# rootJS

Generated by Doxygen 1.8.6

Sat Feb 13 2016 14:01:06

# Contents

# Chapter 1

# Hierarchical Index

## 1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

# Chapter 2

# Class Index

## 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 rootJS::AsyncRunner Class Reference

```
#include <AsyncRunner.h>
```

### Public Member Functions

- AsyncRunner (AsyncFunction func, void ∗param, v8::Persistent< v8::Function, v8::CopyablePersistentTraits< v8::Function >> callback)
- void run ()
- void setResult (std::vector< ObjectProxy ∗ > result)

### Static Public Member Functions

- static void uvRunner (uv_work_t ∗req)
- static void uvCallback (uv_work_t ∗req, int status)

### 3.1.1 Detailed Description

The AsyncRunner provides methods to enable asyncronous function execution.

Definition at line 16 of file AsyncRunner.h.

### 3.1.2 Constructor & Destructor Documentation

#### 3.1.2.1 rootJS::AsyncRunner::AsyncRunner ( AsyncFunction *func,* void ∗ *param,* v8::Persistent< v8::Function, v8::CopyablePersistentTraits< v8::Function >> *callback* )

Creates a new AsyncRunner.

**Parameters**

| | |
|---|---|
| *func* | The function to be called asynchronously. |
| ∗*param* | The parameters for the function. |
| *callback* | The callback to be executed in the node thread after the asynchronous function is finished. |

Definition at line 4 of file AsyncRunner.cc.

### 3.1.3 Member Function Documentation

#### 3.1.3.1 void rootJS::AsyncRunner::run ( )

Runs the function given with the constructor asynchronously. After it has finished, the callback function is executed in the original node thread.

Definition at line 9 of file AsyncRunner.cc.

#### 3.1.3.2 void rootJS::AsyncRunner::setResult ( std::vector< **ObjectProxy** * > *result* ) [inline]

Allows the asynchronous function to set its result.

**Parameters**

| | |
|---|---|
| *result* | The result of the asynchronous function. |

Definition at line 46 of file AsyncRunner.h.

#### 3.1.3.3 void rootJS::AsyncRunner::uvCallback ( uv_work_t * *req*, int *status* ) [static]

Executes the callback function.

**Parameters**

| | |
|---|---|
| **req* | The libuv request, used to get context. |
| *status* | If the callback is canceled with uv_cancel() this is UV_ECANCELED. |

Definition at line 20 of file AsyncRunner.cc.

#### 3.1.3.4 void rootJS::AsyncRunner::uvRunner ( uv_work_t * *req* ) [static]

This needs to be implemented for libuv.

**Parameters**

| | |
|---|---|
| **req* | A libuv request. |

Definition at line 15 of file AsyncRunner.cc.

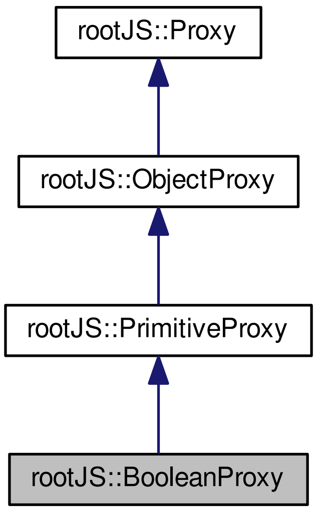
The documentation for this class was generated from the following files:

- src/AsyncRunner.h
- src/AsyncRunner.cc

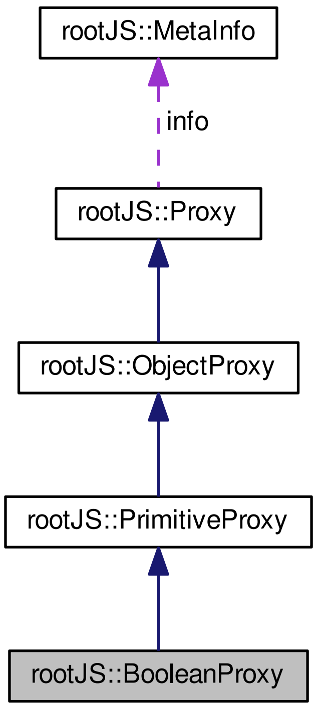
## 3.2 rootJS::BooleanProxy Class Reference

```
#include <BooleanProxy.h>
```

Inheritance diagram for rootJS::BooleanProxy:



Collaboration diagram for rootJS::BooleanProxy:



## Public Member Functions

- BooleanProxy (MetaInfo &info, TClass ∗scope)
- virtual v8::Local< v8::Value > get ()
- virtual void setValue (v8::Local< v8::Value > value)

## Static Public Member Functions

- static bool isBoolean (std::string type)
- static ObjectProxy ∗ boolConstruct (MetaInfo &info, TClass ∗scope)

## Additional Inherited Members

### 3.2.1 Detailed Description

Maps a C++/ROOT boolean to JavaScript boolean.

Definition at line 14 of file BooleanProxy.h.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 rootJS::BooleanProxy::BooleanProxy ( **MetaInfo** & *info,* **TClass** ∗ *scope* )

Create a new BooleanProxy.

**Parameters**

| | |
|---|---|
| *info* | the type of the excapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 22 of file BooleanProxy.cc.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 **ObjectProxy** ∗ rootJS::BooleanProxy::boolConstruct ( **MetaInfo** & *info,* **TClass** ∗ *scope* ) `[static]`

Creates a BooleanProxy, can be derefered to be added to a map

**Parameters**

| | |
|---|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 31 of file BooleanProxy.cc.

#### 3.2.3.2 v8::Local< v8::Value > rootJS::BooleanProxy::get ( ) `[virtual]`

Returns a v8 Boolean depending on MetaInfo

**Returns**

boolean depending on MetaInfo

Reimplemented from rootJS::ObjectProxy.

Definition at line 36 of file BooleanProxy.cc.

#### 3.2.3.3 bool rootJS::BooleanProxy::isBoolean ( std::string *type* ) [static]

Check if the type is a boolean type.

**Parameters**

| | |
|---|---|
| *type* | the type to be checked |

**Returns**

if the type is a boolean type

Definition at line 7 of file BooleanProxy.cc.

#### 3.2.3.4 void rootJS::BooleanProxy::setValue ( v8::Local< v8::Value > *value* ) [virtual]

Sets the boolen in memory, using the data passed via JavaScript

**Parameters**

| | |
|---|---|
| *value* | The value to be set |

Reimplemented from rootJS::ObjectProxy.

Definition at line 42 of file BooleanProxy.cc.

The documentation for this class was generated from the following files:

- src/BooleanProxy.h
- src/BooleanProxy.cc

## 3.3 rootJS::CallbackHandler Class Reference

### Static Public Member Functions

- static void registerGlobalObject (const std::string &name, ObjectProxy ∗proxy)
- static void globalGetterCallback (v8::Local< v8::String > property, const v8::PropertyCallbackInfo< v8::Value > &info)
- static void globalSetterCallback (v8::Local< v8::String > property, v8::Local< v8::Value > value, const v8::-PropertyCallbackInfo< void > &info)
- static void globalFunctionCallback (const v8::FunctionCallbackInfo< v8::Value > &info)
- static void registerStaticObject (const std::string &name, TClass ∗scope, ObjectProxy ∗proxy)
- static void staticGetterCallback (v8::Local< v8::String > property, const v8::PropertyCallbackInfo< v8::Value > &info)
- static void staticSetterCallback (v8::Local< v8::String > property, v8::Local< v8::Value > value, const v8::-PropertyCallbackInfo< void > &info)
- static void staticFunctionCallback (const v8::FunctionCallbackInfo< v8::Value > &info)
- static void ctorCallback (const v8::FunctionCallbackInfo< v8::Value > &info)
- static void memberGetterCallback (v8::Local< v8::String > property, const v8::PropertyCallbackInfo< v8::-Value > &info)
- static void memberSetterCallback (v8::Local< v8::String > property, v8::Local< v8::Value > value, const v8::PropertyCallbackInfo< void > &info)
- static void memberFunctionCallback (const v8::FunctionCallbackInfo< v8::Value > &info)
- static v8::Local< v8::Value > **createFunctionCallbackData** (std::string functionName, TClass ∗scope)
- static v8::Local< v8::Value > **createFunctionCallbackData** (TClass ∗scope)

### Static Public Attributes

- static const std::string **CALLBACK_DATA_DELIMITER** = "#"

### 3.3.1 Detailed Description

Definition at line 19 of file CallbackHandler.h.

### 3.3.2 Member Function Documentation

#### 3.3.2.1 void rootJS::CallbackHandler::ctorCallback ( const v8::FunctionCallbackInfo< v8::Value > & *info* ) [static]

This callback method may be invoked whenever a JavaScript prototype function of an encapsulated ROOT class was called.

Based on the supplied arguments the suitable C++ constructor will be called. Then the newly created instance will be wrapped into an ObjectProxy. As result the ObjectProxy's corresponding JavaScript object will be returned to the Node.js application.

In order to enable non blocking object creation one can supply a JavaScript callback function as last argument of the prototype function. After the asynchronous object creation is finished the supplied callback will be invoked for returning the generated JavasSript proxy.

**Parameters**

| | |
|---|---|
| *info* | the argument information given to this function call callback |

Definition at line 175 of file CallbackHandler.cc.

#### 3.3.2.2 void rootJS::CallbackHandler::globalFunctionCallback ( const v8::FunctionCallbackInfo< v8::Value > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *info* | |

Definition at line 44 of file CallbackHandler.cc.

#### 3.3.2.3 void rootJS::CallbackHandler::globalGetterCallback ( v8::Local< v8::String > *property*, const v8::PropertyCallbackInfo< v8::Value > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *info* | |

Definition at line 26 of file CallbackHandler.cc.

#### 3.3.2.4 void rootJS::CallbackHandler::globalSetterCallback ( v8::Local< v8::String > *property*, v8::Local< v8::Value > *value*, const v8::PropertyCallbackInfo< void > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *value* | |
| *info* | |

Definition at line 35 of file CallbackHandler.cc.

#### 3.3.2.5 void rootJS::CallbackHandler::memberFunctionCallback ( const v8::FunctionCallbackInfo< v8::Value > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *info* | |

Definition at line 250 of file CallbackHandler.cc.

#### 3.3.2.6 void rootJS::CallbackHandler::memberGetterCallback ( v8::Local< v8::String > *property,* const v8::PropertyCallbackInfo< v8::Value > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *info* | |

Definition at line 244 of file CallbackHandler.cc.

#### 3.3.2.7 void rootJS::CallbackHandler::memberSetterCallback ( v8::Local< v8::String > *property,* v8::Local< v8::Value > *value,* const v8::PropertyCallbackInfo< void > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *value* | |
| *info* | |

Definition at line 247 of file CallbackHandler.cc.

#### 3.3.2.8 void rootJS::CallbackHandler::registerGlobalObject ( const std::string & *name,* ObjectProxy * *proxy* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *name* | |
| *proxy* | |

Definition at line 21 of file CallbackHandler.cc.

#### 3.3.2.9 void rootJS::CallbackHandler::registerStaticObject ( const std::string & *name,* TClass * *scope,* ObjectProxy * *proxy* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *name* | |
| *proxy* | |

Definition at line 108 of file CallbackHandler.cc.

#### 3.3.2.10 void rootJS::CallbackHandler::staticFunctionCallback ( const v8::FunctionCallbackInfo< v8::Value > & *info* ) `[static]`

TODO: fill in description

**Parameters**

| | |
|---|---|
| *info* | |

Definition at line 126 of file CallbackHandler.cc.

#### 3.3.2.11 void rootJS::CallbackHandler::staticGetterCallback ( v8::Local< v8::String > *property*, const v8::PropertyCallbackInfo< v8::Value > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *info* | |

Definition at line 120 of file CallbackHandler.cc.

#### 3.3.2.12 void rootJS::CallbackHandler::staticSetterCallback ( v8::Local< v8::String > *property*, v8::Local< v8::Value > *value*, const v8::PropertyCallbackInfo< void > & *info* ) [static]

TODO: fill in description

**Parameters**

| | |
|---|---|
| *property* | |
| *value* | |
| *info* | |

Definition at line 123 of file CallbackHandler.cc.

The documentation for this class was generated from the following files:

- src/CallbackHandler.h
- src/CallbackHandler.cc

## 3.4 rootJS::ClassExposer Class Reference

### Static Public Member Functions

- static void **expose** (TClass $*$, v8::Local< v8::Object >) throw (std::invalid_argument)
- static std::vector< std::string > **splitClassName** (std::string name, std::vector< std::string > &vec)

### 3.4.1 Detailed Description

Definition at line 9 of file ClassExposer.h.

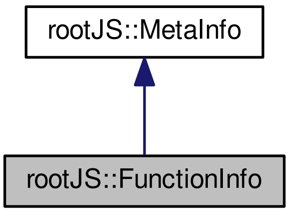
The documentation for this class was generated from the following files:

- src/ClassExposer.h
- src/ClassExposer.cc

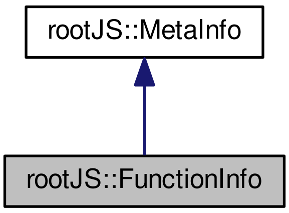
## 3.5 rootJS::FunctionInfo Class Reference

```
#include <FunctionInfo.h>
```

Inheritance diagram for rootJS::FunctionInfo:



Collaboration diagram for rootJS::FunctionInfo:



## Public Member Functions

- **FunctionInfo** (const TFunction &type, void ∗baseAddress)
- **FunctionInfo** (const TFunction &type, void ∗baseAddress, bool isGlobal)
- virtual Long_t getOffset ()
- virtual bool isGlobal ()
- virtual bool isConst ()
- virtual bool isStatic ()
- virtual const char ∗ getTypeName ()
- virtual const char ∗ getName ()
- virtual FunctionInfo ∗ clone ()

## Additional Inherited Members

### 3.5.1 Detailed Description

This class contains the info for a TFunction

Definition at line 14 of file FunctionInfo.h.

## 3.5.2 Member Function Documentation

### 3.5.2.1 FunctionInfo * rootJS::FunctionInfo::clone ( ) [virtual]

Makes a clone of the MetaInfo instance.

**Returns**

Pointer to the cloned MetaInfo instance

Implements rootJS::MetaInfo.

Definition at line 45 of file FunctionInfo.cc.

### 3.5.2.2 const char * rootJS::FunctionInfo::getName ( ) [virtual]

Returns the name of the TObject.

**Returns**

name of the TObject

Reimplemented from rootJS::MetaInfo.

Definition at line 40 of file FunctionInfo.cc.

### 3.5.2.3 Long_t rootJS::FunctionInfo::getOffset ( ) [virtual]

Get the offset. This calls up the TDataMember::GetOffset() function.

**Returns**

The offset

Reimplemented from rootJS::MetaInfo.

Definition at line 15 of file FunctionInfo.cc.

### 3.5.2.4 const char * rootJS::FunctionInfo::getTypeName ( ) [virtual]

Returns the typename of the TObject.

**Returns**

Typename of the TObject

Implements rootJS::MetaInfo.

Definition at line 35 of file FunctionInfo.cc.

### 3.5.2.5 bool rootJS::FunctionInfo::isConst ( ) [virtual]

Checks if the TObject is a constant.

**Returns**

If the TObject is a constant

Implements rootJS::MetaInfo.

Definition at line 25 of file FunctionInfo.cc.

#### 3.5.2.6 bool rootJS::FunctionInfo::isGlobal ( ) [virtual]

Checks if the TObject is global.

**Returns**

If the TObject is global

Reimplemented from rootJS::MetaInfo.

Definition at line 20 of file FunctionInfo.cc.

#### 3.5.2.7 bool rootJS::FunctionInfo::isStatic ( ) [virtual]

Checks if the TObject is static.

**Returns**

If the TObject is static

Implements rootJS::MetaInfo.

Definition at line 30 of file FunctionInfo.cc.

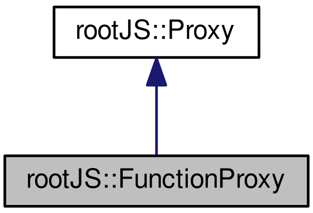
The documentation for this class was generated from the following files:

- src/FunctionInfo.h
- src/FunctionInfo.cc

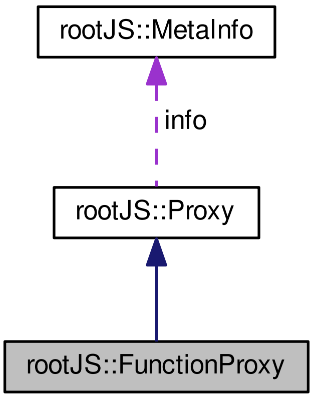
## 3.6 rootJS::FunctionProxy Class Reference

```
#include <FunctionProxy.h>
```

Inheritance diagram for rootJS::FunctionProxy:



Collaboration diagram for rootJS::FunctionProxy:



### Public Member Functions

- FunctionProxy (void ∗address, FunctionInfo &info, TFunction ∗function, TClass ∗scope)
- FunctionProxy ∗ clone ()
- void **prepareCall** (const v8::Local< v8::Array > &args)
- ObjectProxy ∗ call (bool isConstructorCall=false)
- void setSelfAddress (void ∗addr)
- virtual bool isConst ()
- virtual bool isGlobal ()
- virtual bool isStatic ()
- virtual bool isTemplate ()

### Static Public Member Functions

- static std::vector< TFunction ∗ > getMethodsFromName (TClassRef scope, std::string name)
- static CallFunc_t ∗ getCallFunc (const TClassRef &klass, TFunction ∗method)

### Additional Inherited Members

## 3.6.1 Detailed Description

Represents a ROOT callable and provides functionality to invoke those callables. Also acts as a static cache for already created FunctionProxy objects.

Definition at line 30 of file FunctionProxy.h.

## 3.6.2 Constructor & Destructor Documentation

#### 3.6.2.1 rootJS::FunctionProxy::FunctionProxy ( void ∗ *address,* FunctionInfo & *info,* TFunction ∗ *function,* TClass ∗ *scope* )

Create a new FunctionProxy.

**Parameters**

| | |
|---|---|
| *address* | memory address of the proxied function |
| *function* | the function's reflection object |
| *scope* | the class that the function belongs to |

Definition at line 61 of file FunctionProxy.cc.

## 3.6.3 Member Function Documentation

### 3.6.3.1 ObjectProxy ∗ rootJS::FunctionProxy::call ( bool *isConstructorCall* = `false` )

Invokes the proxied function.

**Parameters**

| | |
|---|---|
| *args* | the arguments for the function call. |

**Returns**

the function's return value encasulated in an ObjectProxy

Definition at line 176 of file FunctionProxy.cc.

### 3.6.3.2 FunctionProxy ∗ rootJS::FunctionProxy::clone ( )

Makes a clone of the current FunctionProxy

**Returns**

a pointer to the clone

Definition at line 67 of file FunctionProxy.cc.

### 3.6.3.3 CallFunc_t ∗ rootJS::FunctionProxy::getCallFunc ( const TClassRef & *klass*, TFunction ∗ *method* ) `[static]`

Get a pointer to a CallFunc object, which encapsulates the ROOT function in memory.

**Parameters**

| | |
|---|---|
| *method* | the callable whose CallFunc object shall be returned |

**Returns**

a pointer to the CallFunc object provided by cling

Definition at line 76 of file FunctionProxy.cc.

### 3.6.3.4 std::vector< TFunction ∗ > rootJS::FunctionProxy::getMethodsFromName ( TClassRef *scope*, std::string *name* ) `[static]`

Get all methods of the specified class with the specified name.

**Parameters**

| | |
|---|---|
| *scope* | reference to the class which is checked for methods with the specified name |
| *name* | name of the overloaded methods which shall be returned |

**Returns**

a vector of methods that match the specified name

Definition at line 43 of file FunctionProxy.cc.

#### 3.6.3.5 virtual bool rootJS::FunctionProxy::isConst ( ) `[inline]`, `[virtual]`

Check if this proxy encapsulates a constant.

**Returns**

true if this ProxyObject encapsulates a constant

Implements rootJS::Proxy.

Definition at line 110 of file FunctionProxy.h.

#### 3.6.3.6 virtual bool rootJS::FunctionProxy::isGlobal ( ) `[inline]`, `[virtual]`

Check if this proxy encapsulates a global.

**Returns**

true if this ProxyObject encapsulates a global

Implements rootJS::Proxy.

Definition at line 120 of file FunctionProxy.h.

#### 3.6.3.7 virtual bool rootJS::FunctionProxy::isStatic ( ) `[inline]`, `[virtual]`

Check if this proxy encapsulates a static.

**Returns**

true if this ProxyObject encapsulates a static

Implements rootJS::Proxy.

Definition at line 130 of file FunctionProxy.h.

#### 3.6.3.8 virtual bool rootJS::FunctionProxy::isTemplate ( ) `[inline]`, `[virtual]`

Check if this proxy encapsulates a template.

**Returns**

true if this ProxyObject encapsulates a template

Implements rootJS::Proxy.

Definition at line 140 of file FunctionProxy.h.

#### 3.6.3.9 void rootJS::FunctionProxy::setSelfAddress ( void ∗ *addr* ) [inline]

TODO: verify

Check whether the arguments encapsulated in the FunctionCallbackInfo are valid arguments to the function. The parameters are then wrapped in proxies so they can be<char> v; used by the call method.

**Parameters**

| | |
|---|---|
| *args* | contains the arguments which shall be validated |

**Returns**

an array of proxies for the validated arguments std::vector<ObjectProxy∗> validateArgs(v8::Function-CallbackInfo<v8::Value> args);

Determines which overloaded function is wanted

**Parameters**

| | |
|---|---|
| *info* | The info of the overloaded function |

**Returns**

true if the overloaded function is found bool determineOverload(const v8::Local<v8::Array>& info); Sets the address of the function

**Parameters**

| | |
|---|---|
| *addr* | The address the function will be set to |

Definition at line 100 of file FunctionProxy.h.

The documentation for this class was generated from the following files:

- src/FunctionProxy.h
- src/FunctionProxy.cc

## 3.7 rootJS::FunctionProxyFactory Class Reference

### Static Public Member Functions

- static FunctionProxy ∗ createFunctionProxy (TFunction ∗function, TClass ∗scope)
- static TFunction ∗ determineFunction (std::string const &name, TClass ∗scope, const v8::Local< v8::Array > args)
- static FunctionProxy ∗ fromArgs (std::string const &name, TClass ∗scope, const v8::Local< v8::Array > args)

### 3.7.1 Detailed Description

Definition at line 23 of file FunctionProxyFactory.h.

### 3.7.2 Member Function Documentation

#### 3.7.2.1 FunctionProxy ∗ rootJS::FunctionProxyFactory::createFunctionProxy ( TFunction ∗ *function,* TClass ∗ *scope* ) [static]

Create a new FunctionProxy of the given function

**Parameters**

| | |
|---:|---|
| *function* | the function to be proxied |
| *scope* | the type of the instance that will be created |

**Returns**

the pointer to the newly created FunctionProxy

Definition at line 33 of file FunctionProxyFactory.cc.

#### 3.7.2.2 TFunction ∗ rootJS::FunctionProxyFactory::determineFunction ( std::string const & *name*, TClass ∗ *scope*, const v8::Local< v8::Array > *args* ) `[static]`

Uses the parameters to determine which function is to be called up

**Parameters**

| | |
|---:|---|
| *name* | the name of the function |
| *scope* | the type of the instance that will be created |
| *args* | the arguments of the function |

**Returns**

the pointer to the function which was determined

Definition at line 39 of file FunctionProxyFactory.cc.

#### 3.7.2.3 FunctionProxy ∗ rootJS::FunctionProxyFactory::fromArgs ( std::string const & *name*, TClass ∗ *scope*, const v8::Local< v8::Array > *args* ) `[static]`

Determines which overloaded function should be called up

**Parameters**

| | |
|---:|---|
| *name* | the name of the function |
| *scope* | the type of the instance that will be created |
| *args* | the arguments of the function |

**Returns**

the pointer to the function proxy of the overloaded function

Definition at line 93 of file FunctionProxyFactory.cc.

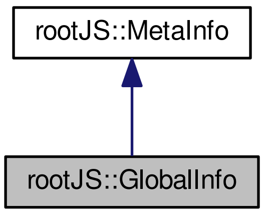
The documentation for this class was generated from the following files:

- src/FunctionProxyFactory.h
- src/FunctionProxyFactory.cc

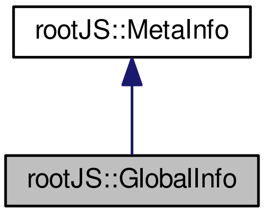
## 3.8 rootJS::GlobalInfo Class Reference

```
#include <GlobalInfo.h>
```

Inheritance diagram for rootJS::GlobalInfo:



Collaboration diagram for rootJS::GlobalInfo:



## Public Member Functions

- **GlobalInfo** (const TGlobal &type)
- virtual Long_t **GetOffset** ()
- virtual bool isGlobal ()
- virtual bool isConst ()
- virtual bool isStatic ()
- virtual const char ∗ getTypeName ()
- virtual const char ∗ getName ()
- virtual GlobalInfo ∗ clone ()

## Additional Inherited Members

### 3.8.1 Detailed Description

This class contains the info for a TGlobal

Definition at line 14 of file GlobalInfo.h.

## 3.8.2 Member Function Documentation

### 3.8.2.1 GlobalInfo ∗ rootJS::GlobalInfo::clone ( ) [virtual]

Makes a clone of the MetaInfo instance.

**Returns**

Pointer to the cloned MetaInfo instance

Implements rootJS::MetaInfo.

Definition at line 44 of file GlobalInfo.cc.

### 3.8.2.2 const char ∗ rootJS::GlobalInfo::getName ( ) [virtual]

Returns the name of the TObject.

**Returns**

name of the TObject

Reimplemented from rootJS::MetaInfo.

Definition at line 39 of file GlobalInfo.cc.

### 3.8.2.3 const char ∗ rootJS::GlobalInfo::getTypeName ( ) [virtual]

Returns the typename of the TObject.

**Returns**

Typename of the TObject

Implements rootJS::MetaInfo.

Definition at line 34 of file GlobalInfo.cc.

### 3.8.2.4 bool rootJS::GlobalInfo::isConst ( ) [virtual]

Checks if the TObject is a constant.

**Returns**

If the TObject is a constant

Implements rootJS::MetaInfo.

Definition at line 24 of file GlobalInfo.cc.

### 3.8.2.5 bool rootJS::GlobalInfo::isGlobal ( ) [virtual]

Checks if the TObject is global.

**Returns**

If the TObject is global

Reimplemented from rootJS::MetaInfo.

Definition at line 19 of file GlobalInfo.cc.

#### 3.8.2.6 bool rootJS::GlobalInfo::isStatic ( ) [virtual]

Checks if the TObject is static.

**Returns**

If the TObject is static

Implements rootJS::MetaInfo.

Definition at line 29 of file GlobalInfo.cc.

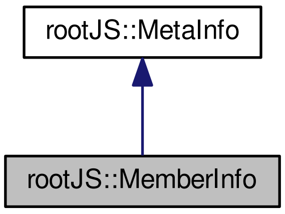
The documentation for this class was generated from the following files:

- src/GlobalInfo.h
- src/GlobalInfo.cc

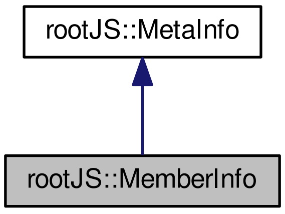
## 3.9 rootJS::MemberInfo Class Reference

`#include <MemberInfo.h>`

Inheritance diagram for rootJS::MemberInfo:



Collaboration diagram for rootJS::MemberInfo:



### Public Member Functions

- **MemberInfo** (const TDataMember &, void ∗baseAddress)

- virtual Long_t getOffset ()
- virtual bool isGlobal ()
- virtual bool isConst ()
- virtual bool isStatic ()
- virtual const char ∗ getTypeName ()
- virtual const char ∗ getName ()
- virtual MemberInfo ∗ clone ()

## Additional Inherited Members

### 3.9.1 Detailed Description

This class contains the info for a TDataMember

Definition at line 14 of file MemberInfo.h.

### 3.9.2 Member Function Documentation

#### 3.9.2.1 **MemberInfo ∗ rootJS::MemberInfo::clone ( )** `[virtual]`

Makes a clone of the MetaInfo instance.

**Returns**

Pointer to the cloned MetaInfo instance

Implements rootJS::MetaInfo.

Definition at line 42 of file MemberInfo.cc.

#### 3.9.2.2 **const char ∗ rootJS::MemberInfo::getName ( )** `[virtual]`

Returns the name of the TObject.

**Returns**

name of the TObject

Reimplemented from rootJS::MetaInfo.

Definition at line 37 of file MemberInfo.cc.

#### 3.9.2.3 **Long_t rootJS::MemberInfo::getOffset ( )** `[virtual]`

Get the offset. This calls up the TDataMember::GetOffset() function.

**Returns**

The offset

Reimplemented from rootJS::MetaInfo.

Definition at line 12 of file MemberInfo.cc.

#### 3.9.2.4 const char * rootJS::MemberInfo::getTypeName ( ) [virtual]

Returns the typename of the TObject.

**Returns**

Typename of the TObject

Implements rootJS::MetaInfo.

Definition at line 32 of file MemberInfo.cc.

#### 3.9.2.5 bool rootJS::MemberInfo::isConst ( ) [virtual]

Checks if the TObject is a constant.

**Returns**

If the TObject is a constant

Implements rootJS::MetaInfo.

Definition at line 22 of file MemberInfo.cc.

#### 3.9.2.6 bool rootJS::MemberInfo::isGlobal ( ) [virtual]

Checks if the TObject is global.

**Returns**

If the TObject is global

Reimplemented from rootJS::MetaInfo.

Definition at line 17 of file MemberInfo.cc.

#### 3.9.2.7 bool rootJS::MemberInfo::isStatic ( ) [virtual]

Checks if the TObject is static.

**Returns**

If the TObject is static

Implements rootJS::MetaInfo.

Definition at line 27 of file MemberInfo.cc.

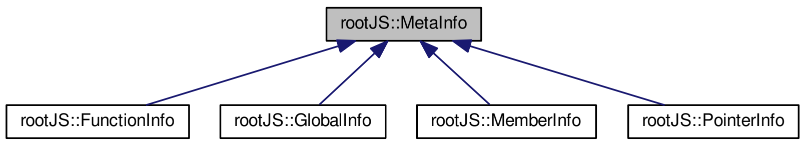
The documentation for this class was generated from the following files:

- src/MemberInfo.h
- src/MemberInfo.cc

## 3.10 rootJS::MetaInfo Class Reference

```
#include <MetaInfo.h>
```

Inheritance diagram for rootJS::MetaInfo:



### Public Member Functions

- MetaInfo (void ∗baseAddress)
- virtual bool isGlobal ()
- virtual Long_t getOffset ()
- virtual bool isConst ()=0
- virtual bool isStatic ()=0
- virtual const char ∗ getTypeName ()=0
- virtual const char ∗ getName ()
- virtual void ∗ getBaseAddress ()
- virtual void ∗ getAddress ()
- virtual MetaInfo ∗ clone ()=0

### Protected Attributes

- void ∗ baseAddress

## 3.10.1 Detailed Description

This class encapsulates the differences in behaviour between TMember and TGlobal

Definition at line 13 of file MetaInfo.h.

## 3.10.2 Constructor & Destructor Documentation

### 3.10.2.1 rootJS::MetaInfo::MetaInfo ( void ∗ *baseAddress* ) `[inline]`

Creates MetaInfo with a specific TObject and its base address

Definition at line 24 of file MetaInfo.h.

## 3.10.3 Member Function Documentation

### 3.10.3.1 virtual **MetaInfo**∗ rootJS::MetaInfo::clone ( ) `[pure virtual]`

Makes a clone of the MetaInfo instance.

**Returns**

Pointer to the cloned MetaInfo instance

Implemented in rootJS::PointerInfo, rootJS::FunctionInfo, rootJS::GlobalInfo, and rootJS::MemberInfo.

#### 3.10.3.2 virtual void∗ rootJS::MetaInfo::getAddress ( ) [inline], [virtual]

Returns the address of the TObject.

**Returns**

Address of the TObject

Reimplemented in rootJS::PointerInfo.

Definition at line 91 of file MetaInfo.h.

#### 3.10.3.3 virtual void∗ rootJS::MetaInfo::getBaseAddress ( ) [inline], [virtual]

Returns the base address of the TObject.

**Returns**

Base address of the TObject

Definition at line 82 of file MetaInfo.h.

#### 3.10.3.4 virtual const char∗ rootJS::MetaInfo::getName ( ) [inline], [virtual]

Returns the name of the TObject.

**Returns**

name of the TObject

Reimplemented in rootJS::FunctionInfo, rootJS::GlobalInfo, and rootJS::MemberInfo.

Definition at line 73 of file MetaInfo.h.

#### 3.10.3.5 virtual Long_t rootJS::MetaInfo::getOffset ( ) [inline], [virtual]

Get the offset. This calls up the TDataMember::GetOffset() function.

**Returns**

The offset

Reimplemented in rootJS::FunctionInfo, and rootJS::MemberInfo.

Definition at line 46 of file MetaInfo.h.

#### 3.10.3.6 virtual const char∗ rootJS::MetaInfo::getTypeName ( ) [pure virtual]

Returns the typename of the TObject.

**Returns**

Typename of the TObject

Implemented in rootJS::PointerInfo, rootJS::FunctionInfo, rootJS::GlobalInfo, and rootJS::MemberInfo.

#### 3.10.3.7 virtual bool rootJS::MetaInfo::isConst ( ) [pure virtual]

Checks if the TObject is a constant.

**Returns**

If the TObject is a constant

Implemented in rootJS::PointerInfo, rootJS::FunctionInfo, rootJS::GlobalInfo, and rootJS::MemberInfo.

#### 3.10.3.8 virtual bool rootJS::MetaInfo::isGlobal ( ) [inline], [virtual]

Checks if the TObject is global.

**Returns**

If the TObject is global

Reimplemented in rootJS::FunctionInfo, rootJS::GlobalInfo, rootJS::MemberInfo, and rootJS::PointerInfo.

Definition at line 36 of file MetaInfo.h.

#### 3.10.3.9 virtual bool rootJS::MetaInfo::isStatic ( ) [pure virtual]

Checks if the TObject is static.

**Returns**

If the TObject is static

Implemented in rootJS::PointerInfo, rootJS::FunctionInfo, rootJS::GlobalInfo, and rootJS::MemberInfo.

### 3.10.4 Member Data Documentation

#### 3.10.4.1 void∗ rootJS::MetaInfo::baseAddress [protected]

The base address of the specific TObject

Definition at line 19 of file MetaInfo.h.

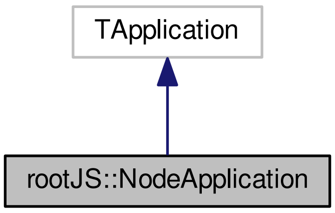
The documentation for this class was generated from the following file:

- src/MetaInfo.h

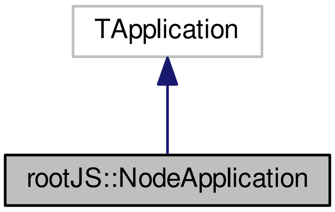
## 3.11 rootJS::NodeApplication Class Reference

```
#include <NodeApplication.h>
```

Inheritance diagram for rootJS::NodeApplication:



Collaboration diagram for rootJS::NodeApplication:



## Public Member Functions

- NodeApplication (const char ∗acn, Int_t ∗argc, char ∗∗argv)
- virtual ∼NodeApplication ()

## Static Public Member Functions

- static Bool_t CreateNodeApplication ()
- static Bool_t InitROOTGlobals ()

### 3.11.1 Detailed Description

NodeApplication is used to handle ROOT GUIs

Definition at line 10 of file NodeApplication.h.

### 3.11.2 Constructor & Destructor Documentation

#### 3.11.2.1 rootJS::NodeApplication::NodeApplication ( const char ∗ *acn,* Int_t ∗ *argc,* char ∗∗ *argv* )

Constructor for NodeApplication Accepts commandline arguments

**Parameters**

| | |
|---|---|
| *acn* | Application name |
| *argc* | number of parameters |
| *argv* | actual parameters |

Definition at line 12 of file NodeApplication.cc.

#### 3.11.2.2 virtual rootJS::NodeApplication::∼NodeApplication ( ) `[inline]`, `[virtual]`

Destructor for NodeApplication

Definition at line 26 of file NodeApplication.h.

### 3.11.3 Member Function Documentation

#### 3.11.3.1 Bool_t rootJS::NodeApplication::CreateNodeApplication ( ) `[static]`

Instamciates a new NdoeApplication and puts it to the right place (gApplicaiton)

Definition at line 25 of file NodeApplication.cc.

#### 3.11.3.2 Bool_t rootJS::NodeApplication::InitROOTGlobals ( ) `[static]`

This method should be used to initialize everything root needs to function prperly

Definition at line 42 of file NodeApplication.cc.

The documentation for this class was generated from the following files:

- src/NodeApplication.h
- src/NodeApplication.cc

## 3.12 rootJS::NodeHandler Class Reference

```
#include <NodeHandler.h>
```

### Public Member Functions

- v8::Local< v8::Object > **getExports** (void)

### Static Public Member Functions

- static void initialize (v8::Local< v8::Object >, v8::Local< v8::Object >)

### 3.12.1 Detailed Description

The NodeHandler is the main entry point when you require rootJS

Definition at line 16 of file NodeHandler.h.

### 3.12.2 Member Function Documentation

#### 3.12.2.1 void rootJS::NodeHandler::initialize ( v8::Local< v8::Object > *exports*, v8::Local< v8::Object > *module* ) `[static]`

The method which starts rootJS.

Definition at line 23 of file NodeHandler.cc.

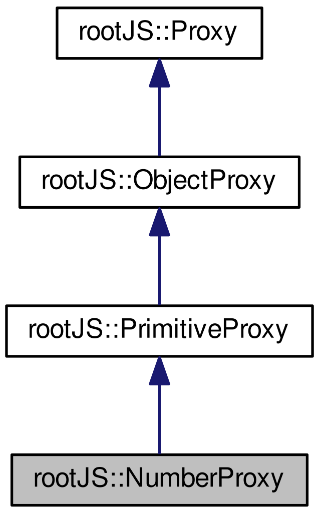
The documentation for this class was generated from the following files:

- src/NodeHandler.h
- src/NodeHandler.cc

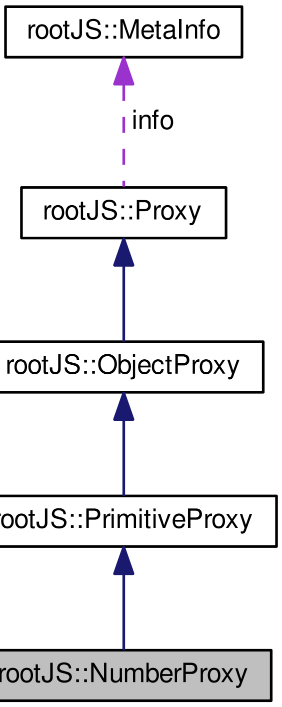
## 3.13 rootJS::NumberProxy Class Reference

`#include <NumberProxy.h>`

Inheritance diagram for rootJS::NumberProxy:



Collaboration diagram for rootJS::NumberProxy:



## Public Member Functions

- NumberProxy (MetaInfo &info, TClass ∗scope)
- virtual v8::Local< v8::Value > get ()
- virtual void setValue (v8::Local< v8::Value > value)

## Static Public Member Functions

- static bool isNumber (std::string type)

- static ObjectProxy ∗ intConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ uintConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ shortConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ ushortConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ floatConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ doubleConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ ldoubleConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ longConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ ulongConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ llongConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ ullongConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ _int64Construct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ u_int64Construct (MetaInfo &info, TClass ∗scope)

## Additional Inherited Members

### 3.13.1 Detailed Description

The NumberProxy is the proxy between C++ numbers and JavaScript number. The NumberProxy uses a C++ macro to map all C++ numbers to JavaScript numbers, and all number are casted to doubles, as doubles are the number type supported by JavaScipt.

Definition at line 29 of file NumberProxy.h.

### 3.13.2 Constructor & Destructor Documentation

#### 3.13.2.1 rootJS::NumberProxy::NumberProxy ( **MetaInfo** & *info,* TClass ∗ *scope* )

Create a new NumberProxy.

**Parameters**

| | |
|---:|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 15 of file NumberProxy.cc.

### 3.13.3 Member Function Documentation

#### 3.13.3.1 **ObjectProxy** ∗ rootJS::NumberProxy::_int64Construct ( **MetaInfo** & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---:|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 85 of file NumberProxy.cc.

#### 3.13.3.2 **ObjectProxy** ∗ rootJS::NumberProxy::doubleConstruct ( **MetaInfo** & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---:|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 76 of file NumberProxy.cc.

#### 3.13.3.3 **ObjectProxy** ∗ rootJS::NumberProxy::floatConstruct ( **MetaInfo** & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---:|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 88 of file NumberProxy.cc.

#### 3.13.3.4 v8::Local< v8::Value > rootJS::NumberProxy::get ( ) [virtual]

Return the encapsulating javascript value.

**Returns**

the encapsulating javascript value

Reimplemented from rootJS::ObjectProxy.

Definition at line 22 of file NumberProxy.cc.

#### 3.13.3.5 ObjectProxy ∗ rootJS::NumberProxy::intConstruct ( MetaInfo & *info*, TClass ∗ *scope* ) [static]

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---:|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 70 of file NumberProxy.cc.

#### 3.13.3.6 bool rootJS::NumberProxy::isNumber ( std::string *type* ) [static]

Check if the type is a number type.

**Parameters**

| | |
|---:|---|
| *type* | the type to be checked |

**Returns**

if the type is a number type

Definition at line 9 of file NumberProxy.cc.

#### 3.13.3.7 ObjectProxy ∗ rootJS::NumberProxy::ldoubleConstruct ( MetaInfo & *info*, TClass ∗ *scope* ) [static]

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---:|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 77 of file NumberProxy.cc.

#### 3.13.3.8 ObjectProxy ∗ rootJS::NumberProxy::llongConstruct ( MetaInfo & *info*, TClass ∗ *scope* ) [static]

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 82 of file NumberProxy.cc.

#### 3.13.3.9 ObjectProxy ∗ rootJS::NumberProxy::longConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 79 of file NumberProxy.cc.

#### 3.13.3.10 void rootJS::NumberProxy::setValue ( v8::Local< v8::Value > *value* ) `[virtual]`

Setter for v8 values, writes new data to memory

**Parameters**

| | |
|---|---|
| *value* | the value set via node, to be stored at the memory address |

Reimplemented from rootJS::ObjectProxy.

Definition at line 91 of file NumberProxy.cc.

#### 3.13.3.11 ObjectProxy ∗ rootJS::NumberProxy::shortConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 73 of file NumberProxy.cc.

#### 3.13.3.12 ObjectProxy ∗ rootJS::NumberProxy::u_int64Construct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 86 of file NumberProxy.cc.

#### 3.13.3.13 ObjectProxy ∗ rootJS::NumberProxy::uintConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 71 of file NumberProxy.cc.

#### 3.13.3.14 ObjectProxy ∗ rootJS::NumberProxy::ullongConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 83 of file NumberProxy.cc.

#### 3.13.3.15 ObjectProxy ∗ rootJS::NumberProxy::ulongConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 80 of file NumberProxy.cc.

#### 3.13.3.16 ObjectProxy ∗ rootJS::NumberProxy::ushortConstruct ( MetaInfo & *info,* TClass ∗ *scope* ) `[static]`

This calls the constructor. We cannot create pointers to constructors, but need to map the constructors in an Factory. This is a macro to declare the constructors for the various number types.

**Parameters**

| | |
|---|---|
| *type* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 74 of file NumberProxy.cc.

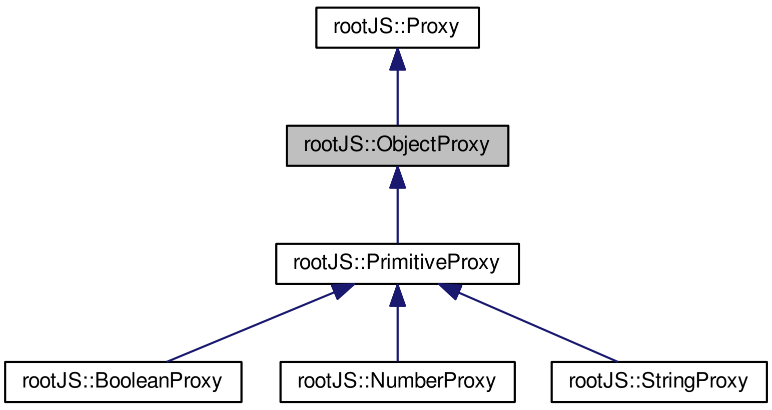
The documentation for this class was generated from the following files:

- src/NumberProxy.h
- src/NumberProxy.cc

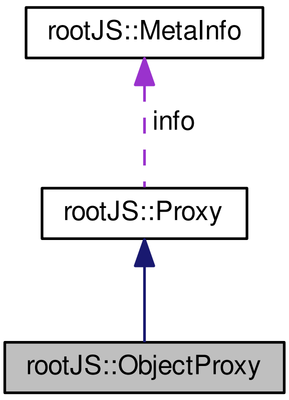
## 3.14 rootJS::ObjectProxy Class Reference

`#include <ObjectProxy.h>`

Inheritance diagram for rootJS::ObjectProxy:



Collaboration diagram for rootJS::ObjectProxy:



## Public Member Functions

- ObjectProxy (MetaInfo &info, TClass ∗scope)
- const char ∗ getTypeName ()
- Long_t getOffset ()
- virtual void set (ObjectProxy &value)
- virtual v8::Local< v8::Value > get ()
- virtual void setProxy (v8::Local< v8::Object > proxy)
- virtual v8::Local< v8::Object > getProxy ()
- virtual void **setValue** (v8::Local< v8::Value > value)
- virtual bool isPrimitive ()
- virtual bool isTemplate ()

- virtual bool isGlobal ()
- virtual bool isConst ()
- virtual bool isStatic ()
- void **registerMallocedSpace** (void ∗)
- v8::Persistent< v8::Object > & **getWeakPeristent** ()

## Protected Attributes

- v8::Persistent< v8::Object > proxy

## Additional Inherited Members

### 3.14.1 Detailed Description

The ObjectProxy class is used to represent ROOT objects. It differentiates between primitive and non-primitive object types.

Definition at line 24 of file ObjectProxy.h.

### 3.14.2 Constructor & Destructor Documentation

#### 3.14.2.1 rootJS::ObjectProxy::ObjectProxy ( **MetaInfo** & *info,* TClass ∗ *scope* )

Create a new ObjectProxy of a TObject.

**Parameters**

| | |
|---|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 11 of file ObjectProxy.cc.

### 3.14.3 Member Function Documentation

#### 3.14.3.1 v8::Local< v8::Value > rootJS::ObjectProxy::get ( ) [virtual]

Return the encapsulating javascript value.

**Returns**

the encapsulating javascript value

Reimplemented in rootJS::NumberProxy, rootJS::StringProxy, and rootJS::BooleanProxy.

Definition at line 49 of file ObjectProxy.cc.

#### 3.14.3.2 Long_t rootJS::ObjectProxy::getOffset ( )

Get the offset

**Returns**

the offset

Definition at line 40 of file ObjectProxy.cc.

#### 3.14.3.3 v8::Local< v8::Object > rootJS::ObjectProxy::getProxy ( ) [virtual]

Return the encapsulating javascript object.

**Returns**

the encapsulating javascript object

Definition at line 58 of file ObjectProxy.cc.

#### 3.14.3.4 const char * rootJS::ObjectProxy::getTypeName ( )

Return the name of the type

**Returns**

the name of the type

Definition at line 36 of file ObjectProxy.cc.

#### 3.14.3.5 bool rootJS::ObjectProxy::isConst ( ) [virtual]

Check if this proxy encapsulates a constant.

**Returns**

true if this ProxyObject encapsulates a constant

Implements rootJS::Proxy.

Definition at line 74 of file ObjectProxy.cc.

#### 3.14.3.6 bool rootJS::ObjectProxy::isGlobal ( ) [virtual]

Check if this proxy encapsulates a global.

**Returns**

true if this ProxyObject encapsulates a global

Implements rootJS::Proxy.

Definition at line 70 of file ObjectProxy.cc.

#### 3.14.3.7 bool rootJS::ObjectProxy::isPrimitive ( ) [virtual]

Check if this proxy encapsulates a primitive type.

**Returns**

true if this ProxyObject encapsulates a primitive data type

Reimplemented in rootJS::PrimitiveProxy.

Definition at line 62 of file ObjectProxy.cc.

#### 3.14.3.8 bool rootJS::ObjectProxy::isStatic ( ) `[virtual]`

Check if this proxy encapsulates a static.

**Returns**

true if this ProxyObject encapsulates a static

Implements rootJS::Proxy.

Definition at line 78 of file ObjectProxy.cc.

#### 3.14.3.9 bool rootJS::ObjectProxy::isTemplate ( ) `[virtual]`

Check if this proxy encapsulates a template.

**Returns**

true if this ProxyObject encapsulates a template

Implements rootJS::Proxy.

Definition at line 66 of file ObjectProxy.cc.

#### 3.14.3.10 void rootJS::ObjectProxy::set ( ObjectProxy & *value* ) `[virtual]`

Assign the specified value to this ObjectProxy.

**Parameters**

| | |
|---|---|
| *value* | the value to assign to this ObjectProxy |

Definition at line 44 of file ObjectProxy.cc.

#### 3.14.3.11 void rootJS::ObjectProxy::setProxy ( v8::Local< v8::Object > *proxy* ) `[virtual]`

Set the encapsulating javascript object.

**Parameters**

| | |
|---|---|
| *proxy* | the encapsulating javascript object |

Definition at line 54 of file ObjectProxy.cc.

### 3.14.4 Member Data Documentation

#### 3.14.4.1 v8::Persistent<v8::Object> rootJS::ObjectProxy::proxy `[protected]`

the exposed javascript object

Definition at line 123 of file ObjectProxy.h.

The documentation for this class was generated from the following files:

- src/ObjectProxy.h
- src/ObjectProxy.cc

## 3.15 rootJS::ObjectProxyFactory Class Reference

### Static Public Member Functions

- static ObjectProxy ∗ **createObjectProxy** (TGlobal &global)
- static ObjectProxy ∗ **createObjectProxy** (MetaInfo &info, TClass ∗scope) throw (std::invalid_argument)
- static ObjectProxy ∗ createObjectProxy (void ∗address, TClass ∗type, v8::Local< v8::Object > proxy)
- static ObjectProxy ∗ **createPrimitiveProxy** (MetaInfo &info, TClass ∗clazz)
- static void **initializeProxyMap** (void)

### 3.15.1 Detailed Description

Definition at line 22 of file ObjectProxyFactory.h.

### 3.15.2 Member Function Documentation

#### 3.15.2.1 ObjectProxy ∗ rootJS::ObjectProxyFactory::createObjectProxy ( void ∗ *address,* TClass ∗ *type,* v8::Local< v8::Object > *proxy* ) `[static]`

Encapsulate the data at the specified address into the specified JavaScript object.

**Parameters**

| | |
|---|---|
| *address* | the address of the data which should be encapsulated |
| *type* | the type of the data which should be encapsulated |
| *proxy* | the JavaScript object used for encapsulation |

**Returns**

a new ObjectProxy holding the specified JavaScript Object for exposure

Definition at line 137 of file ObjectProxyFactory.cc.

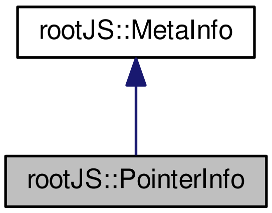
The documentation for this class was generated from the following files:

- src/ObjectProxyFactory.h
- src/ObjectProxyFactory.cc

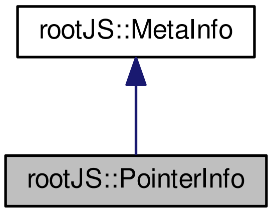
## 3.16 rootJS::PointerInfo Class Reference

```
#include <PointerInfo.h>
```

Inheritance diagram for rootJS::PointerInfo:



Collaboration diagram for rootJS::PointerInfo:



## Public Member Functions

- **PointerInfo** (void ∗baseAddr, const char ∗typeName, int ptrDepth=2)
- virtual bool isGlobal ()
- virtual Long_t **GetOffset** ()
- virtual bool isConst ()
- virtual bool isStatic ()
- virtual const char ∗ getTypeName ()
- virtual PointerInfo ∗ clone ()
- virtual void ∗ getAddress ()

## Protected Attributes

- const char ∗ typeName
- int **ptrDepth**
- void ∗∗ **ptr**
- void ∗∗∗ **ptrptr**

### 3.16.1 Detailed Description

This class contains the info for a pointer

Definition at line 12 of file PointerInfo.h.

### 3.16.2 Member Function Documentation

#### 3.16.2.1 virtual **PointerInfo**∗ rootJS::PointerInfo::clone ( ) `[inline]`, `[virtual]`

Makes a clone of the MetaInfo instance.

**Returns**

Pointer to the cloned MetaInfo instance

Implements rootJS::MetaInfo.

Definition at line 45 of file PointerInfo.h.

#### 3.16.2.2 void ∗ rootJS::PointerInfo::getAddress ( ) `[virtual]`

Returns the address of the TObject.

**Returns**

Address of the TObject

Reimplemented from rootJS::MetaInfo.

Definition at line 8 of file PointerInfo.cc.

#### 3.16.2.3 virtual const char∗ rootJS::PointerInfo::getTypeName ( ) `[inline]`, `[virtual]`

Returns the typename of the TObject.

**Returns**

Typename of the TObject

Implements rootJS::MetaInfo.

Definition at line 40 of file PointerInfo.h.

#### 3.16.2.4 virtual bool rootJS::PointerInfo::isConst ( ) `[inline]`, `[virtual]`

Checks if the TObject is a constant.

**Returns**

If the TObject is a constant

Implements rootJS::MetaInfo.

Definition at line 30 of file PointerInfo.h.

#### 3.16.2.5 virtual bool rootJS::PointerInfo::isGlobal ( ) [inline], [virtual]

Checks if the TObject is global.

**Returns**

If the TObject is global

Reimplemented from rootJS::MetaInfo.

Definition at line 20 of file PointerInfo.h.

#### 3.16.2.6 virtual bool rootJS::PointerInfo::isStatic ( ) [inline], [virtual]

Checks if the TObject is static.

**Returns**

If the TObject is static

Implements rootJS::MetaInfo.

Definition at line 35 of file PointerInfo.h.

### 3.16.3 Member Data Documentation

#### 3.16.3.1 const char∗ rootJS::PointerInfo::typeName [protected]

Type of the pointer

Definition at line 56 of file PointerInfo.h.

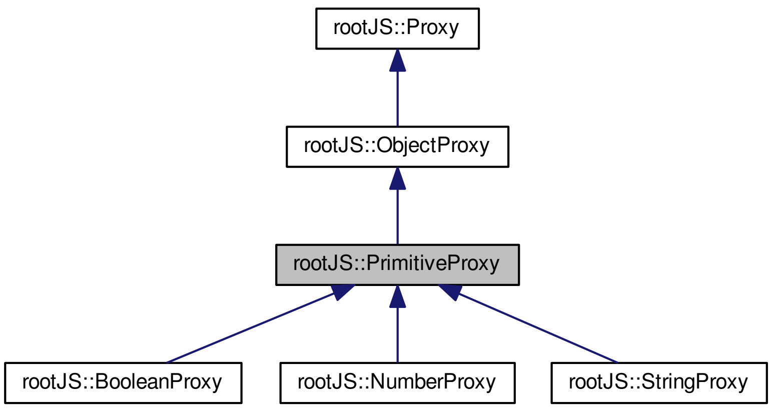
The documentation for this class was generated from the following files:

- src/PointerInfo.h
- src/PointerInfo.cc

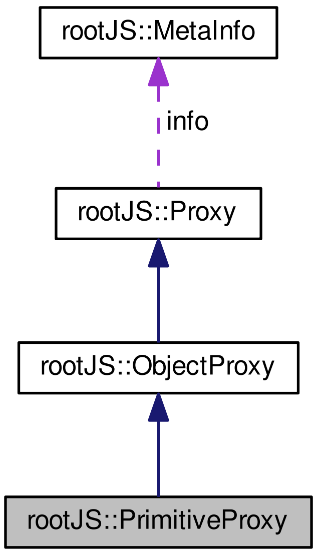
## 3.17 rootJS::PrimitiveProxy Class Reference

`#include <PrimitiveProxy.h>`

Inheritance diagram for rootJS::PrimitiveProxy:



Collaboration diagram for rootJS::PrimitiveProxy:



## Public Member Functions

- PrimitiveProxy (MetaInfo &type, TClass ∗scope)
- virtual bool isPrimitive ()

## Additional Inherited Members

### 3.17.1 Detailed Description

Maps a C++/ROOT primitive to a JavaScript primitive

Definition at line 11 of file PrimitiveProxy.h.

### 3.17.2 Constructor & Destructor Documentation

#### 3.17.2.1 rootJS::PrimitiveProxy::PrimitiveProxy ( **MetaInfo & *type*, TClass ∗ *scope*** )

Create a new PrimitiveProxy.

**Parameters**

| | |
|---|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 5 of file PrimitiveProxy.cc.

### 3.17.3 Member Function Documentation

#### 3.17.3.1 bool rootJS::PrimitiveProxy::isPrimitive ( ) `[virtual]`

Check if this proxy encapsulates a primitive type.

**Returns**

true if this ProxyObject encapsulates a primitive data type

Reimplemented from rootJS::ObjectProxy.

Definition at line 9 of file PrimitiveProxy.cc.

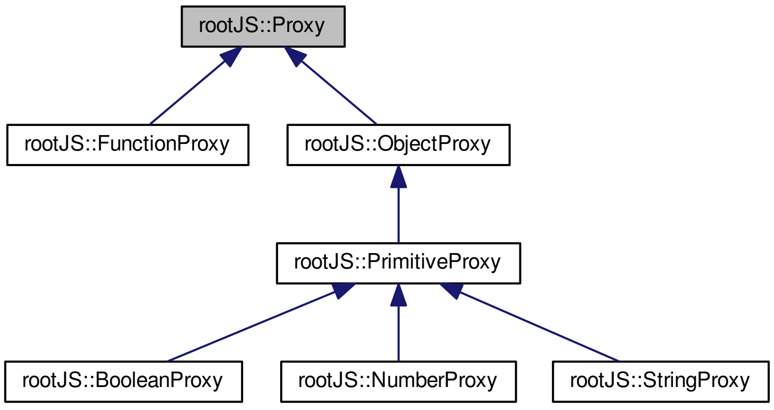
The documentation for this class was generated from the following files:

- src/PrimitiveProxy.h
- src/PrimitiveProxy.cc

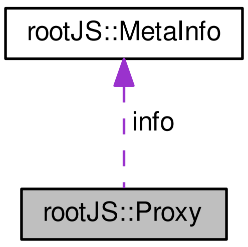
## 3.18 rootJS::Proxy Class Reference

`#include <Proxy.h>`

Inheritance diagram for rootJS::Proxy:



Collaboration diagram for rootJS::Proxy:



## Public Member Functions

- virtual void setAddress (void ∗address)
- virtual void ∗ getAddress ()
- TClass ∗ getScope ()
- virtual MetaInfo ∗ getTypeInfo ()
- virtual bool isTemplate ()=0
- virtual bool isGlobal ()=0
- virtual bool isConst ()=0
- virtual bool isStatic ()=0

## Protected Member Functions

- **Proxy** (MetaInfo &info, TClass ∗scope)

## Protected Attributes

- MetaInfo ∗ info
- TClassRef scope

### 3.18.1 Detailed Description

The proxy super class from which both proxies inherit. The proxies act as intermediary between Node.js and ROOT.

Definition at line 17 of file Proxy.h.

### 3.18.2 Member Function Documentation

#### 3.18.2.1 void ∗ rootJS::Proxy::getAddress ( ) `[virtual]`

get the address of the encapsulated object

**Returns**

the encapsulated object's address

Definition at line 19 of file Proxy.cc.

#### 3.18.2.2 TClass ∗ rootJS::Proxy::getScope ( )

get meta information about the encapsulated objcet's scope

**Returns**

meta information about the scope

Definition at line 24 of file Proxy.cc.

#### 3.18.2.3 MetaInfo ∗ rootJS::Proxy::getTypeInfo ( ) `[virtual]`

get meta information about the encapsulated objcet's type

**Returns**

meta information about the type

Definition at line 29 of file Proxy.cc.

#### 3.18.2.4 virtual bool rootJS::Proxy::isConst ( ) `[pure virtual]`

check if the encapsulated object is constant

**Returns**

if the encapsulated object is constant

Implemented in rootJS::FunctionProxy, and rootJS::ObjectProxy.

#### 3.18.2.5 virtual bool rootJS::Proxy::isGlobal ( ) `[pure virtual]`

check if the encapsulated object is global

**Returns**

if the encapsulated object is global

Implemented in rootJS::FunctionProxy, and rootJS::ObjectProxy.

#### 3.18.2.6 virtual bool rootJS::Proxy::isStatic ( ) `[pure virtual]`

check if the encapsulated object is static

**Returns**

if the encapsulated object is static

Implemented in rootJS::FunctionProxy, and rootJS::ObjectProxy.

#### 3.18.2.7 virtual bool rootJS::Proxy::isTemplate ( ) `[pure virtual]`

check if the encapsulated object is a template

**Returns**

if the encapsulated object is a template

Implemented in rootJS::FunctionProxy, and rootJS::ObjectProxy.

#### 3.18.2.8 void rootJS::Proxy::setAddress ( void * *address* ) `[virtual]`

set the address this proxy points to

**Parameters**

| | |
|---|---|
| *address* | the new address |

Definition at line 14 of file Proxy.cc.

### 3.18.3 Member Data Documentation

#### 3.18.3.1 MetaInfo* rootJS::Proxy::info `[protected]`

type meta information of encapsulated object

Definition at line 73 of file Proxy.h.

#### 3.18.3.2 TClassRef rootJS::Proxy::scope `[protected]`

scope meta information of encapsulated object

Definition at line 74 of file Proxy.h.

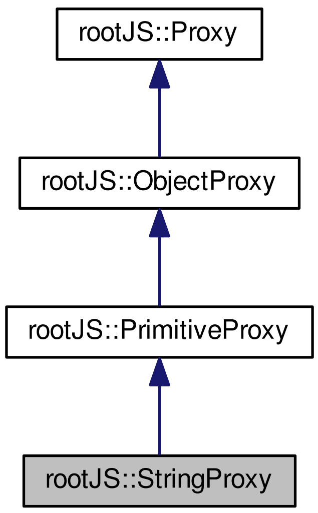
The documentation for this class was generated from the following files:

- src/Proxy.h
- src/Proxy.cc

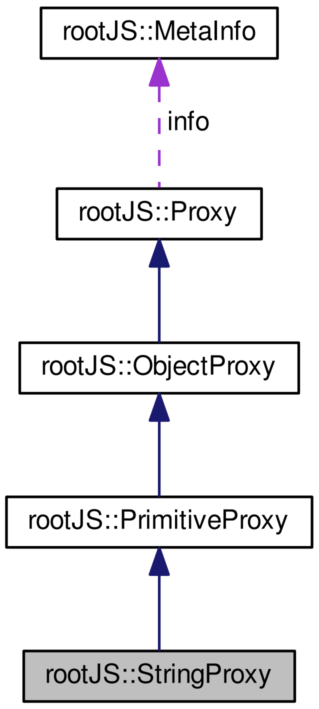
## 3.19 rootJS::StringProxy Class Reference

```
#include <StringProxy.h>
```

Inheritance diagram for rootJS::StringProxy:



Collaboration diagram for rootJS::StringProxy:



## Public Member Functions

- StringProxy (MetaInfo &info, TClass ∗scope)
- virtual v8::Local< v8::Value > get ()
- virtual void setValue (v8::Local< v8::Value > value)

## Static Public Member Functions

- static bool isString (std::string type)
- static ObjectProxy ∗ charConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ stringConstruct (MetaInfo &info, TClass ∗scope)
- static ObjectProxy ∗ tStringConstruct (MetaInfo &info, TClass ∗scope)

## Additional Inherited Members

### 3.19.1 Detailed Description

Maps C++ strings and c-strings to JavaScript strings.

Definition at line 14 of file StringProxy.h.

### 3.19.2 Constructor & Destructor Documentation

#### 3.19.2.1 rootJS::StringProxy::StringProxy ( **MetaInfo** & *info*, TClass $*$ *scope* )

Enum value representing the type Create a new StringProxy.

**Parameters**

| | |
|---|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 8 of file StringProxy.cc.

### 3.19.3 Member Function Documentation

#### 3.19.3.1 ObjectProxy ∗ rootJS::StringProxy::charConstruct ( MetaInfo & *info*, TClass ∗ *scope* ) [static]

Creates a StringProxy based on a const char∗, nullterminated string

**Parameters**

| | |
|---|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 53 of file StringProxy.cc.

#### 3.19.3.2 v8::Local< v8::Value > rootJS::StringProxy::get ( ) [virtual]

Returns a v8 String Copies the c_String which is used to power the Object represented by the MetaInfo object

Reimplemented from rootJS::ObjectProxy.

Definition at line 21 of file StringProxy.cc.

#### 3.19.3.3 bool rootJS::StringProxy::isString ( std::string *type* ) [static]

Check if the type is a boolean type.

**Parameters**

| | |
|---|---|
| *type* | the type to be checked |

**Returns**

if the type is a string type.

Definition at line 11 of file StringProxy.cc.

#### 3.19.3.4 void rootJS::StringProxy::setValue ( v8::Local< v8::Value > *value* ) [virtual]

When the base is an immutable string (std::String, TString) this will set a new value

**Parameters**

| | |
|---|---|
| *value* | The value to be set |

Reimplemented from rootJS::ObjectProxy.

Definition at line 74 of file StringProxy.cc.

#### 3.19.3.5 ObjectProxy ∗ rootJS::StringProxy::stringConstruct ( MetaInfo & *info*, TClass ∗ *scope* ) [static]

Creates a StringProxy based on a const std::string, nullterminated string

**Parameters**

| | |
|---:|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 60 of file StringProxy.cc.

#### 3.19.3.6 ObjectProxy $*$ rootJS::StringProxy::tStringConstruct ( MetaInfo & *info*, TClass $*$ *scope* ) [static]

Creates a StringProxy based on a const TString, nullterminated string

**Parameters**

| | |
|---:|---|
| *info* | the type of the encapsulated object |
| *scope* | the scope of the encapsulated object |

Definition at line 67 of file StringProxy.cc.

The documentation for this class was generated from the following files:

- src/StringProxy.h
- src/StringProxy.cc

## 3.20 rootJS::TemplateFactory Class Reference

### Static Public Member Functions

- static v8::Local< v8::Object > **getInstance** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local< v8::Function > **getConstructor** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::ObjectTemplate > **createNamespaceTemplate** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::ObjectTemplate > **createEnumTemplate** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::ObjectTemplate > **createArrayTemplate** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::FunctionTemplate > **createUnionTemplate** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::FunctionTemplate > **createStructTemplate** (TClass ∗clazz) throw (std::invalid_argument)
- static v8::Local
  < v8::FunctionTemplate > **createClassTemplate** (TClass ∗clazz) throw (std::invalid_argument)

### 3.20.1 Detailed Description

Definition at line 16 of file TemplateFactory.h.

The documentation for this class was generated from the following files:

- src/TemplateFactory.h
- src/TemplateFactory.cc

## 3.21 rootJS::Toolbox Class Reference

```
#include <Toolbox.h>
```

## Public Types

- enum InternalFieldData { **ObjectProxyPtr**, **PropertyMapPtr** }

## Static Public Member Functions

- static void throwException (const std::string &message)
- static void logInfo (const std::string &message)
- static void logError (const std::string &message)

## Static Public Attributes

- static const int **INTERNAL_FIELD_COUNT** = 2

## 3.21.1 Detailed Description

Utility class for various purposes.

Definition at line 11 of file Toolbox.h.

## 3.21.2 Member Enumeration Documentation

### 3.21.2.1 enum rootJS::Toolbox::InternalFieldData

Enumerates the internal fields of v8::Objects.

Definition at line 19 of file Toolbox.h.

## 3.21.3 Member Function Documentation

### 3.21.3.1 void rootJS::Toolbox::logError ( const std::string & *message* ) [static]

Log the specified message at error level.

**Parameters**

| | |
|---|---|
| *the* | message to log |

Definition at line 20 of file Toolbox.cc.

### 3.21.3.2 void rootJS::Toolbox::logInfo ( const std::string & *message* ) [static]

Log the specified message at info level.

**Parameters**

| | |
|---|---|
| *the* | message to log |

Definition at line 14 of file Toolbox.cc.

### 3.21.3.3 void rootJS::Toolbox::throwException ( const std::string & *message* ) [static]

Throws a new v8 exception.

**Parameters**

| | |
|---|---|
| *message* | the exception message |

Definition at line 9 of file Toolbox.cc.

The documentation for this class was generated from the following files:

- src/Toolbox.h
- src/Toolbox.cc

## 3.22 rootJS::Types Class Reference

### Static Public Member Functions

- static bool **isV8Boolean** (v8::Local< v8::Value > value)
- static bool **isV8Number** (v8::Local< v8::Value > value)
- static bool **isV8String** (v8::Local< v8::Value > value)
- static bool **isV8Primitive** (v8::Local< v8::Value > value)

### 3.22.1 Detailed Description

Definition at line 8 of file Types.h.

The documentation for this class was generated from the following files:

- src/Types.h
- src/Types.cc