [operation on InheritanceFormatter] |
| <code>visitInheritance(self, node)</code> | [operation on InheritanceFormatter] |
| <code>visitClass(self, node)</code> | [operation on InheritanceFormatter] |
| <code>visitOperation(self, operation)</code> | [operation on InheritanceFormatter] |
| <code>visitVariable(self, variable)</code> | [operation on InheritanceFormatter] |

SingleInheritanceFormatter [class]
 parents: parent class A Formatter that generates an inheritance graph for a specific class. This Visitor visits the AST upwards, i.e. following the inheritance links, instead of the declarations contained in a given scope.

`__init__(self, os, operations, attributes, levels, types)` [operation on `SingleInheritanceFormatter`]

`visitDeclared(self, type)` [operation on `SingleInheritanceFormatter`]

`visitInheritance(self, node)` [operation on `SingleInheritanceFormatter`]

`visitClass(self, node)` [operation on `SingleInheritanceFormatter`]

CollaborationFormatter [class]

parents: parent class parent class A Formatter that generates a collaboration graph

`__init__(self, output)` [operation on `CollaborationFormatter`]

`write(self, text)` [operation on `CollaborationFormatter`]

`visitClass(self, node)` [operation on `CollaborationFormatter`]

`visitVariable(self, node)` [operation on `CollaborationFormatter`]

`visitBaseType(self, type)` [operation on `CollaborationFormatter`]

`visitUnknown(self, type)` [operation on `CollaborationFormatter`]

`visitDeclared(self, type)` [operation on `CollaborationFormatter`]

`visitModifier(self, type)` [operation on `CollaborationFormatter`]

`visitTemplate(self, type)` [operation on `CollaborationFormatter`]

`visitParametrized(self, type)` [operation on `CollaborationFormatter`]

`visitFunctionType(self, type)` [operation on `CollaborationFormatter`]

5.7 The HTML_Simple Module

| | |
|--|--------------------------------|
| entity (<i>type</i> , <i>body</i>) | [function] |
| href (<i>ref</i> , <i>label</i>) | [function] |
| name (<i>ref</i> , <i>label</i>) | [function] |
| span (<i>clas</i> , <i>body</i>) | [function] |
| div (<i>clas</i> , <i>body</i>) | [function] |
| desc (<i>text</i>) | [function] |
| usage () | [function] |
| Print usage to stdout | |
| --parseArgs (<i>args</i>) | [function] |
| format (<i>args</i> , <i>ast</i> , <i>config_obj</i>) | [function] |
| TableOfContents | [class] |
| parents: parent class | |
| generate a dictionary of all declarations which can be looked up to create cross references. Names are fully scoped. | |
| --init-- (<i>self</i>) | [operation on TableOfContents] |
| write (<i>self</i> , <i>os</i>) | [operation on TableOfContents] |
| lookup (<i>self</i> , <i>name</i>) | [operation on TableOfContents] |
| insert (<i>self</i> , <i>name</i>) | [operation on TableOfContents] |
| visitAST (<i>self</i> , <i>node</i>) | [operation on TableOfContents] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on TableOfContents] |
| visitGroup (<i>self</i> , <i>group</i>) | [operation on TableOfContents] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on TableOfContents] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on TableOfContents] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on TableOfContents] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on TableOfContents] |

| | |
|--|--------------------------------|
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on TableOfContents] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on TableOfContents] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on TableOfContents] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on TableOfContents] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on TableOfContents] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on TableOfContents] |

HTMLFormatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

| | |
|--|--|
| __init__ (<i>self</i> , <i>os</i> , <i>toc</i>) | [operation on HTMLFormatter] |
| scope (<i>self</i>) | [operation on HTMLFormatter] |
| write (<i>self</i> , <i>text</i>) | [operation on HTMLFormatter] |
| reference (<i>self</i> , <i>ref</i> , <i>label</i>) | [operation on HTMLFormatter] reference takes two strings, a reference (used to look up the symbol and generated the reference), and the label (used to actually write it) |
| label (<i>self</i> , <i>ref</i>) | [operation on HTMLFormatter] |
| visitBaseType (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitUnknown (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitDeclared (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitModifier (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitParametrized (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitFunctionType (<i>self</i> , <i>type</i>) | [operation on HTMLFormatter] |
| visitDeclarator (<i>self</i> , <i>node</i>) | [operation on HTMLFormatter] |
| visitTypedef (<i>self</i> , <i>typedef</i>) | [operation on HTMLFormatter] |
| visitVariable (<i>self</i> , <i>variable</i>) | [operation on HTMLFormatter] |
| visitConst (<i>self</i> , <i>const</i>) | [operation on HTMLFormatter] |

| | |
|--|------------------------------|
| visitGroup (<i>self</i> , <i>group</i>) | [operation on HTMLFormatter] |
| visitModule (<i>self</i> , <i>module</i>) | [operation on HTMLFormatter] |
| visitClass (<i>self</i> , <i>clas</i>) | [operation on HTMLFormatter] |
| visitInheritance (<i>self</i> , <i>inheritance</i>) | [operation on HTMLFormatter] |
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on HTMLFormatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on HTMLFormatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on HTMLFormatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on HTMLFormatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on HTMLFormatter] |

5.8 The TexInfo Module

| | |
|---|---------------------------------|
| <code>_replace(<i>comm</i>, <i>old</i>, <i>new</i>)</code> | [function] |
| <code>usage()</code> Print usage to stdout | [function] |
| <code>--parseArgs(<i>args</i>)</code> | [function] |
| <code>format(<i>args</i>, <i>ast</i>, <i>config</i>)</code> | [function] |
| Struct Dummy class. Initialise with keyword args. | [class] |
| <code>--init__(<i>self</i>, <i>**keys</i>)</code> | [operation on Struct] |
| CommentFormatter A class that takes a comment Struct and formats its contents. | [class] |
| <code>parse(<i>self</i>, <i>comm</i>)</code> Parse the comment struct | [operation on CommentFormatter] |
| JavadocFormatter parents: parent class A formatter that formats comments similar to Javadoc @tags | [class] |
| <code>--init__(<i>self</i>)</code> Create regex objects for regexps | [operation on JavadocFormatter] |
| <code>parse(<i>self</i>, <i>comm</i>)</code> Parse the comm.detail for @tags | [operation on JavadocFormatter] |
| <code>extract(<i>self</i>, <i>regexp</i>, <i>str</i>)</code> Extracts all matches of the regexp from the text. The MatchObjects are returned in a list | [operation on JavadocFormatter] |
| <code>parseTags(<i>self</i>, <i>str</i>, <i>joiner</i>)</code> Returns text, tags | [operation on JavadocFormatter] |
| <code>parseText(<i>self</i>, <i>str</i>, <i>decl</i>)</code> | [operation on JavadocFormatter] |
| <code>parse_see(<i>self</i>, <i>str</i>, <i>decl</i>)</code> Parses inline tags | [operation on JavadocFormatter] |
| <code>format_params(<i>self</i>, <i>param_tags</i>)</code> Formats a list of (param, description) tags | [operation on JavadocFormatter] |

| | |
|--|---------------------------------|
| format_attrs (<i>self</i> , <i>attr_tags</i>) | [operation on JavadocFormatter] |
| Formats a list of (attr, description) tags | |
| format_return (<i>self</i> , <i>return_tag</i>) | [operation on JavadocFormatter] |
| Formats a since description string | |
| format_see (<i>self</i> , <i>see_tags</i> , <i>decl</i>) | [operation on JavadocFormatter] |
| Formats a list of (ref,description) tags | |
| find_link (<i>self</i> , <i>ref</i> , <i>decl</i>) | [operation on JavadocFormatter] |
| Given a "reference" and a declaration, returns a HTML link. Various methods are tried to resolve the reference. First the parameters are taken off, then we try to split the ref using '.' or '::'. The params are added back, and then we try to match this scoped name against the current scope. If that fails, then we recursively try enclosing scopes. | |
| find_link_at (<i>self</i> , <i>ref</i> , <i>scope</i>) | [operation on JavadocFormatter] |
| find_method_entry (<i>self</i> , <i>name</i> , <i>scope</i>) | [operation on JavadocFormatter] |
| Tries to find a TOC entry for a method adjacent to decl. The enclosing scope is found using the types dictionary, and the realname()'s of all the functions compared to ref. | |

Escapifier [class]

parents: parent class escapify the strings to become valid texinfo text. Only replace '@' by '@@' if these are not part of valid texinfo tags.

| | |
|--|---------------------------|
| __init__ (<i>self</i>) | [operation on Escapifier] |
| parse (<i>self</i> , <i>comm</i>) | [operation on Escapifier] |

MenuMaker [class]

parents: parent class generate a texinfo menu for the declarations of a given scope

| | |
|--|--------------------------|
| __init__ (<i>self</i> , <i>scope</i> , <i>os</i>) | [operation on MenuMaker] |
| write (<i>self</i> , <i>text</i>) | [operation on MenuMaker] |
| start (<i>self</i>) | [operation on MenuMaker] |
| end (<i>self</i>) | [operation on MenuMaker] |
| visitDeclaration (<i>self</i> , <i>node</i>) | [operation on MenuMaker] |

Formatter [class]

parents: parent class parent class

The type visitors should generate names relative to the current scope. The generated references however are fully scoped names

__init__(*self*, *os*) [operation on Formatter]

scope(*self*) [operation on Formatter]

write(*self*, *text*) [operation on Formatter]

escapify(*self*, *label*) [operation on Formatter]

type_label(*self*) [operation on Formatter]

decl_label(*self*, *decl*) [operation on Formatter]

formatType(*self*, *type*) [operation on Formatter]

Returns a reference string for the given type object

formatComments(*self*, *decl*) [operation on Formatter]

visitBaseType(*self*, *type*) [operation on Formatter]

visitUnknown(*self*, *type*) [operation on Formatter]

visitDeclared(*self*, *type*) [operation on Formatter]

visitModifier(*self*, *type*) [operation on Formatter]

visitParametrized(*self*, *type*) [operation on Formatter]

visitFunctionType(*self*, *type*) [operation on Formatter]

visitDeclarator(*self*, *node*) [operation on Formatter]

visitTypedef(*self*, *typedef*) [operation on Formatter]

visitVariable(*self*, *variable*) [operation on Formatter]

visitConst(*self*, *const*) [operation on Formatter]

visitModule(*self*, *module*) [operation on Formatter]

visitClass(*self*, *clas*) [operation on Formatter]

visitInheritance(*self*, *inheritance*) [operation on Formatter]

| | |
|--|--------------------------|
| visitParameter (<i>self</i> , <i>parameter</i>) | [operation on Formatter] |
| visitFunction (<i>self</i> , <i>function</i>) | [operation on Formatter] |
| visitOperation (<i>self</i> , <i>operation</i>) | [operation on Formatter] |
| visitEnumerator (<i>self</i> , <i>enumerator</i>) | [operation on Formatter] |
| visitEnum (<i>self</i> , <i>enum</i>) | [operation on Formatter] |

Type Index

__locale_struct * 236
 void * 264
 -
 __gnu_cxx 236
 __locale_struct 235
 __locale_t 187
 ‘
 ‘0027 286
 ‘0028 286
 ‘0029 286
 {
 {builder.cc} 236
 {filter.cc} 237
 {link.cc} 237
 {link_map.cc} 241
 {linkstore.cc} 242
 {lookup.cc} 243
 {occ.cc} 246
 A
 Access 161
 AccessRestrictor 109, 333
 Action 9
 ActionCanvas 132
 ActionColorizer 131
 ActionDialog 139
 ActionIcon 131
 ActionPropertiesDialog 131
 actionvis 128
 ActionVisitor 9
 actionwiz 135
 AddActionStrategy 130
 AddWizard 140
 Array 35, 153, 226, 260
 ASCII 8, 40
 ASCIIFormatter 40, 345
 AST 12, 13, 161
 ASTFormatter 64, 347
 ASTTranslator 123, 327

B

Base 6, 33, 151, 223
 BaseAST 88, 367
 BigHashTable 265
 Bind 261
 BindClassName 262
 BindEnumName 262
 BindTemplateClass 263
 BindTemplateFunction 263
 BindTypedefName 262
 BindVarName 261
 bool 317
 BoostBook 8, 42
 browse 141
 BrowserWindow 141
 BufferFilePages 144
 BufferPage 102, 378
 BufferScopePages 144
 Builder 179

C

C++ 121, 161
 CacherAction 12
 CacherExecutor 26
 CacherPage 136
 CanvasStrategy 128
 CanvasView 134
 CanvasWindow 134
 ChangedMemberList 270
 char_traits<unsigned> 186
 Class 18, 170, 221
 ClassArray 273
 ClassBodyWalker 249
 ClassBrowser 143
 ClassHierarchyGraph 93, 370
 ClassHierarchySimple 92, 370
 ClassInfo 126, 330
 ClassTree 44, 45
 ClassWalker 250
 cleanup 194
 ClientDataLink 253
 Cmem 270
 code::iterator 187
 CollaborationFormatter 53, 392
 Comment 22, 161
 CommentedLeaf 289
 CommentFormatter 58, 69, 70, 352, 396
 CommentFormatterStrategy 70, 352
 CommentProcessor 110, 333
 Comments 109, 114, 333, 337
 CommentTag 21
 CompilerInfo 122, 160

Config 5, 73, 116, 339, 355
 ConnectStrategy 130
 Const 19, 174
 Context 199
 core 72, 354
 Core 9
 CppParserPage 137
 CrossReferencer 63
 CXX 6
 CXX2IDL 117, 340

D

Data 191
 Declaration 15, 162
 Declared 34, 152, 224
 DeclKind 274
 DeclStyle 75, 356
 Decoder 188
 Default 90, 368
 Dependent 33, 152, 223
 Detail 68, 351
 DetailAST 92, 370
 DetailCommenter 92, 370
 DetailsPage 139
 Dia 8, 46
 DiaFormatter 46, 386
 Dictionary 36, 155, 189
 DirBrowse 77, 358
 Directory 28
 DocBook 8, 48
 DocFormatter 50, 389
 DocoBrowser 144
 DODetail 79, 360
 DODetailAST 78, 359
 DOScopeSorter 78, 358
 DOSummary 79, 359
 DOSummaryAST 78, 359
 DOSummaryCommenter 78, 359
 Dot 8, 50
 doxygen 77, 358
 Dummies 112, 335
 DUMP 8, 54
 Dumper 54, 192, 385
 DupLeaf 289

E

EmptyNS 114, 338
 emul 121, 160
 encode 202
 encode_name 202
 Encoding 254
 Enum 19, 174, 221
 Enumerator 18, 173
 Environment 257
 EqualScope 185

Error 32, 151
 Escapifier 60, 397
 ExcludeSourceRule 10
 Executor 24
 ExecutorCreator 24
 exparse 124, 329

F

FakeGC 194
 File 28
 FileDetails 79, 80, 360
 FileFilter 194
 FileIndexer 80, 361
 FileLayout 81, 361
 FileListing 84, 364
 FileSource 7, 85, 364
 FileTree 7, 27, 86, 365
 FileTreeJS 86, 365
 FormatAction 12
 FormatExecutor 27
 FormatStrategy 87, 366
 Formatter 6, 40, 42, 48, 60, 388, 398
 FormatterPage 138
 Forward 16, 171
 FramesIndex 93, 371
 FuncImplCache 215
 FuncPtr 227
 Function 20, 36, 154, 176
 function_type 273
 FunctionHeuristic 245
 FunctionInfo 126, 330

G

gc 317
 gc_cleanup 317
 GlobSourceRule 10
 GraphBrowser 145
 Group 16
 Grouper 113, 337

H

HashTable 264
 HashTableEntry 234
 Heading 67, 91, 350, 369
 HTML 7, 63
 HTML_Doxygen 8
 HTML_Simple 8, 55
 HTMLFormatter 56, 394

I

Icon 132, 145
 IDL 6, 122
 igrph 145
 IGraphWindow 145
 Include 13, 165
 Inheritance 17, 69, 93, 169, 351, 371
 InheritanceAdder 237
 InheritanceFormatter 51, 390
 InheritanceGraph 7, 93, 94, 371, 372
 InheritanceTree 94, 95, 372
 int 186, 317
 isType 244

J

JavaComments 110, 334
 JavadocFormatter 58, 71, 353, 396
 JavaTags 114, 337
 JSTree 95, 372

K

KeyError 190
 Kind 261

L

LanguageMapper 115, 339
 Leaf 288
 LeafAUTO 290
 LeafBOOLEAN 290
 LeafCHAR 290
 LeafCONST 291
 LeafDOUBLE 291
 LeafEXTERN 291
 LeafFLOAT 291
 LeafFRIEND 291
 LeafINLINE 292
 LeafINT 292
 LeafLONG 292
 LeafMUTABLE 292
 LeafName 289
 LeafNAMESPACE 293
 LeafPRIVATE 293
 LeafPROTECTED 293
 LeafPUBLIC 293
 LeafREGISTER 293
 LeafReserved 289
 LeafSHORT 294
 LeafSIGNED 294
 LeafSTATIC 294
 LeafThis 290
 LeafUNSIGNED 294
 LeafUserKeyword2 295
 LeafUSING 295

LeafVIRTUAL 295
 LeafVOID 295
 LeafVOLATILE 295
 Lex 313
 Line 132
 LineMapNode 231
 Link 238
 Linker 6, 62, 109, 116, 339
 LinkerAction 11
 LinkerExecutor 26
 LinkMap 198
 LinkStore 199
 ListFiller 142
 Lookup 202
 lt_col 238

M

Macro 16, 167
 main 146
 MainWindow 146
 Mapper 117, 340
 mbstate_t 186
 Mem 268
 Member 266
 MemberFunction 268
 MemberList 268
 MenuHandler 129
 MenuMaker 60, 397
 Metaclass 271
 MetaModule 17
 Modifier 34, 153, 225
 Module 17
 ModuleIndexer 96, 373
 ModuleInfo 127, 331
 ModuleListing 7, 97, 374
 ModuleListingJS 96, 374
 MultipleError 190

N

Named 33, 151, 222
 NameIndex 98, 375
 NameMapper 117, 340
 NameScope 324
 Namespace 169
 NamespaceType 179
 NestedFileLayout 83, 363
 Node 27, 241
 NonLeaf 296

O

omni 122, 327
 opcxx_ListOfMetaclass 273
 Operation 21, 117, 178, 340
 ostream_ptr_iterator 236, 243

P

| | |
|------------------------------|------------------------------|
| PackageBrowser | 143 |
| Page | 99, 100, 376, 377 |
| PageFormat | 99, 376 |
| PageManager | 75, 357 |
| Parameter | 20, 175 |
| Parameterized | 227 |
| Parametrized | 35, 154 |
| Parser | 6, 121, 160, 274 |
| ParserAction | 11 |
| ParserExecutor | 25 |
| ParserPage | 137 |
| Part | 64, 347 |
| PostDivFormatter | 78, 359 |
| Postfix_Flag | 215 |
| PostSummaryDiv | 79, 359 |
| PreDivFormatter | 78, 359 |
| PreSummaryDiv | 78, 359 |
| Previous | 113, 336 |
| Private | 185, 197, 199, 201, 220, 231 |
| Program | 231 |
| ProgramFile | 248 |
| ProgramFromStdin | 248 |
| ProgramString | 248 |
| project | 147 |
| Project | 28 |
| ProjectActions | 29 |
| ProjectReader | 31 |
| ProjectWindow | 147 |
| ProjectWriter | 31 |
| Ptree | 281 |
| PtreeAccessDecl | 303 |
| PtreeAccessSpec | 303 |
| PtreeArray | 287 |
| PtreeArrayExpr | 310 |
| PtreeArrowMemberExpr | 312 |
| PtreeAssignExpr | 307 |
| PtreeBlock | 297 |
| PtreeBrace | 296 |
| PtreeBreakStatement | 305 |
| PtreeCaseStatement | 306 |
| PtreeCastExpr | 308 |
| PtreeClassBody | 297 |
| PtreeClassSpec | 302 |
| PtreeCommaExpr | 307 |
| PtreeCondExpr | 307 |
| PtreeContinueStatement | 305 |
| PtreeDeclaration | 299 |
| PtreeDeclarator | 300 |
| PtreeDefaultStatement | 306 |
| PtreeDeleteExpr | 310 |
| PtreeDoStatement | 304 |
| PtreeDotMemberExpr | 312 |
| PtreeEnumSpec | 302 |
| PtreeExprStatement | 306 |
| PtreeExternTemplate | 298 |
| PtreeForStatement | 304 |
| PtreeFstyleCastExpr | 301 |
| PtreeFuncallExpr | 311 |
| PtreeGotoStatement | 305 |
| PtreeHead | 288 |
| PtreeIfStatement | 303 |
| PtreeInfixExpr | 308 |
| PtreeIter | 287 |
| PtreeLabelStatement | 306 |
| PtreeLinkageSpec | 298 |
| PtreeMetaClassDecl | 298 |
| PtreeName | 301 |
| PtreeNamespaceSpec | 299 |
| PtreeNewExpr | 310 |
| PtreeParenExpr | 312 |
| PtreePmExpr | 308 |
| PtreePostfixExpr | 311 |
| PtreeReturnStatement | 305 |
| PtreeSizeofExpr | 309 |
| PtreeStaticUserStatementExpr | 313 |
| PtreeSwitchStatement | 304 |
| PtreeTemplateDecl | 297 |
| PtreeTemplateInstantiation | 298 |
| PtreeThrowExpr | 309 |
| PtreeTryStatement | 305 |
| PtreeTypedef | 297 |
| PtreeTypeidExpr | 309 |
| PtreeTypeofExpr | 310 |
| PtreeUnaryExpr | 308 |
| PtreeUserAccessSpec | 303 |
| PtreeUserdefKeyword | 303 |
| PtreeUserStatementExpr | 311 |
| PtreeUsing | 299 |
| PtreeWhileStatement | 304 |
| python | 127, 331 |
| Python | 6, 124 |
| PyWriter | 39, 157 |
| PyWriterStruct | 40, 158 |
| | |
| Q | |
| Qt | 128 |
| QtComments | 111, 335 |
| QtDocFormatter | 72, 354 |
| QuoteClass | 313 |
| QuoteHTML | 70, 352 |
| | |
| R | |
| RawFilePages | 102, 379 |
| Reference | 161 |
| Replacement | 231 |
| rw_table | 236 |

S

| | |
|--|---|
| Scope | 17, 168 |
| ScopeInfo | 205 |
| ScopePages | 7, 8, 103, 379 |
| ScopeSorter | 104, 380 |
| SectionFormatter | 72, 354 |
| SelectionListener | 141 |
| SelectionStrategy | 129 |
| SimpleSourceRule | 10 |
| SingleInheritanceFormatter | 53, 391 |
| Slot | 315 |
| SourceAction | 11 |
| SourceActionEditor | 135 |
| SourceBrowser | 144 |
| sourceeditexclude | 148 |
| SourceEditExclude | 148 |
| sourceeditglob | 148 |
| SourceEditGlob | 148 |
| sourceeditsimple | 148 |
| SourceEditSimple | 148 |
| SourceExecutor | 25 |
| SourceFile | 14, 166 |
| sourceinsertwizard | 149 |
| SourceInsertWizard | 149 |
| SourceLinker | 91, 369 |
| SourceMimeFactory | 144 |
| sourceoptionsdialog | 150 |
| SourceOptionsDialog | 150 |
| SourceRule | 10 |
| SSComments | 111, 334 |
| SSDComments | 110, 333 |
| std | 186 |
| std::basic_string<unsigned char > | 187 |
| std::map< AST::Scope *, ScopeInfo *> | 185 |
| std::map< AST::SourceFile *, Streams > | 202 |
| std::map< void *, PyObject *> | 220 |
| std::map<int, Line> | 239, 241 |
| std::map<ScopedName, std::vector<AST::Reference>> | 185 |
| std::map<std::string, AST::SourceFile *> | 197 |
| std::map<std::string, std::string> | 239 |
| std::multimap<std::string, Types::Named *> | 237 |
| std::set< Link *, lt_col > | 239 |
| std::set<Node> | 241 |
| std::streamoff | 186 |
| std::streampos | 186 |
| std::vector< AST::Parameter *> | 224, 245 |
| std::vector< Comment *> | 162 |
| std::vector< Declaration *> | 163 |
| std::vector< Enumerator *> | 173 |
| std::vector< Function *> | 176 |
| std::vector< Include *> | 165 |
| std::vector< Inheritance *> | 169 |
| std::vector< Named *> | 222 |
| std::vector< Parameter *> | 175 |
| std::vector< ScopeInfo *> | 179 |
| std::vector< SourceFile *> | 166 |
| std::vector< Template *> | 224 |
| std::vector< Type *> | 222 |
| std::vector< Types::Named *> | 190 |
| std::vector< Types::Type *> | 245 |
| std::vector<FuncImplCache> | 215 |
| std::vector<FuncImplVec> | 215 |
| std::vector<size_t> | 172 |
| std::vector<std::string> | 167, 169, 175, 176, 185, 197, 222, 238 |
| StoreType | 198 |
| STrace | 206 |
| Strategy | 87, 366 |
| Streams | 201 |
| Stripper | 117, 118, 340, 341 |
| Struct | 58, 75, 135, 356, 396 |
| SuiteFuncInfo | 126, 330 |
| SuiteInfoBase | 125, 330 |
| Summarizer | 114, 337 |
| Summary | 68, 350 |
| SummaryAST | 90, 368 |
| SummaryCommenter | 91, 369 |
| SWalker | 207 |
| Synopsis | 5, 160, 216 |
| SystemError | 51, 390 |
| T | |
| TableOfContents | 55, 62, 393 |
| Tags | 105, 381 |
| Template | 34, 153, 224 |
| TemplateClass | 272 |
| TemplateDeclKind | 275 |
| TemplatePageFormat | 100, 376 |
| TexInfo | 8, 58 |
| TOC | 62 |
| TocEntry | 62 |
| ToDecl | 94, 371 |
| Token | 313 |
| TokenFifo | 314 |
| Trace | 216 |
| Transformer | 112, 335 |
| TranslateError | 206 |
| TreeFormatter | 107, 383 |
| TreeFormatterJS | 106, 382 |
| Type | 32, 151, 222, 238, 241 |
| Typedef | 18, 172 |
| TypeFormatter | 191 |
| TypeInfo | 230 |
| TypeInfoId | 316 |
| TypeResolver | 247 |
| Types | 222 |
| TypeStorer | 242 |
| TypeTranslator | 122, 327 |

U

UI 128
Unduplicator 118, 119, 341
Unknown 33, 152, 223
unsigned char 186
unsigned int 317
Util 37

V

v_Param::iterator 245
v_Type::iterator 245
Variable 19, 172

Visitor 23, 37, 155, 178, 228

W

Walker 317
wrong_type_cast 229

X

xref 63
XRefCompiler 6, 120, 121, 343
XRefLinker 91, 108, 369, 384
XRefPages 108, 384

| | |
|------------------|--|
| - | |
| __add_dir | (on FileTree 28 |
| __add_file | (on FileTree 28 |
| __call__ | (on ToDecl 94, 371 |
| __cmp__ | (on Array 35, 154 |
| __cmp__ | (on Base 33, 152 |
| __cmp__ | (on Declared 34, 152 |
| __cmp__ | (on Dependent 33, 152 |
| __cmp__ | (on Function 21 |
| __cmp__ | (on Modifier 35, 153 |
| __cmp__ | (on Parameter 20 |
| __cmp__ | (on Parametrized 36, 154 |
| __cmp__ | (on Template 34, 153 |
| __cmp__ | (on Type 32, 151 |
| __cmp__ | (on Unknown 34, 152 |
| __convert_from_v | (..... 187 |
| __delitem__ | (on Dictionary 36, 155 |
| __fire | (on ProjectActions 30 |
| __getitem__ | (on Dictionary 36, 155 |
| __getitem__ | (on SuiteInfoBase 126, 330 |
| __init__ | (on AccessRestrictor 109, 333 |
| __init__ | (on Action 9 |
| __init__ | (on ActionCanvas 133 |
| __init__ | (on ActionColorizer 131 |
| __init__ | (on ActionDialog 139 |
| __init__ | (on ActionIcon 131 |
| __init__ | (on ActionPropertiesDialog 131 |
| __init__ | (on AddActionStrategy 131 |
| __init__ | (on AddWizard 140 |
| __init__ | (on Array 35, 153 |
| __init__ | (on ASCII 8 |
| __init__ | (on ASCIIFormatter 40, 345 |
| __init__ | (on AST 13 |
| __init__ | (on ASTTranslator 123, 327 |
| __init__ | (on Base 8, 33, 151 |
| __init__ | (on BoostBook 8 |
| __init__ | (on BrowserWindow 141 |
| __init__ | (on BufferFilePages 145 |
| __init__ | (on BufferScopePages 144 |
| __init__ | (on CacherAction 12 |
| __init__ | (on CacherExecutor 26 |
| __init__ | (on CacherPage 136 |
| __init__ | (on CanvasStrategy 128 |
| __init__ | (on CanvasView 134 |
| __init__ | (on CanvasWindow 134 |
| __init__ | (on Class 18 |
| __init__ | (on ClassBrowser 143 |
| __init__ | (on ClassInfo 126, 330 |
| __init__ | (on ClassTree 45 |
| __init__ | (on CollaborationFormatter 53, 392 |
| __init__ | (on Comment 22 |
| __init__ | (on CommentFormatter 70, 352 |
| __init__ | (on Comments 114, 338 |
| __init__ | (on CommentTag 22 |
| __init__ | (on CompilerInfo 122, 160 |
| __init__ | (on Config 73, 116, 339, 355 |
| __init__ | (on ConnectStrategy 130 |
| __init__ | (on Const 19 |
| __init__ | (on CppParserPage 138 |
| __init__ | (on CrossReferencer 63 |
| __init__ | (on CXX 6 |
| __init__ | (on CXX2IDL 117, 340 |
| __init__ | (on Declaration 15 |
| __init__ | (on Declared 34, 152 |
| __init__ | (on DeclStyle 75, 356 |
| __init__ | (on Dependent 33, 152 |
| __init__ | (on Detail 68, 351 |
| __init__ | (on DetailsPage 139 |
| __init__ | (on Dia 8 |
| __init__ | (on DiaFormatter 46, 386 |
| __init__ | (on Dictionary 36, 155 |
| __init__ | (on DirBrowse 77, 358 |
| __init__ | (on Directory 28 |
| __init__ | (on DocBook 8 |
| __init__ | (on DocoBrowser 144 |
| __init__ | (on Dot 8 |
| __init__ | (on DUMP 8 |
| __init__ | (on Dumper 54, 385 |
| __init__ | (on EmptyNS 115, 338 |
| __init__ | (on Enum 19 |
| __init__ | (on Enumerator 19 |
| __init__ | (on Error 32, 151 |
| __init__ | (on Escapifier 60, 397 |
| __init__ | (on ExcludeSourceRule 10 |
| __init__ | (on ExecutorCreator 24 |
| __init__ | (on File 28 |
| __init__ | (on FileDetails 80, 360 |
| __init__ | (on FileIndexer 80, 361 |
| __init__ | (on FileLayout 81, 361 |
| __init__ | (on FileListing 84, 364 |
| __init__ | (on FileSource 85, 364 |
| __init__ | (on FileTree 28, 86, 365 |
| __init__ | (on FormatAction 12 |
| __init__ | (on FormatExecutor 27 |
| __init__ | (on Formatter 43, 48, 60, 388, 398 |
| __init__ | (on FormatterPage 139 |
| __init__ | (on Forward 16 |
| __init__ | (on FramesIndex 93, 371 |
| __init__ | (on Function 20, 36, 154 |
| __init__ | (on FunctionInfo 126, 330 |
| __init__ | (on GlobSourceRule 10 |
| __init__ | (on GraphBrowser 145 |
| __init__ | (on Group 16 |
| __init__ | (on Grouper 113, 337 |
| __init__ | (on Heading 67, 350 |
| __init__ | (on HTML 7 |
| __init__ | (on HTML_Simple 8 |
| __init__ | (on HTMLFormatter 56, 394 |
| __init__ | (on Icon 132, 145 |
| __init__ | (on IDL 6 |
| __init__ | (on IGraphWindow 146 |
| __init__ | (on Include 13 |
| __init__ | (on Inheritance 17, 69, 351 |
| __init__ | (on InheritanceFormatter 51, 390 |

- __init__(on InheritanceGraph..... 94, 372
- __init__(on InheritanceTree..... 95, 372
- __init__(on JavaComments..... 110, 334
- __init__(on JavadocFormatter.. 58, 71, 353, 396
- __init__(on JavaTags..... 114, 337
- __init__(on JSTree..... 95, 373
- __init__(on Line..... 132
- __init__(on Linker..... 6
- __init__(on LinkerAction..... 11
- __init__(on LinkerExecutor..... 26
- __init__(on ListFiller..... 142
- __init__(on Macro..... 16
- __init__(on MainWindow..... 146
- __init__(on Mapper..... 117, 340
- __init__(on MenuHandler..... 129
- __init__(on MenuMaker..... 60, 397
- __init__(on MetaModule..... 17
- __init__(on Modifier..... 34, 153
- __init__(on Module..... 17
- __init__(on ModuleIndexer..... 96, 373
- __init__(on ModuleInfo..... 127, 331
- __init__(on ModuleListing..... 97, 374
- __init__(on ModuleListingJS..... 97, 374
- __init__(on Named..... 33, 151
- __init__(on NameIndex..... 98, 375
- __init__(on NestedFileLayout..... 83, 363
- __init__(on Node..... 28
- __init__(on Operation..... 21
- __init__(on PackageBrowser..... 143
- __init__(on Page..... 100, 377
- __init__(on PageFormat..... 99, 376
- __init__(on PageManager..... 75, 357
- __init__(on Parameter..... 20
- __init__(on Parametrized..... 35, 154
- __init__(on ParserAction..... 11
- __init__(on ParserExecutor..... 26
- __init__(on ParserPage..... 137
- __init__(on Part..... 64, 347
- __init__(on Project..... 28
- __init__(on ProjectActions..... 29
- __init__(on ProjectReader..... 31
- __init__(on ProjectWindow..... 147
- __init__(on Python..... 6
- __init__(on PyWriter..... 39, 157
- __init__(on PyWriterStruct..... 40, 158
- __init__(on QtComments..... 111, 335
- __init__(on QtDocFormatter..... 72, 354
- __init__(on RawFilePages..... 102, 379
- __init__(on Scope..... 17
- __init__(on ScopePages..... 103, 379
- __init__(on ScopeSorter..... 104, 380
- __init__(on SectionFormatter..... 72, 354
- __init__(on SelectionStrategy..... 129
- __init__(on SimpleSourceRule..... 10
- __init__(on SingleInheritanceFormatter... 53, 392
- __init__(on SourceAction..... 11
- __init__(on SourceActionEditor..... 135
- __init__(on SourceBrowser..... 145
- __init__(on SourceEditExclude..... 148
- __init__(on SourceEditGlob..... 148
- __init__(on SourceEditSimple..... 148
- __init__(on SourceExecutor..... 25
- __init__(on SourceFile..... 14
- __init__(on SourceInsertWizard..... 149
- __init__(on SourceOptionsDialog..... 150
- __init__(on SourceRule..... 10
- __init__(on SSComments..... 111, 334
- __init__(on SSDComments..... 110, 334
- __init__(on Strategy..... 87, 366
- __init__(on Stripper..... 118, 341
- __init__(on Struct..... 58, 75, 135, 356, 396
- __init__(on SuiteInfoBase..... 125, 330
- __init__(on Summarizer..... 114, 337
- __init__(on Summary..... 68, 350
- __init__(on SystemError..... 51, 390
- __init__(on TableOfContents..... 55, 62, 393
- __init__(on Template..... 34, 153
- __init__(on TemplatePageFormat..... 100, 376
- __init__(on TexInfo..... 8
- __init__(on TocEntry..... 62
- __init__(on Transformer..... 112, 335
- __init__(on TreeFormatter..... 107, 383
- __init__(on TreeFormatterJS..... 106, 382
- __init__(on Type..... 32, 151
- __init__(on Typedef..... 18
- __init__(on TypeTranslator..... 122, 327
- __init__(on Unduplicator..... 119, 341
- __init__(on Unknown..... 33, 152
- __init__(on Variable..... 19
- __init__(on XRefCompiler..... 121, 343
- __init__(on XRefLinker..... 91, 108, 369, 384
- __init__(on XRefPages..... 108, 384
- __parseArgs(... 40, 42, 46, 48, 51, 54, 55, 58, 73, 116, 122, 128, 327, 332, 339, 345, 354, 385, 386, 388, 390, 393, 396
- __repr__(on Error..... 32, 151
- __repr__(on SystemError..... 51, 390
- __setitem__(on Dictionary..... 36, 155
- __str__(on Array..... 35, 154
- __str__(on Base..... 33, 152
- __str__(on Comment..... 23
- __str__(on Declared..... 34, 153
- __str__(on Dependent..... 33, 152
- __str__(on Modifier..... 35, 153
- __str__(on Parameter..... 20
- __str__(on Parametrized..... 36, 154
- __str__(on Template..... 34, 153
- __str__(on Unknown..... 34, 152
- _add_decl(on ScopeSorter..... 104, 381
- _addFromImport(on SuiteInfoBase..... 126, 330
- _addImport(on SuiteInfoBase..... 126, 330
- _checkMain(on FileLayout..... 81, 362
- _child_filter(on ModuleListing..... 98, 375

- `_child_filter`(on `ModuleListingJS` 97, 374
 - `_config_base_dir`(on `Config` 74, 355
 - `_config_comment_formatters`(on `Config` 74, 355
 - `_config_comment_processors`(on `Config` 116, 340
 - `_config_datadir`(on `Config` 73, 355
 - `_config_default_toc`(on `Config` 74, 355
 - `_config_exclude_globs`(on `Config` 74, 356
 - `_config_file_layout`(on `Config` 74, 355
 - `_config_map_declaration_names`(on `Config` 116, 339
 - `_config_map_declaration_type`(on `Config` .. 116, 339
 - `_config_mapper_list`(on `Config` 116, 339
 - `_config_max_access`(on `Config` 116, 339
 - `_config_operations`(on `Config` 116, 339
 - `_config_output_dir`(on `Config` 73, 355
 - `_config_page_format`(on `Config` 74, 356
 - `_config_pages`(on `Config` 73, 355
 - `_config_sorter`(on `Config` 73, 355
 - `_config_start_dir`(on `Config` 74, 355
 - `_config_strip`(on `Config` 116, 339
 - `_config_structs_as_classes`(on `Config` 74, 356
 - `_config_stylesheet`(on `Config` 73, 355
 - `_config_stylesheet_file`(on `Config` 73, 355
 - `_config_toc_in`(on `Config` 74, 355
 - `_config_toc_out`(on `Config` 74, 355
 - `_config_tree_formatter`(on `Config` 74, 355
 - `_config_verbose`(on `Config` 116, 339
 - `_convert_map`(..... 51, 390
 - `_copyAttrs`(on `AddWizard` 140
 - `_count_not_forwards`(on `EmptyNS` 115, 338
 - `_dictKey`(on `Dumper` 55, 385
 - `_extract_info`(on `SuiteInfoBase` 126, 330
 - `_fill_list`(on `DetailsPage` 139
 - `_find_link_at`(on `JavadocFormatter` 59, 72, 353, 397
 - `_find_method_entry`(on `JavadocFormatter` ... 60, 72, 354, 397
 - `_format`(..... 51, 390
 - `_format_html`(..... 51, 390
 - `_format_png`(..... 51, 390
 - `_get_children`(on `ModuleListing` 98, 375
 - `_get_files`(on `RawFilePages` 103, 379
 - `_get_pages`(on `ActionDialog` 139
 - `_get_parts`(on `ScopePages` 103, 380
 - `_get_timestamp`(on `CacherExecutor` 27
 - `_handle_group`(on `ScopeSorter` 105, 381
 - `_import`(..... 38, 156
 - `_init_default_formatters`(on `Detail` ... 68, 351
 - `_init_default_formatters`(on `Heading` .. 67, 350
 - `_init_default_formatters`(on `Inheritance` .. 69, 351
 - `_init_default_formatters`(on `Summary` .. 68, 350
 - `_init_formatters`(on `Part` 64, 347
 - `_init_page`(on `ModuleListingJS` 97, 374
 - `_instAttr`(on `Dumper` 55, 385
 - `_is_leaf`(on `ClassTree` 45
 - `_is_root`(on `ClassTree` 45
 - `_is_up_to_date`(on `CacherExecutor` 27
 - `_link_href`(on `ModuleListing` 98, 375
 - `_link_href`(on `ModuleListingJS` 97, 374
 - `_loadPages`(on `PageManager` 76, 357
 - `_make_graphs`(on `ClassTree` 45
 - `_make_layout`(on `CacherPage` 136
 - `_make_layout`(on `CppParserPage` 138
 - `_make_layout`(on `DetailsPage` 139
 - `_make_layout`(on `FormatterPage` 139
 - `_make_layout`(on `ParserPage` 137
 - `_make_left`(on `BrowserWindow` 141
 - `_make_right`(on `BrowserWindow` 141
 - `_makeCacherPage`(on `AddWizard` 140
 - `_makeDict`(on `NameIndex` 98, 375
 - `_makeFinishPage`(on `AddWizard` 140
 - `_makeFormatPropPage`(on `AddWizard` 140
 - `_makeFormatterPage`(on `AddWizard` 140
 - `_makeNewTool`(on `CanvasWindow` 134
 - `_makePageHeading`(on `ModuleIndexer` 96, 374
 - `_makeParserPage`(on `AddWizard` 140
 - `_makeSourcePage`(on `AddWizard` 140
 - `_makeStartPage`(on `AddWizard` 140
 - `_node_sorter`(on `FileListing` 85, 364
 - `_node_sorter`(on `FileTree` 86, 365
 - `_process_class`(on `Inheritance` 69, 351
 - `_process_superclasses`(on `Inheritance` 69, 351
 - `_processItem`(on `NameIndex` 99, 376
 - `_rel`(..... 51, 390
 - `_replace`(..... 58, 396
 - `_section_of`(on `DOScopeSorter` 78, 359
 - `_section_of`(on `ScopeSorter` 104, 381
 - `_set_section_names`(on `ScopeSorter` ... 105, 381
 - `_short_name`(on `Inheritance` 69, 351
 - `_sort_sections`(on `ScopeSorter` 104, 381
 - `_stripFilename`(on `FileLayout` 82, 362
 - `_stripName`(on `Stripper` 118, 341
 - `_take_control`(on `BufferPage` 102, 378
 - `_update_def_list`(on `CppParserPage` 138
 - `_update_path_list`(on `CppParserPage` 138
- ## A
- `accept`(on `Action` 9
 - `accept`(on `Array` 35, 154
 - `accept`(on `AST` 13
 - `accept`(on `Base` 33, 151
 - `accept`(on `CacherAction` 12
 - `accept`(on `Class` 18
 - `accept`(on `Const` 20
 - `accept`(on `Declaration` 15
 - `accept`(on `Declared` 34, 152
 - `accept`(on `Dependent` 33, 152

- accept(on Enum 19
 - accept(on Enumerator 19
 - accept(on FormatAction 12
 - accept(on Forward 16
 - accept(on Function 21, 36, 155
 - accept(on Group 17
 - accept(on Inheritance 18
 - accept(on LinkerAction 11
 - accept(on Macro 16
 - accept(on MetaModule 17
 - accept(on Modifier 35, 153
 - accept(on Module 17
 - accept(on Operation 21
 - accept(on Parameter 20
 - accept(on Parametrized 36, 154
 - accept(on ParserAction 11
 - accept(on Scope 17
 - accept(on SourceAction 11
 - accept(on Template 34, 153
 - accept(on Type 32, 151
 - accept(on Typedef 18
 - accept(on Unknown 34, 152
 - accept(on Variable 19
 - Accesson function 164
 - accessibility(on Declaration 15
 - action(on CacherPage 136
 - action(on CppParserPage 138
 - action(on DetailsPage 139
 - action(on FormatterPage 139
 - action(on ParserPage 137
 - action_added(on ActionCanvas 133
 - action_changed(on ActionCanvas 133
 - action_changed(on ProjectActions 30
 - action_moved(on ActionCanvas 133
 - action_removed(on ActionCanvas 133
 - actions(on Project 29
 - actions(on ProjectActions 29
 - activate(on BrowserWindow 142
 - activate(on CanvasWindow 134
 - activate(on ProjectWindow 147
 - add(on AccessRestrictor 109, 333
 - add(on EmptyNS 115, 338
 - add(on Transformer 112, 335
 - add(on TypeTranslator 122, 327
 - add_action(on ProjectActions 30
 - add_channel(on ProjectActions 30
 - add_class(on ClassTree 45
 - add_inheritance(on ClassTree 45
 - add_listener(on BrowserWindow 141
 - add_listener(on ProjectActions 29
 - add_params(..... 127, 331
 - add_sub(on Icon 145
 - add_super(on Icon 146
 - add_window(on MainWindow 147
 - AddCc2Option(..... 234
 - AddCcOption(..... 232
 - AddCppOption(..... 232
 - addDeclaration(..... 127, 331
 - addDeclaration(on ASTTranslator 123, 328
 - addDeclaration(on ListFiller 143
 - addDeclaration(on Unduplicator 120, 343
 - AddDirectoryButton_clicked(on SourceEditGlob 148
 - AddDirectoryButton_clicked(on SourceInsertWizard 150
 - AddFilesButton_clicked(on SourceEditSimple 149
 - AddFilesButton_clicked(on SourceInsertWizard 149
 - addFormatter(on Part 64, 347
 - addGroup(on ListFiller 143
 - addPage(on PageManager 76, 357
 - addRootPage(on PageManager 76, 357
 - addType(on ASTTranslator 123, 328
 - alias(on Array 35, 154
 - alias(on Modifier 35, 153
 - alias(on Typedef 18
 - anglebrackets(..... 106, 382
 - append(..... 237
 - append(on Unduplicator 119, 342
 - attribute(on DiaFormatter 47, 386
 - attributes(on Inheritance 18
- ## B
- BaseRadio_clicked(on SourceInsertWizard 149
 - Bind::Kindon function 261, 262, 263
 - boolon function 162, 165, 166, 172, 173, 183, 185, 189, 190, 195, 196, 197, 203, 239, 241, 244, 245, 250, 254, 256, 258, 259, 261, 262, 264, 265, 266, 267, 268, 271, 272, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 285, 286, 287, 288, 296, 313, 314, 315, 316, 318, 319, 326
- ## C
- calculateStart(on PageManager 76, 357
 - CancelButton_clicked(on SourceEditExclude 148
 - CancelButton_clicked(on SourceEditGlob .. 148
 - CancelButton_clicked(on SourceEditSimple 149
 - CancelButton_clicked(on SourceOptionsDialog 150
 - ccmp(..... 12, 32, 151
 - ccolonName(..... 38, 156
 - channel_added(on ActionCanvas 133
 - channel_removed(on ActionCanvas 133
 - charon function 248, 315
 - char>on function 186, 187
 - check_name(on ProjectActions 30
 - checkPrevious(on Previous 113, 336

- checkThreadOutput (on ProjectWindow..... 147
 - child (on ScopeSorter 105, 381
 - children (on ScopeSorter 105, 381
 - classes (on ClassTree 45
 - cleanup_temp_file() 121, 160
 - clear (on Dictionary 37, 155
 - clear (on Dumper 54, 385
 - clear (on Icon 146
 - clear (on ListFiller 142
 - clearType (on InheritanceFormatter 52, 391
 - clone (on ExcludeSourceRule 11
 - clone (on GlobSourceRule 10
 - clone (on SimpleSourceRule 10
 - clone (on SourceRule 10
 - close_file (on BufferPage 102, 379
 - close_file (on Page 101, 378
 - CloseCcOptions() 232
 - codeon function 188
 - comments (on Declaration 15
 - compare_sections (..... 104, 380
 - Compile (..... 233
 - compile_glob (..... 73, 354
 - compile_glob (on SourceExecutor 25
 - config (on FormatAction 12
 - config (on LinkerAction 11
 - config (on ParserAction 11
 - configure_for_gui (..... 73, 355
 - consolidate (on InheritanceGraph 94, 372
 - const AST::Declaration *on function 224
 - const AST::Declaration::vector &on function
..... 180
 - const Attributes &on function 170
 - const char *on function 206, 229
 - const Comment::vector &on function 164
 - const Declaration::vector &on function .. 166,
168
 - const Include::vector &on function 167
 - const Inheritance::vector &on function ... 170
 - const param_vector &on function 225
 - const Parameters *on function 168
 - const ScopedName &on function 163, 222
 - const std::string &on function .. 162, 163, 166,
168, 173, 175, 176, 177
 - const Type::Mods &on function 228
 - const Type::vector &on function 227, 228
 - const Types::Declared *on function 164
 - const Types::Type *on function 176
 - const vector &on function 225
 - constr (on Typedef 18
 - constr (on Variable 19
 - contentsMouseDoubleClickEvent (on CanvasView
..... 134
 - contentsMouseMoveEvent (on CanvasView 134
 - contentsMousePressEvent (on CanvasView ... 134
 - contentsMouseReleaseEvent (on CanvasView
..... 134
 - copyFile (on FileLayout 81, 361
 - CountArgs (..... 235
 - create (on ExecutorCreator 25
 - CreateClass (..... 235
 - CreateMetaClass (..... 234
 - createObjectID (on DiaFormatter 47, 387
 - CreateQuoteClass (..... 234
 - CreateTemplateClass (..... 235
 - ctype (on Const 19
 - current_ast (on BrowserWindow 142
 - current_ast_changed (on ClassBrowser 144
 - current_ast_changed (on DocoBrowser 144
 - current_ast_changed (on PackageBrowser ... 143
 - current_ast_changed (on SelectionListener
..... 141
 - current_ast_changed (on SourceBrowser 145
 - current_decl (on BrowserWindow 142
 - current_decl_changed (on DocoBrowser 144
 - current_decl_changed (on GraphBrowser 145
 - current_decl_changed (on SelectionListener
..... 141
 - current_decl_changed (on SourceBrowser ... 145
 - current_package_changed (on ClassBrowser
..... 143
 - current_package_changed (on SelectionListener
..... 141
 - currscope (on AccessRestrictor 109, 333
 - currscope (on EmptyNS 115, 338
 - currscope (on Transformer 112, 335
- ## D
- data (on SourceMimeFactory 144
 - data_dir (on Project 29
 - deactivate (on BrowserWindow 142
 - deactivate (on CanvasWindow 134
 - deactivate (on ProjectWindow 147
 - decl_label (on Formatter 60, 398
 - declaration (on Declared 34, 152
 - declarations (on AST 13
 - declarations (on Group 17
 - declarations (on SourceFile 14
 - declarations (on Stripper 118, 341
 - declarator (on Part 65, 348
 - declared_cast (..... 230
 - decode (..... 240
 - decr (on ASCIIFormatter 41, 345
 - decr (on DiaFormatter 46, 386
 - default_formatter (on Project 29
 - delete_all() 194
 - desc (..... 55, 106, 382, 393
 - describe_decl (on XRefPages 109, 384
 - DirList_selectionChanged (on SourceEditGlob
..... 148
 - DirList_selectionChanged (on
SourceInsertWizard 150
 - DirRadio_clicked (on SourceInsertWizard .. 149

DISABLED_current_package_changed(on
 PackageBrowser 143
 div(..... 55, 105, 382, 393
 dmatch(..... 125, 330
 do_compile(..... 120, 343
 do_function(on Formatter 44
 do_open_file(on MainWindow 147
 do_open_project(on MainWindow 146
 do_parse(..... 247
 doDiagramData(on DiaFormatter 47, 387
 doInheritance(on DiaFormatter 47, 387
 dotName(..... 38, 156
 doubleClick(on CanvasStrategy 129
 doubleClick(on SelectionStrategy 130
 dump_links() 240

E

edit(on SourceActionEditor 135
 emulate_compiler(..... 246
 end(on MenuMaker 60, 397
 end_entity(on Formatter 43, 49, 388
 end_file(on Page 102, 378
 end_file(on ScopePages 104, 380
 endTag(on DiaFormatter 47, 386
 endTree(on TreeFormatter 107, 383
 endTree(on TreeFormatterJS 107, 383
 ensure_import(on PyWriter 39, 157
 ensure_struct(on PyWriter 39, 157
 enterScope(on ASCIIFormatter 41, 345
 entity(..... 55, 106, 382, 393
 entity(on Formatter 43, 49, 388
 enumerators(on Enum 19
 escapify(on Formatter 60, 398
 escapifyString(..... 38, 156
 exceptions(on Function 21
 ExcludeRadio_clicked(on SourceInsertWizard
 149
 execute(on AccessRestrictor 109, 333
 execute(on Comments 114, 338
 execute(on EmptyNS 115, 338
 execute(on LanguageMapper 115, 339
 execute(on NameMapper 117, 340
 execute(on Operation 117, 340
 execute(on Stripper 118, 341
 execute(on Unduplicator 119, 341
 execute(on XRefCompiler 121, 343
 execute_project(on MainWindow 147
 execute_project(on ProjectWindow 147
 extend(..... 236
 extract(on JavadocFormatter... 59, 71, 353, 396

F

file(on Comment 23
 file(on Declaration 15
 FileList_selectionChanged(on
 SourceEditSimple 149
 FileList_selectionChanged(on
 SourceInsertWizard 150
 filename(on DirBrowse 77, 358
 filename(on FileDetails 80, 360
 filename(on FileIndexer 81, 361
 filename(on FileListing 84, 364
 filename(on FileSource 85, 364
 filename(on FileTree 86, 365
 filename(on FramesIndex 93, 371
 filename(on InheritanceGraph 94, 372
 filename(on InheritanceTree 95, 372
 filename(on ModuleIndexer 96, 373
 filename(on ModuleListing 97, 375
 filename(on ModuleListingJS 97, 374
 filename(on NameIndex 98, 375
 filename(on Page 100, 377
 filename(on Part 64, 347
 filename(on Project 28
 filename(on RawFilePages 102, 379
 filename(on ScopePages 103, 380
 filename(on SourceFile 14
 filename(on XRefPages 108, 384
 filename_for_dir(on DirBrowse 77, 358
 filename_info(on PageManager 77, 358
 files(on AST 13
 fillDefaults(on Config 73, 355
 fillFrom(on ListFiller 142
 filter_names(..... 125, 329
 filterName(..... 117, 340
 find_common_name(..... 93, 371
 find_compiler_info(..... 122, 160
 find_link(on JavadocFormatter 59, 71, 353,
 397
 findModulePath(..... 124, 329
 flatten_struct(on PyWriter 40, 158
 flush(on PyWriter 39, 157
 format(... 40, 42, 46, 48, 51, 54, 55, 58, 73, 125,
 329, 345, 354, 385, 386, 388, 390, 393, 396
 format(on CommentFormatter 70, 352
 format(on CommentFormatterStrategy ... 70, 352
 format(on JavadocFormatter 71, 353
 format(on QuoteHTML 70, 352
 format(on SectionFormatter 72, 354
 format_attrs(on JavadocFormatter 59, 71,
 353, 397
 format_inline_see(on JavadocFormatter ... 71,
 353
 format_params(on JavadocFormatter 59, 71,
 353, 396
 format_return(on JavadocFormatter 59, 71,
 353, 397

- format_see(on JavadocFormatter.... 59, 71, 353, 397
 - format_see(on QtDocFormatter..... 72, 354
 - format_source(..... 141
 - format_summary(on CommentFormatter... 70, 352
 - format_summary(on CommentFormatterStrategy..... 70, 352
 - format_summary(on QuoteHTML..... 70, 352
 - formatClass(on BaseAST..... 89, 368
 - formatClass(on ClassHierarchyGraph... 93, 370
 - formatClass(on ClassHierarchySimple.. 93, 370
 - formatClass(on Default..... 90, 369
 - formatClass(on Heading..... 92, 370
 - formatClass(on Strategy..... 88, 367
 - formatComments(on Formatter..... 61, 398
 - formatConst(on BaseAST..... 89, 368
 - formatConst(on Default..... 90, 369
 - formatConst(on Strategy..... 88, 367
 - formatDeclaration(on BaseAST..... 89, 367
 - formatDeclaration(on DetailCommenter.... 92, 370
 - formatDeclaration(on DOSummaryCommenter.. 78, 359
 - formatDeclaration(on Inheritance..... 93, 371
 - formatDeclaration(on Part..... 65, 348
 - formatDeclaration(on PostDivFormatter.... 78, 359
 - formatDeclaration(on PostSummaryDiv.. 79, 359
 - formatDeclaration(on PreDivFormatter..... 78, 359
 - formatDeclaration(on PreSummaryDiv... 79, 359
 - formatDeclaration(on SourceLinker.... 91, 369
 - formatDeclaration(on Strategy..... 88, 366
 - formatDeclaration(on SummaryCommenter.... 91, 369
 - formatDeclaration(on XRefLinker..... 91, 369
 - formatEnum(on BaseAST..... 89, 368
 - formatEnum(on Default..... 90, 369
 - formatEnum(on DetailAST..... 92, 370
 - formatEnum(on DOSummaryAST..... 78, 359
 - formatEnum(on Strategy..... 88, 367
 - formatEnumerator(on BaseAST..... 89, 368
 - formatEnumerator(on DetailAST..... 92, 370
 - formatForward(on BaseAST..... 89, 367
 - formatForward(on Default..... 90, 368
 - formatForward(on Strategy..... 88, 366
 - formatFunction(on BaseAST..... 89, 368
 - formatFunction(on Default..... 91, 369
 - formatFunction(on DODetailAST..... 78, 359
 - formatFunction(on Inheritance..... 93, 371
 - formatFunction(on Strategy..... 88, 367
 - formatGroup(on BaseAST..... 89, 367
 - formatGroup(on Default..... 90, 368
 - formatGroup(on Strategy..... 88, 367
 - formatGroup(on SummaryCommenter..... 91, 369
 - formatHeader(on PageManager..... 76, 357
 - formatImage(on JSTree..... 95, 373
 - formatImage(on TreeFormatterJS..... 106, 382
 - formatInheritance(on ClassHierarchySimple..... 93, 370
 - formatMetaModule(on BaseAST..... 89, 367
 - formatMetaModule(on Default..... 90, 368
 - formatMetaModule(on Heading..... 92, 370
 - formatMetaModule(on Strategy..... 88, 367
 - formatModifiers(on Strategy..... 87, 366
 - formatModule(on BaseAST..... 89, 367
 - formatModule(on Default..... 90, 368
 - formatModule(on Heading..... 92, 370
 - formatModule(on Strategy..... 88, 367
 - formatName(on Heading..... 91, 369
 - formatNameInNamespace(on Heading.... 91, 369
 - formatNamespaceOfName(on Heading.... 92, 370
 - formatOperation(on BaseAST..... 90, 368
 - formatOperation(on Default..... 91, 369
 - formatOperation(on Inheritance..... 93, 371
 - formatOperation(on Strategy..... 88, 367
 - formatOperationExceptions(on DetailAST... 92, 370
 - formatOperationExceptions(on SummaryAST.. 90, 368
 - formatParameter(on BaseAST..... 90, 368
 - formatParameter(on Heading..... 92, 370
 - formatParameters(on BaseAST..... 89, 367
 - formatScope(on BaseAST..... 89, 367
 - formatScope(on Default..... 90, 368
 - formatScope(on Strategy..... 88, 367
 - formatType(on ASCIIFormatter..... 41, 345
 - formatType(on DiaFormatter..... 47, 387
 - formatType(on Formatter..... 44, 61, 398
 - formatType(on InheritanceFormatter... 52, 391
 - formatType(on Part..... 66, 349
 - formatTypedef(on BaseAST..... 89, 368
 - formatTypedef(on Default..... 90, 369
 - formatTypedef(on Strategy..... 88, 367
 - formatVariable(on BaseAST..... 89, 368
 - formatVariable(on Default..... 90, 369
 - formatVariable(on Strategy..... 88, 367
 - full_filename(on SourceFile..... 14
- ## G
- generator(on DocoBrowser..... 144
 - generator(on SourceBrowser..... 145
 - get(on TypeTranslator..... 123, 327
 - get_action(on ProjectActions..... 29
 - get_action_at(on ActionCanvas..... 133
 - get_all_names(on CrossReferencer..... 63
 - get_base_names(on ClassInfo..... 127, 331
 - get_buffer(on BufferPage..... 102, 379
 - get_cache_filename(on CacherExecutor..... 27
 - get_children(on ModuleListingJS..... 97, 374
 - get_class_info(on SuiteInfoBase.... 126, 330
 - get_class_names(on SuiteInfoBase.... 126, 330
 - get_compiler_info(..... 121, 160

get_compiler_timestamp(..... 122, 160
 get_docs(..... 125, 329
 get_docstring(on SuiteInfoBase..... 126, 330
 get_fallback(..... 121, 160
 get_function_info(on SuiteFuncInfo .. 126, 330
 get_function_names(on SuiteFuncInfo 126,
 330
 get_icon_for(on ActionCanvas 133
 get_info(on CrossReferencer 63
 get_line_at(on ActionCanvas 133
 get_linker(on LinkerExecutor 26
 get_method_info(on ClassInfo..... 127, 330
 get_method_names(on ClassInfo..... 127, 330
 get_mime_data(on DocoBrowser 144
 get_name(on SuiteInfoBase 126, 330
 get_names_only(..... 125, 329
 get_output(on CacherExecutor 27
 get_output(on Executor 24
 get_output(on FormatExecutor 27
 get_output(on LinkerExecutor 26
 get_output(on ParserExecutor 26
 get_output(on SourceExecutor 25
 get_output_names(on CacherExecutor..... 26
 get_output_names(on Executor 24
 get_output_names(on FormatExecutor..... 27
 get_output_names(on LinkerExecutor..... 26
 get_output_names(on ParserExecutor..... 26
 get_output_names(on SourceExecutor..... 25
 get_page_for(on CrossReferencer..... 63
 get_page_info(on CrossReferencer..... 63
 get_param_defaults(on FunctionInfo .. 126, 330
 get_params(on FunctionInfo..... 126, 330
 get_parser(on ParserExecutor 26
 get_possible_names(on CrossReferencer..... 63
 get_selected_index(on SourceActionEditor
 136
 get_synopsis(..... 128, 332
 get_temp_file() 121, 160
 get_toc(on Page 101, 377
 get_toc(on ScopePages 103, 380
 get_toc(on XRefPages 108, 384
 getClassname(on InheritanceFormatter..... 52,
 391
 getId(on JSTree 95, 373
 getId(on TreeFormatterJS 106, 382
 getObjectID(on DiaFormatter..... 47, 387
 getopt_spec(..... 39, 157
 getopts(..... 246
 getPage(on PageManager 75, 357
 getType(on ASTTranslator 123, 328
 globalScope(on PageManager 76, 357
 graphs(on ClassTree 45

H

has_key(on Dictionary 36, 155
 HashValueon function..... 264
 hide(on Icon 132
 hide(on Line 132
 highlighted(on SourceBrowser 145
 hilite(on SelectionStrategy 129
 href(..... 55, 105, 381, 393

I

identifier(on Parameter..... 20
 import_object(..... 38, 156
 IncludeRadio_clicked(on SourceInsertWizard
 149
 includes(on SourceFile..... 15
 incr(on ASCIIFormatter 41, 345
 incr(on DiaFormatter 46, 386
 indent(on ASCIIFormatter..... 40, 345
 indent(on DiaFormatter 46, 386
 indent(on Dumper 54, 385
 indent(on PyWriter..... 39, 157
 indexModule(on ModuleListing..... 98, 375
 indexModule(on ModuleListingJS..... 97, 374
 init(on SourceInsertWizard 149
 init(on SourceOptionsDialog 150
 init_dialog(on SourceActionEditor..... 135
 InitializeOtherKeywords() 236
 initlink() 241
 initocc() 247
 inputs(on Action 9
 insert(on TableOfContents..... 56, 62, 393
 inton function 162, 163, 198, 201, 206, 207,
 213, 216, 245, 256, 259, 264, 265, 266, 269,
 271, 274, 282, 283, 284, 289, 290, 291, 292,
 293, 294, 295, 296, 297, 298, 299, 300, 301,
 302, 303, 304, 305, 306, 307, 308, 309, 310,
 311, 312, 313, 314, 315, 316
 internalize(on TypeTranslator..... 122, 327
 is_blank(..... 316
 is_digit(..... 316
 is_duplicate(..... 240
 is_eletter(..... 316
 is_float_suffix(..... 316
 is_hexdigit(..... 316
 is_int_suffix(..... 316
 is_letter(..... 316
 is_macro(on Include..... 14
 is_main(on SourceFile..... 14
 is_multi(on ParserExecutor 26
 is_next(on Include 14
 is_project_file(..... 28
 is_suspect(on Comment..... 23
 is_valid_channel(on ProjectActions..... 31
 is_xletter(..... 316
 IsCxxSource(..... 233
 isStructor(..... 237

items(on Dictionary 37, 155

J

join(..... 186
 js_init(on JSTree..... 95, 373
 js_init(on TreeFormatterJS..... 106, 382

K

k2a(..... 46, 105, 381, 386
 keep_changes(on SourceActionEditor..... 135
 key(on CanvasStrategy..... 129
 key(on SelectionStrategy..... 130
 keyPressEvent(on CanvasView..... 134
 keys(on Dictionary..... 36, 155
 Kindon function 261

L

label(on Formatter 43, 49, 388
 label(on HTMLFormatter 57, 394
 label(on Part 65, 348
 label(on Summary..... 68, 350
 language(on Declaration..... 15
 language(on SourceFile..... 14
 language(on Type..... 32, 151
 layOutTitleRow(on AddWizard 141
 leaves(on ClassTree 45
 leaveScope(on ASCIIFormatter 41, 345
 line(on Comment 23
 line(on Declaration..... 15
 link(on FileLayout..... 83, 363
 link(on Linker 62
 link(on Unknown 33, 152
 link(on XRefLinker..... 108, 384
 link_file() 240
 linkType(on Unduplicator 119, 342
 lnwrite(on Dumper..... 54, 385
 load(..... 12
 load(on CrossReferencer..... 63
 load(on Project 29
 load(on TableOfContents..... 62
 load_compiler_infos() 121, 160
 load_deps(..... 12
 load_file(on BrowserWindow 142
 load_file(on TemplatePageFormat 100, 376
 LoadMetaclass(..... 233
 LoadSoLib(..... 233
 long(on PyWriter..... 39, 157
 lookup(on Dictionary 37, 155
 lookup(on TableOfContents..... 55, 62, 393
 lookup(on Unduplicator 119, 342
 LookupSymbol(..... 233

M

main(..... 234
 main() 121, 160
 make_code(..... 187
 make_Comment(..... 247
 make_deps(..... 13
 make_Leaf(..... 247
 make_relative(..... 135
 makedirs(..... 242
 MakeRelativeButton_clicked(on SourceEditGlob
 148
 MakeRelativeButton_clicked(on
 SourceEditSimple..... 149
 MakeTempFilename(..... 233
 map(on CXX2IDL 117, 340
 map_rest(..... 125, 329
 map_second(..... 125, 329
 mapTypes(..... 116, 339
 match(..... 125, 329
 merge(on AST..... 13
 merge(on Dictionary 37, 155
 merge_comments(on Unduplicator..... 120, 342
 mid(on Icon..... 145
 modeChanged(on CanvasView 134
 module_declarations(on MetaModule 17
 MopErrorMessage(..... 235
 MopErrorMessage2(..... 235
 MopMoreWarningMessage(..... 235
 MopWarningMessage(..... 235
 MopWarningMessage2(..... 235
 move(on AddActionStrategy 131
 move(on CanvasStrategy..... 129
 move(on ConnectStrategy..... 130
 move(on SelectionStrategy 129
 move_action(on ProjectActions 30
 move_action_by(on ProjectActions..... 30
 move_by(on Action 9
 move_sub(on Icon 146
 move_super(on Icon 146
 move_to(on Action 9
 move_to(on Icon 145

N

name(..... 55, 105, 381, 393
 name(on Action 9
 name(on CommentTag 22
 name(on Declaration..... 15
 name(on Named 33, 151
 name(on Project 29
 named_cast(..... 230
 nameOfFileDetails(on FileLayout..... 82, 362
 nameOfFileDetails(on NestedFileLayout 83,
 363
 nameOfFileIndex(on FileLayout..... 82, 362
 nameOfFileIndex(on NestedFileLayout .. 83, 363
 nameOfFileSource(on FileLayout..... 82, 362

nameOfFileSource(on NestedFileLayout 83, 363
 nameOfIndex(on FileLayout 82, 362
 nameOfIndex(on NestedFileLayout 84, 363
 nameOfModuleIndex(on FileLayout 83, 362
 nameOfModuleIndex(on NestedFileLayout 84, 363
 nameOfModuleTree(on FileLayout 83, 362
 nameOfModuleTree(on NestedFileLayout 84, 363
 nameOfScope(on FileLayout 82, 362
 nameOfScope(on NestedFileLayout 83, 363
 nameOfScopedSpecial(on FileLayout 82, 362
 nameOfScopedSpecial(on NestedFileLayout.. 84, 363
 nameOfSpecial(on FileLayout 82, 362
 nameOfSpecial(on NestedFileLayout 84, 363
 NameScopeon function 325
 new_project(on MainWindow 147
 newCacheAction(on CanvasWindow 135
 newFormatterAction(on CanvasWindow 135
 newLinkerAction(on CanvasWindow 135
 newParserAction(on CanvasWindow 135
 newSourceAction(on CanvasWindow 135
 nullObj() 248

O

occParse(..... 247
 occUsage(..... 247
 OkButton_clicked(on SourceEditExclude ... 148
 OkButton_clicked(on SourceEditGlob 148
 OkButton_clicked(on SourceEditSimple 149
 OkButton_clicked(on SourceOptionsDialog 150
 old_init_formatters(on DODetail 79, 360
 old_init_formatters(on DOSummary 79, 359
 old_reference(..... 72, 354
 on_default_formatter(on MenuHandler 129
 on_default_formatter(on SelectionStrategy 130
 on_delete(on SourceActionEditor 136
 on_delete_action(on MenuHandler 129
 on_delete_action(on SelectionStrategy ... 130
 on_delete_line(on MenuHandler 129
 on_delete_line(on SelectionStrategy 130
 on_down(on SourceActionEditor 136
 on_edit(on SourceActionEditor 136
 on_insert(on SourceActionEditor 136
 on_insert_finished(on SourceActionEditor 136
 on_properties(on MenuHandler 129
 on_properties(on SelectionStrategy 130
 on_selection(on SourceActionEditor 136
 on_test(on SourceActionEditor 136
 on_up(on SourceActionEditor 136
 onActionName(on AddWizard 140

onActionName(on CacherPage 137
 onActionName(on FormatterPage 139
 onActionName(on ParserPage 137
 onActionType(on AddWizard 140
 onAddDefine(on CppParserPage 138
 onAddGCC(on CppParserPage 138
 onAddPath(on CppParserPage 138
 onBrowseDir(on CacherPage 137
 onBrowseFile(on CacherPage 137
 onButtonDir(on CacherPage 137
 onButtonFile(on CacherPage 137
 onDefSelected(on CppParserPage 138
 onDirChanged(on CacherPage 137
 onFileChanged(on CacherPage 137
 onFormatModule(on AddWizard 140
 onMainFile(on CppParserPage 138
 onModuleSelected(on ParserPage 137
 onPathSelected(on CppParserPage 138
 onRemoveDefine(on CppParserPage 138
 onRemovePath(on CppParserPage 138
 onSourceAddPath(on AddWizard 140
 opcxx_init_Class() 235
 opcxx_init_MetaClass() 234
 opcxx_init_QuoteClass() 234
 opcxx_init_TemplateClass() 235
 open(..... 39, 157
 open_file(on BufferPage 102, 378
 open_file(on MainWindow 147
 open_file(on Page 101, 378
 open_project(on MainWindow 146
 OpenCxxOutputFileName(..... 233
 openGraph(on BrowserWindow 142
 operator<<(..... 186, 188, 231, 239, 243, 286
 organize(on IGraphWindow 146
 os(on Page 100, 377
 os(on Part 64, 347
 outdent(on Dumper 54, 385
 outdent(on PyWriter 39, 157
 output(on DiaFormatter 47, 387
 outputs(on Action 9

P

page(on Part 64, 347
 page_footer(on PageFormat 99, 376
 page_footer(on TemplatePageFormat ... 100, 377
 page_header(on PageFormat 99, 376
 page_header(on TemplatePageFormat ... 100, 377
 parameter(on InheritanceFormatter 51, 391
 parameter(on Part 65, 348
 parameters(on Function 21, 36, 155
 parameters(on Macro 16
 parameters(on Parametrized 36, 154
 parameters(on Template 34, 153
 parent(on Inheritance 17
 parents(on Class 18
 parse(..... 122, 128, 327, 332

- parse(on CommentFormatter 58, 396
 - parse(on Escapifier 60, 397
 - parse(on JavadocFormatter 58, 396
 - parse_args(..... 240
 - parse_see(on JavadocFormatter 59, 396
 - parse_ss(on SSSComments 111, 335
 - parse_ssd(on SSDComments 110, 334
 - ParseCcOptions(..... 234
 - ParseCmdOptions(..... 232
 - ParseOpencxx(..... 234
 - ParseOptions(..... 234
 - parseTags(on JavadocFormatter 59, 396
 - ParseTargetSpecificOptions(..... 233
 - parseText(on JavadocFormatter 59, 396
 - parseText(on QtDocFormatter 72, 354
 - pop(on AccessRestrictor 109, 333
 - pop(on EmptyNS 115, 338
 - pop(on Previous 113, 336
 - pop(on Transformer 112, 335
 - pop(on Unduplicator 119, 342
 - pop() 127, 331
 - pop_only(on EmptyNS 115, 338
 - pop_scope(on Formatter 43, 48, 388
 - pos(on Action 9
 - postmod(on Modifier 35, 153
 - postmodifier(on Function 21
 - postmodifier(on Parameter 20
 - pre_show(on CacherPage 136
 - pre_show(on CppParserPage 138
 - pre_show(on DetailsPage 139
 - pre_show(on FormatterPage 139
 - pre_show(on ParserPage 137
 - prefix(on PageFormat 99, 376
 - premod(on Function 36, 154
 - premod(on Modifier 35, 153
 - premodifier(on Function 21
 - premodifier(on Parameter 20
 - prepare_output(on CacherExecutor 27
 - prepare_output(on Executor 24
 - press(on CanvasStrategy 128
 - press(on ConnectStrategy 130
 - press(on SelectionStrategy 129
 - print_types(..... 40, 345
 - process(on CommentProcessor 110, 333
 - process(on Detail 69, 351
 - process(on DirBrowse 77, 358
 - process(on FileDetails 80, 360
 - process(on FileIndexer 81, 361
 - process(on FileListing 85, 364
 - process(on FileSource 85, 365
 - process(on FileTree 86, 365, 366
 - process(on FramesIndex 93, 371
 - process(on Heading 67, 350
 - process(on Inheritance 69, 351
 - process(on InheritanceGraph 94, 372
 - process(on InheritanceTree 95, 372
 - process(on JavaComments 110, 334
 - process(on JavaTags 114, 337
 - process(on ModuleIndexer 96, 373
 - process(on ModuleListing 98, 375
 - process(on ModuleListingJS 97, 374
 - process(on NameIndex 98, 375
 - process(on Page 101, 378
 - process(on PageManager 76, 357
 - process(on Part 65, 348
 - process(on QtComments 111, 335
 - process(on RawFilePages 103, 379
 - process(on ScopePages 104, 380
 - process(on SSSComments 111, 334
 - process(on SSDComments 110, 334
 - process(on Summarizer 114, 337
 - process(on Summary 68, 351
 - process(on XRefPages 108, 384
 - process_ClassInfo(..... 128, 332
 - process_dir(on DirBrowse 77, 358
 - process_file(on RawFilePages 103, 379
 - process_FunctionInfo(..... 128, 331
 - process_link(on XRefPages 109, 384
 - process_MethodInfo(..... 128, 331
 - process_ModuleInfo(..... 127, 331
 - process_name(on XRefPages 109, 384
 - process_node(on FileSource 85, 365
 - process_scope(on FileDetails 80, 361
 - process_scope(on FileIndexer 81, 361
 - process_scope(on Page 101, 378
 - process_scope(on ScopePages 104, 380
 - processAll(on CommentProcessor 110, 333
 - processAll(on Previous 113, 336
 - processAll(on Transformer 112, 335
 - processClassInheritance(on InheritanceTree 95, 372
 - processComment(on JavaComments 111, 334
 - processComment(on QtComments 111, 335
 - processComment(on SSSComments 111, 334
 - processComment(on SSDComments 110, 334
 - processFileTreeNode(on FileListing ... 85, 364
 - processFileTreeNode(on FileTree 86, 365
 - processFileTreeNodePage(on FileTree .. 86, 365
 - processNamespaceIndex(on ModuleIndexer ... 96, 374
 - project(on ExecutorCreator 24
 - pruneScope(..... 38, 156
 - push(..... 127, 331
 - push(on AccessRestrictor 109, 333
 - push(on EmptyNS 115, 338
 - push(on Previous 113, 336
 - push(on Transformer 112, 335
 - push(on Unduplicator 119, 342
 - push_scope(on Formatter 43, 48, 388
 - py_link(..... 241
- Q**
- quote(..... 39, 46, 157, 386

R

- rank_down(on IGraphWindow 146
- read(on ProjectReader 31
- read_CacherAction(on ProjectReader 32
- read_FormatAction(on ProjectReader 32
- read_LinkerAction(on ProjectReader 32
- read_links() 240
- read_ParserAction(on ProjectReader 32
- read_Project(on ProjectReader 31
- read_ProjectActions(on ProjectReader 32
- read_SourceAction(on ProjectReader 32
- read_SourceRule(on ProjectReader 32
- read_tocs() 240
- ReadFile(..... 233
- ReadStdin() 233
- realname(on Function 21
- RecordCmdOption(..... 234
- reference(on Formatter 43, 49, 388
- reference(on HTMLFormatter 57, 394
- reference(on Page 102, 378
- reference(on Part 65, 348
- refresh_compiler_infos(..... 122, 160
- register(on DirBrowse 77, 358
- register(on FileListing 84, 364
- register(on FileTree 86, 365
- register(on FramesIndex 93, 371
- register(on InheritanceGraph 94, 372
- register(on InheritanceTree 95, 372
- register(on ModuleIndexer 96, 373
- register(on ModuleListing 97, 375
- register(on NameIndex 98, 375
- register(on Page 101, 377
- register_filename(on PageManager 76, 358
- register_filenames(on DirBrowse 77, 358
- register_filenames(on FileDetails 80, 360
- register_filenames(on FileIndexer 81, 361
- register_filenames(on FileListing 84, 364
- register_filenames(on FileSource 85, 365
- register_filenames(on FileTree 86, 366
- register_filenames(on Page 101, 377
- register_filenames(on RawFilePages .. 103, 379
- register_filenames(on ScopePages 104, 380
- register_filenames(on XRefPages 109, 384
- rel(..... 105, 381
- RelativeCurrentRadio_clicked(on
 - SourceInsertWizard 150
- RelativeNoneRadio_clicked(on
 - SourceInsertWizard 150
- RelativeThisButton_clicked(on
 - SourceInsertWizard 150
- RelativeThisRadio_clicked(on
 - SourceInsertWizard 150
- release(on AddActionStrategy 131
- release(on CanvasStrategy 128
- release(on ConnectStrategy 130
- release(on SelectionStrategy 129
- remove_action(on ProjectActions 30
- remove_channel(on ProjectActions 30
- remove_window(on MainWindow 147
- RemoveDirectoryButton_clicked(on
 - SourceEditGlob 148
- RemoveDirectoryButton_clicked(on
 - SourceInsertWizard 150
- RemoveFileButton_clicked(on SourceEditSimple
 - 149
- RemoveFileButton_clicked(on
 - SourceInsertWizard 150
- removeSuspect(on Previous 113, 336
- rename_action(on ProjectActions 30
- replace_spaces(..... 106, 382
- reset(on AddActionStrategy 131
- reset(on CanvasStrategy 128
- reset(on ConnectStrategy 130
- reset(on SelectionStrategy 129
- reset() 241
- resizeEvent(on CanvasWindow 134
- resolve(..... 116, 339
- resolve(on Unknown 33, 152
- returnType(on Function 21, 36, 154
- root(on FileTree 28
- roots(on ClassTree 45
- rules(on SourceAction 11
- RunCompiler(..... 233
- RunLinker() 233
- RunOpencxx(..... 234, 246
- RunPreprocessor(..... 233, 246
- RunSoCompiler(..... 233

S

- save(..... 13
- save(on Project 29
- saveProject(on CanvasWindow 135
- saveProjectAs(on CanvasWindow 135
- scope(on ASCIIFormatter 41, 345
- scope(on ASTTranslator 123, 328
- scope(on DiaFormatter 46, 386
- scope(on Formatter 43, 48, 60, 388, 398
- scope(on HTMLFormatter 56, 394
- scope(on InheritanceFormatter 51, 390
- scope(on Part 64, 347
- scope(on ScopePages 104, 380
- ScopedNameon function 188
- scopeName(..... 127, 331
- sections(on ScopeSorter 105, 381
- select_decl_item(on ClassBrowser 143
- select_package_item(on PackageBrowser ... 143
- SelectFilesButton_clicked(on
 - SourceInsertWizard 149
- selfish_expansion(on ClassBrowser 144
- set(on AddActionStrategy 131
- set(on CanvasStrategy 128
- set(on ConnectStrategy 130
- set_accessibility(on Declaration 16

- set_action(on CacherPage 136
- set_action(on CppParserPage 138
- set_action(on DetailsPage 139
- set_action(on FormatterPage 139
- set_action(on ParserPage 137
- set_alias(on Array 35, 154
- set_alias(on Modifier 35, 153
- set_alias(on Typedef 18
- set_ast(on FileTree 28
- set_browser(on SourceMimeFactory 144
- set_class(on IGraphWindow 146
- set_config(on FormatAction 12
- set_config(on LinkerAction 11
- set_config(on ParserAction 11
- set_contents_page(on Config 74, 356
- set_ctype(on Const 20
- set_current_ast(on BrowserWindow 142
- set_current_decl(on BrowserWindow 142
- set_current_package(on BrowserWindow 142
- set_data_dir(on Project 29
- set_default_formatter(on Project 29
- set_filename(on Project 28
- set_hilite(on Icon 132
- set_hilite(on Line 132
- set_index_page(on Config 74, 356
- set_is_main(on SourceFile 14
- set_link_detail(on Summary 68, 350
- set_main_page(on Config 75, 356
- set_name(on Action 9
- set_name(on Declaration 16
- set_name(on Named 33, 151
- set_name(on Project 29
- set_parent(on Inheritance 18
- set_prefix(on PageFormat 99, 376
- set_returnType(on Function 21, 36, 155
- set_scope(on ASCIIFormatter 41, 345
- set_scope(on ScopeSorter 104, 380
- set_summary(on Comment 22
- set_suspect(on Comment 23
- set_target(on Include 14
- set_template(on Class 18
- set_template(on Function 21
- set_template(on Parametrized 36, 154
- set_text(on Comment 22
- set_type(on Parameter 20
- set_using_module_index(on Config 75, 356
- set_verbose(on Project 29
- set_vtype(on Variable 19
- setConnect(on CanvasWindow 135
- setDest(on ConnectStrategy 130
- setGraphEnabled(on BrowserWindow 142
- setMode(on CanvasWindow 134
- setSelect(on CanvasWindow 134
- setSource(on ConnectStrategy 130
- ShowCommandLine(..... 232
- showEvent(on CacherPage 136
- showEvent(on CppParserPage 138
- showEvent(on FormatterPage 139
- showEvent(on ParserPage 137
- ShowHelp(..... 234
- showPage(on AddWizard 140
- ShowVersion() 234
- sighandler(..... 246
- SimpleRadio_clicked(on SourceInsertWizard
..... 149
- size(on TableOfContents 62
- sizes(on Array 35, 154
- SkipSpaces(..... 235
- slashName(..... 37, 156
- solotag(..... 106, 382
- soloTag(on DiaFormatter 47, 386
- sort(..... 44, 72, 354
- sort_ranks(on IGraphWindow 146
- sort_section_names(on DOScopeSorter .. 78, 359
- sort_section_names(on ScopeSorter ... 105, 381
- sort_sections(on ScopeSorter 105, 381
- space_ranks(on IGraphWindow 146
- span(..... 55, 105, 381, 393
- splitAndStrip(..... 38, 157
- start(on MenuMaker 60, 397
- start_entity(on Formatter 43, 48, 388
- start_file(on JSTree 95, 373
- start_file(on Page 101, 378
- startTag(on DiaFormatter 46, 386
- startTagDict(on DiaFormatter 46, 386
- startTree(on TreeFormatter 107, 383
- startTree(on TreeFormatterJS 106, 382
- std::stringon function 185, 188, 189, 191,
192, 196, 205, 207, 209, 242
- std::vector< AST::Inheritance *>on function
..... 209
- store(on TableOfContents 62
- stringify(..... 125, 329
- strip(..... 122, 327
- strip(on Stripper 118, 341
- strip_filename(..... 122, 327
- strip_star(on SSComments 111, 334
- strip_star(on SSDComments 110, 334
- stripDeclarations(on Stripper 118, 341
- stripTypes(on Stripper 118, 341
- style_of(on DeclStyle 75, 356
- stylesheet(on PageFormat 99, 376
- subclasses(on ClassTree 45
- summary(on Comment 22
- superclasses(on ClassTree 45
- synopsis_define_hook(..... 242
- synopsis_include_hook(..... 241
- synopsis_macro_hook(..... 241
- system(..... 50, 390

T

tabChanged(on BrowserWindow 142
 tags(on Comment 22
 target(on Include 14
 template(on Class 18
 template(on Function 21
 template(on Parametrized 35, 154
 text(on Comment 22
 text(on CommentTag 22
 text(on Macro 16
 title(on CacherPage 136
 title(on CppParserPage 138
 title(on DetailsPage 139
 title(on DirBrowse 77, 358
 title(on FileDetails 80, 360
 title(on FileIndexer 81, 361
 title(on FileListing 84, 364
 title(on FileSource 85, 365
 title(on FileTree 86, 365
 title(on FormatterPage 139
 title(on FramesIndex 93, 371
 title(on InheritanceGraph 94, 372
 title(on InheritanceTree 95, 372
 title(on ModuleIndexer 96, 373
 title(on ModuleListing 97, 375
 title(on ModuleListingJS 97, 374
 title(on NameIndex 98, 375
 title(on Page 100, 377
 title(on ParserPage 137
 title(on RawFilePages 103, 379
 title(on ScopePages 103, 380
 title(on XRefPages 108, 384
 top(on Unduplicator 119, 342
 top_dict(on Unduplicator 119, 342
 type(on Declaration 15
 type(on Inheritance 17
 type(on Parameter 20
 type_cast(..... 229
 type_label(on Formatter 43, 49, 60, 388, 398
 type_label(on InheritanceFormatter ... 51, 391
 type_label(on Part 65, 348
 type_ref(on InheritanceFormatter 51, 391
 type_ref(on Part 64, 347
 Type_vector(on function 190
 types(on AST 13

U

ucpp_main(..... 245
 uinton function 231, 248, 260, 265, 273, 274,
 287, 314, 315
 un_offset(on IGraphWindow 146
 unexpected(..... 246
 unsigned char *on function 256
 unsigned int on function 265
 update_list(on SourceActionEditor 136
 update_pos(on Icon 132

update_pos(on Line 132
 usage(.... 40, 42, 46, 48, 50, 54, 55, 58, 73, 116,
 122, 128, 327, 332, 339, 345, 354, 385, 386,
 388, 390, 393, 396
 use_config(on Config 73, 116, 339, 355
 uselocale(..... 236

V

value(on Const 20
 value(on Enumerator 19
 value(on Parameter 20
 values(on Dictionary 37, 155
 verbose(on Project 29
 visit(on Dumper 54, 385
 visitAction(on ActionColorizer 131
 visitAction(on ActionIcon 132
 visitAction(on ActionVisitor 9
 visitAction(on ExecutorCreator 25
 visitArray(on ToDecl 94, 371
 visitArray(on Unduplicator 119, 342
 visitArray(on Visitor 37, 155
 visitAST(on ASTTranslator 123, 328
 visitAST(on ClassTree 46
 visitAST(on TableOfContents 56, 62, 393
 visitAST(on Visitor 23
 visitAttribute(on ASTTranslator 124, 328
 visitBase(on InheritanceFormatter 52, 391
 visitBaseType(on ASCIIFormatter 41, 345
 visitBaseType(on CollaborationFormatter.. 53,
 392
 visitBaseType(on DiaFormatter 47, 387
 visitBaseType(on Formatter 43, 49, 61, 389,
 398
 visitBaseType(on HTMLFormatter 57, 394
 visitBaseType(on Part 66, 349
 visitBaseType(on ToDecl 94, 371
 visitBaseType(on TypeTranslator 123, 327
 visitBaseType(on Unduplicator 119, 342
 visitBaseType(on Visitor 37, 155
 visitCacher(on ActionColorizer 131
 visitCacher(on ActionIcon 132
 visitCacher(on ActionVisitor 10
 visitCacher(on ExecutorCreator 25
 visitClass(on ASCIIFormatter 42, 346
 visitClass(on ClassTree 46
 visitClass(on CollaborationFormatter 53,
 392
 visitClass(on DiaFormatter 48, 387
 visitClass(on DocFormatter 50, 389
 visitClass(on Formatter 44, 50, 61, 389, 398
 visitClass(on HTMLFormatter 57, 395
 visitClass(on InheritanceFormatter ... 52, 391
 visitClass(on Part 66, 348
 visitClass(on SingleInheritanceFormatter
 53, 392
 visitClass(on Stripper 118, 341

- visitClass(on TableOfContents 56, 393
- visitClass(on Unduplicator 120, 343
- visitClass(on Visitor 23
- visitComment(on DocFormatter 50, 389
- visitComment(on Formatter 44, 49, 389
- visitComment(on Visitor 24
- visitConst(on ASCIIFormatter 42, 346
- visitConst(on ASTTranslator 123, 328
- visitConst(on Formatter ... 44, 49, 61, 389, 398
- visitConst(on HTMLFormatter 57, 394
- visitConst(on Part 66, 349
- visitConst(on TableOfContents 56, 394
- visitConst(on Unduplicator 120, 343
- visitConst(on Visitor 23
- visitDeclaration(on AccessRestrictor ... 109, 333
- visitDeclaration(on ASCIIFormatter ... 41, 346
- visitDeclaration(on CommentProcessor ... 110, 333
- visitDeclaration(on DiaFormatter 48, 387
- visitDeclaration(on Dummies 112, 336
- visitDeclaration(on EmptyNS 115, 338
- visitDeclaration(on Grouper 113, 337
- visitDeclaration(on ListFiller 143
- visitDeclaration(on MenuMaker 60, 397
- visitDeclaration(on NameMapper 117, 340
- visitDeclaration(on Part 65, 348
- visitDeclaration(on Previous 113, 336
- visitDeclaration(on Stripper 118, 341
- visitDeclaration(on TableOfContents 62
- visitDeclaration(on Visitor 23
- visitDeclarator(on Formatter .. 44, 49, 61, 389, 398
- visitDeclarator(on HTMLFormatter 57, 394
- visitDeclarator(on TableOfContents ... 56, 393
- visitDeclared(on ASCIIFormatter 41, 345
- visitDeclared(on CollaborationFormatter .. 53, 392
- visitDeclared(on DiaFormatter 47, 387
- visitDeclared(on Formatter 43, 49, 61, 389, 398
- visitDeclared(on HTMLFormatter 57, 394
- visitDeclared(on InheritanceFormatter ... 52, 391
- visitDeclared(on Part 66, 349
- visitDeclared(on SingleInheritanceFormatter 53, 392
- visitDeclared(on ToDecl 94, 371
- visitDeclared(on Unduplicator 119, 342
- visitDeclared(on Visitor 37, 155
- visitDeclaredType(on TypeTranslator 123, 327
- visitDependent(on ASCIIFormatter 41, 345
- visitDependent(on InheritanceFormatter ... 52, 391
- visitDependent(on Part 66, 349
- visitDependent(on Visitor 37, 155
- visitDict(on Dumper 55, 385
- visitEnum(on ASCIIFormatter 42, 346
- visitEnum(on ASTTranslator 124, 328
- visitEnum(on Dummies 112, 336
- visitEnum(on EmptyNS 115, 338
- visitEnum(on Formatter 44, 50, 62, 389, 399
- visitEnum(on Grouper 114, 337
- visitEnum(on HTMLFormatter 58, 395
- visitEnum(on ListFiller 143
- visitEnum(on Part 66, 349
- visitEnum(on Previous 113, 336
- visitEnum(on Stripper 118, 341
- visitEnum(on TableOfContents 56, 394
- visitEnum(on Visitor 23
- visitEnumerator(on ASCIIFormatter ... 42, 346
- visitEnumerator(on ASTTranslator ... 124, 328
- visitEnumerator(on Dummies 112, 336
- visitEnumerator(on Formatter .. 44, 50, 61, 389, 399
- visitEnumerator(on Grouper 114, 337
- visitEnumerator(on HTMLFormatter 58, 395
- visitEnumerator(on Previous 113, 337
- visitEnumerator(on Stripper 118, 341
- visitEnumerator(on TableOfContents ... 56, 393
- visitEnumerator(on Visitor 23
- visitException(on ASTTranslator 124, 328
- visitFloat(on Dumper 55, 385
- visitFormat(on ActionColorizer 131
- visitFormat(on ActionIcon 132
- visitFormat(on ActionVisitor 10
- visitFormat(on ExecutorCreator 25
- visitForward(on ASTTranslator 123, 328
- visitForward(on ListFiller 143
- visitForward(on Part 65, 348
- visitForward(on TableOfContents 63
- visitForward(on Visitor 23
- visitFunction(on ASCIIFormatter 42, 346
- visitFunction(on Formatter 44, 50, 61, 389, 399
- visitFunction(on HTMLFormatter 58, 395
- visitFunction(on Part 66, 349
- visitFunction(on Stripper 118, 341
- visitFunction(on TableOfContents 56, 394
- visitFunction(on Unduplicator 120, 343
- visitFunction(on Visitor 23
- visitFunctionType(on ASCIIFormatter .. 41, 346
- visitFunctionType(on CollaborationFormatter 54, 392
- visitFunctionType(on DiaFormatter ... 48, 387
- visitFunctionType(on Formatter 44, 49, 61, 389, 398
- visitFunctionType(on HTMLFormatter ... 57, 394
- visitFunctionType(on Part 67, 349
- visitFunctionType(on ToDecl 94, 372
- visitFunctionType(on Unduplicator ... 120, 342
- visitFunctionType(on Visitor 37, 155
- visitGroup(on EmptyNS 115, 338

- visitGroup(on HTMLFormatter 57, 395
- visitGroup(on ListFiller 143
- visitGroup(on NameMapper 117, 340
- visitGroup(on Part 65, 348
- visitGroup(on TableOfContents 56, 393
- visitGroup(on Visitor 23
- visitInheritance(on ASCIIFormatter ... 42, 346
- visitInheritance(on DocFormatter 50, 389
- visitInheritance(on Formatter 44, 50, 61, 389, 398
- visitInheritance(on HTMLFormatter ... 57, 395
- visitInheritance(on InheritanceFormatter 52, 391
- visitInheritance(on SingleInheritanceFormatter 53, 392
- visitInheritance(on Unduplicator 120, 343
- visitInheritance(on Visitor 24
- visitInstance(on Dumper 55, 385
- visitInt(on Dumper 54, 385
- visitInterface(on ASTTranslator 123, 328
- visitLinker(on ActionColorizer 131
- visitLinker(on ActionIcon 132
- visitLinker(on ActionVisitor 10
- visitLinker(on ExecutorCreator 25
- visitList(on Dumper 55, 385
- visitLong(on Dumper 54, 385
- visitMacro(on ASCIIFormatter 41, 346
- visitMacro(on Visitor 23
- visitMember(on ASTTranslator 124, 328
- visitMetaModule(on ASCIIFormatter ... 42, 346
- visitMetaModule(on Part 66, 348
- visitMetaModule(on Stripper 118, 341
- visitMetaModule(on Unduplicator 120, 342
- visitMetaModule(on Visitor 23
- visitModifier(on ASCIIFormatter 41, 345
- visitModifier(on CollaborationFormatter .. 54, 392
- visitModifier(on DiaFormatter 47, 387
- visitModifier(on Formatter 43, 49, 61, 389, 398
- visitModifier(on HTMLFormatter 57, 394
- visitModifier(on InheritanceFormatter ... 52, 391
- visitModifier(on Part 66, 349
- visitModifier(on ToDecl 94, 371
- visitModifier(on Unduplicator 119, 342
- visitModifier(on Visitor 37, 155
- visitModule(on ASCIIFormatter 42, 346
- visitModule(on ASTTranslator 123, 328
- visitModule(on DiaFormatter 48, 387
- visitModule(on DocFormatter 50, 389
- visitModule(on EmptyNS 115, 338
- visitModule(on Formatter ... 44, 50, 61, 389, 398
- visitModule(on HTMLFormatter 57, 395
- visitModule(on Part 66, 348
- visitModule(on TableOfContents 56, 393
- visitModule(on Unduplicator 120, 342
- visitModule(on Visitor 23
- visitNamed(on Unduplicator 120, 343
- visitNone(on Dumper 54, 385
- visitOperation(on ASCIIFormatter 42, 346
- visitOperation(on ASTTranslator 124, 328
- visitOperation(on Formatter ... 44, 50, 61, 389, 399
- visitOperation(on HTMLFormatter 58, 395
- visitOperation(on InheritanceFormatter ... 52, 391
- visitOperation(on Part 66, 349
- visitOperation(on Stripper 118, 341
- visitOperation(on TableOfContents ... 56, 394
- visitOperation(on Visitor 24
- visitParameter(on ASCIIFormatter 42, 346
- visitParameter(on ASTTranslator 124, 328
- visitParameter(on Formatter ... 44, 50, 61, 389, 399
- visitParameter(on HTMLFormatter 58, 395
- visitParameter(on TableOfContents ... 56, 394
- visitParameter(on Unduplicator 120, 343
- visitParameter(on Visitor 24
- visitParametrized(on ASCIIFormatter .. 41, 345
- visitParametrized(on CollaborationFormatter 54, 392
- visitParametrized(on DiaFormatter 47, 387
- visitParametrized(on Formatter 43, 49, 61, 389, 398
- visitParametrized(on HTMLFormatter ... 57, 394
- visitParametrized(on InheritanceFormatter 52, 391
- visitParametrized(on Part 66, 349
- visitParametrized(on ToDecl 94, 372
- visitParametrized(on Unduplicator ... 119, 342
- visitParametrized(on Visitor 37, 155
- visitParser(on ActionColorizer 131
- visitParser(on ActionIcon 132
- visitParser(on ActionVisitor 9
- visitParser(on ExecutorCreator 25
- visitScope(on AccessRestrictor 109, 333
- visitScope(on ClassTree 46
- visitScope(on Dummies 112, 336
- visitScope(on Grouper 114, 337
- visitScope(on Part 66, 348
- visitScope(on Previous 113, 336
- visitScope(on Stripper 118, 341
- visitScope(on Visitor 23
- visitSequenceType(on TypeTranslator 123, 327
- visitSource(on ActionColorizer 131
- visitSource(on ActionIcon 132
- visitSource(on ActionVisitor 9
- visitSource(on ExecutorCreator 25
- visitSourceFile(on Unduplicator 120, 342
- visitString(on Dumper 55, 385
- visitStringType(on TypeTranslator ... 123, 327
- visitStruct(on ASTTranslator 124, 328

- visitTemplate(on ASCIIFormatter..... 41, 346
 - visitTemplate(on CollaborationFormatter.. 54, 392
 - visitTemplate(on InheritanceFormatter.... 52, 391
 - visitTemplate(on Part 67, 349
 - visitTemplate(on ToDecl 94, 371
 - visitTemplate(on Unduplicator..... 119, 342
 - visitTemplate(on Visitor 37, 155
 - visitTuple(on Dumper 55, 385
 - visitType(on Dumper 54, 385
 - visitTypedef(on ASCIIFormatter..... 42, 346
 - visitTypedef(on ASTTranslator..... 124, 328
 - visitTypedef(on Formatter.. 44, 49, 61, 389, 398
 - visitTypedef(on HTMLFormatter..... 57, 394
 - visitTypedef(on Part 66, 348
 - visitTypedef(on TableOfContents..... 56, 393
 - visitTypedef(on Unduplicator..... 120, 343
 - visitTypedef(on Visitor..... 23
 - visitUnion(on ASTTranslator..... 124, 328
 - visitUnionCase(on ASTTranslator 124, 328
 - visitUnknown(on ASCIIFormatter..... 41, 345
 - visitUnknown(on CollaborationFormatter... 53, 392
 - visitUnknown(on DiaFormatter..... 47, 387
 - visitUnknown(on Formatter.. 43, 49, 61, 389, 398
 - visitUnknown(on HTMLFormatter..... 57, 394
 - visitUnknown(on InheritanceFormatter.... 52, 391
 - visitUnknown(on Mapper 117, 340
 - visitUnknown(on Part 66, 349
 - visitUnknown(on ToDecl 94, 371
 - visitUnknown(on Unduplicator..... 119, 342
 - visitUnknown(on Visitor 37, 155
 - visitVariable(on ASCIIFormatter..... 42, 346
 - visitVariable(on CollaborationFormatter.. 53, 392
 - visitVariable(on Formatter..... 44, 49, 61, 389, 398
 - visitVariable(on HTMLFormatter..... 57, 394
 - visitVariable(on InheritanceFormatter.... 53, 391
 - visitVariable(on Part 66, 349
 - visitVariable(on TableOfContents 56, 394
 - visitVariable(on Unduplicator..... 120, 343
 - visitVariable(on Visitor..... 23
 - visitWStringType(on TypeTranslator .. 123, 327
 - voidon function 162, 163, 164, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 188, 189, 190, 191, 192, 193, 194, 195, 196, 198, 199, 200, 201, 202, 204, 207, 208, 209, 216, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 237, 242, 243, 244, 245, 247, 248, 249, 250, 253, 254, 255, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 269, 270, 271, 272, 273, 274, 275, 281, 282, 283, 287, 288, 289, 290, 296, 299, 300, 301, 302, 307, 308, 309, 310, 311, 312, 313, 314, 315, 317, 318, 323, 324, 325, 326
 - vttype(on Variable 19
- ## W
- wait(..... 232
 - width(on Icon 145
 - windowActivated(on BrowserWindow..... 142
 - windowActivated(on CanvasWindow 134
 - windowActivated(on ProjectWindow..... 147
 - write(..... 240
 - write(on ASCIIFormatter 41, 345
 - write(on CollaborationFormatter..... 53, 392
 - write(on DiaFormatter 46, 386
 - write(on Dumper..... 54, 385
 - write(on Formatter 43, 48, 60, 388, 398
 - write(on HTMLFormatter 57, 394
 - write(on InheritanceFormatter..... 51, 390
 - write(on MenuMaker..... 60, 397
 - write(on Page 100, 377
 - write(on Part 64, 347
 - write(on PyWriter 39, 157
 - write(on TableOfContents 55, 393
 - write(on TemplatePageFormat..... 100, 377
 - write_attr(on PyWriter 40, 158
 - write_CacherAction(on ProjectWriter 31
 - write_compiler_infos(..... 122, 160
 - write_end(on DOSummary 79, 360
 - write_end(on Part 67, 350
 - write_entity(on Formatter..... 43, 49, 388
 - write_ExcludeSourceRule(on ProjectWriter 31
 - write_FormatAction(on ProjectWriter 31
 - write_GlobSourceRule(on ProjectWriter..... 31
 - write_indent(..... 240
 - write_item(on PyWriter 39, 157
 - write_lineno(..... 240
 - write_LinkerAction(on ProjectWriter 31
 - write_list(on PyWriter 40, 158
 - write_long_list(on PyWriter 40, 158
 - write_ParserAction(on ProjectWriter..... 31
 - write_Project(on ProjectWriter 31
 - write_ProjectActions(on ProjectWriter..... 31
 - write_PyWriterStruct(on PyWriter 40, 158
 - write_SimpleSourceRule(on ProjectWriter .. 31
 - write_SourceAction(on ProjectWriter 31
 - write_start(on DOSummary 79, 360
 - write_start(on Part 67, 349
 - write_top(on PyWriter 39, 157
 - writeComments(on ASCIIFormatter 41, 346
 - writeEdge(on InheritanceFormatter 52, 391
 - writeLeaf(on JSTree 95, 373
 - writeLeaf(on TreeFormatter..... 107, 383
 - writeLeaf(on TreeFormatterJS..... 106, 382
 - writeln(on Dumper 54, 385

| | | | |
|--|----------|--|---------|
| <code>writeNode</code> (on InheritanceFormatter) | 52, 391 | <code>writeSectionItem</code> (on Summary) | 68, 350 |
| <code>writeNodeEnd</code> (on JSTree) | 96, 373 | <code>writeSectionItem_Foo</code> (on DODetail) | 79, 360 |
| <code>writeNodeEnd</code> (on TreeFormatter) | 108, 384 | <code>writeSectionItem_Foo</code> (on DOSummary) | 79, 360 |
| <code>writeNodeEnd</code> (on TreeFormatterJS) | 107, 383 | <code>writeSectionStart</code> (on Detail) | 68, 351 |
| <code>writeNodeStart</code> (on JSTree) | 96, 373 | <code>writeSectionStart</code> (on DOSummary) | 79, 360 |
| <code>writeNodeStart</code> (on TreeFormatter) | 108, 383 | <code>writeSectionStart</code> (on Inheritance) | 69, 351 |
| <code>writeNodeStart</code> (on TreeFormatterJS) | 107, 383 | <code>writeSectionStart</code> (on Part) | 67, 349 |
| <code>writeSectionEnd</code> (on DODetail) | 79, 360 | <code>writeSectionStart</code> (on Summary) | 68, 350 |
| <code>writeSectionEnd</code> (on DOSummary) | 79, 360 | | |
| <code>writeSectionEnd</code> (on Inheritance) | 69, 352 | X | |
| <code>writeSectionEnd</code> (on Part) | 67, 349 | <code>x</code> (on Action) | 9 |
| <code>writeSectionEnd</code> (on Summary) | 68, 350 | | |
| <code>writeSectionItem</code> (on Detail) | 69, 351 | Y | |
| <code>writeSectionItem</code> (on Heading) | 67, 350 | <code>y</code> (on Action) | 9 |
| <code>writeSectionItem</code> (on Inheritance) | 69, 351 | | |
| <code>writeSectionItem</code> (on Part) | 67, 350 | | |

Table of Contents

| | |
|--|------------|
| Synopsis Reference Manual Copyright Notice . . . | 1 |
| Acknowledgments | 2 |
| 1 Overview | 3 |
| 2 The Core Module | 4 |
| 2.1 The AST Module | 5 |
| 2.2 The Type Module | 151 |
| 2.3 The Util Module | 156 |
| 3 The Parser Module | 159 |
| 3.1 The C++ Module | 160 |
| 3.2 The IDL Module | 327 |
| 3.3 The Python Module | 329 |
| 4 The Linker Module | 333 |
| 5 The Formatter Module | 344 |
| 5.1 The ASCII Module | 345 |
| 5.2 The HTML Module | 347 |
| 5.3 The DUMP Module | 385 |
| 5.4 The Dia Module | 386 |
| 5.5 The DocBook Module | 388 |
| 5.6 The Dot Module | 390 |
| 5.7 The HTML_Simple Module | 393 |
| 5.8 The TexInfo Module | 396 |
| Type Index | 400 |
| Function Index | 406 |