languageParser

languageParser

```
new languageParser(languageInformation) → {languageParser}
```

Source: util/languageParser.js, line 39

The language parser class allows for quick indexing and predictive searching of user defined functions.

Parameters:

Name	Type	Description
languageInformation	Object	This is the key to a language, all aspects of the language such as functions, primitive types, and commenting information are defined within this object.

Returns:

Returns a new language parser ready to parse the language defined by

Type languageParser

Methods

```
addFunction(l_function)
```

Source: util/languageParser.js, line 241

This function is used when reading the language description. This function generates a hashmap of functions defined within this languages domain. These functions are looked up when generating IntelliSence

Parameters:

Name	Type	Description
l_function	l_function	The language specific function to add to this languages registered functions.

```
cursorInScope(cursor, scope) → {boolean}
```

Source: util/languageParser.js, line 175

This function takes a cursor and a scope and returns true if the cursor is inside of the scope.

Parameters:

Name	Type	Description
`cursor`		cursor object
`scope`		scope to check cursor against.

Returns:

If cursor 'cursor' is inside of scope 'scope'

Type `boolean`

cursorToScope(cursor) → {*}

Source: util/languageParser.js, line 158

This function returns which scope the cursor is currently in. A cursor object contains a line number and a character.

Parameters:

Name	Type	Description
`cursor`	`Cursor`	An object representing a cursors position inside of this file.

Returns:

Returs the scope that the cursor is inside of.

Type `*`

getLastToken(tokenArrayg) → {*}

Source: util/languageParser.js, line 326

Returns the last token of a line

Parameters:

Name	Type	Description
`tokenArrayg`		

Returns:

Type `*`

getSubScope(scope) → {Array.<Scope>}

Source: util/languageParser.js, line 209

This function returns all child scopes of a desired scope.

Parameters:

Name	Type	Description
scope	Scope	The scope to get the children of.

Returns:

Returns the array of child scopes parented to 'scope'.

Type Array.<Scope>

getSuggestion(string, cursor) → {Array.<l_function>}

Source: util/languageParser.js, line 257

This function returs an array of l_functions which the user could be typing. This function performs a fuzzy search through the list of defined functions and generates a subset of functions that are similar to 'string'

Parameters:

Name	Type	Description
string	String	String to search
cursor	Cursor	position in the file

Returns:

Returns an array of l_functions which the user could be trying to type.

Type Array.<l_function>

loadFileSpecificData(fileData)

Source: util/languageParser.js, line 69

This function takes in data from a file and generate a tree structure for this file representing the scopes of this file. This scoping information is used to determine local variables.

Parameters:

Name	Type	Description

Name	Type	Description
fileData	String	This is a string of all lines of a file, with each line delimited with the \n character.

offsetScopes(delta, cursor)

Source: util/languageParser.js, line 222

When code is added to or removed from a document at a position, scopes need to be offset by that ammount. Example, if there is a scope which starts on line 42, but lines 5-10 are deleted, then the scope starting on 42 will now start on 36. This function updates all scopes so they are still in the correct place.

Parameters:

Name	Type	Description
delta	Integer	Number of lines to offset scopes by, either positive for new lines, or negative for deletions.
cursor	Cursor	The position of the cursor, so we know where these lines were inserted or deleted relative to.

tokeniseString(string) → {Array.<String>}

Source: util/languageParser.js, line 297

This function takes in a string and turns it into an array of string tokens.

Parameters:

Name	Type	Description
string	String	String to tokenise

Returns:

- Array of string tokens

Type Array.<String>