rootJS

Generated by Doxygen 1.8.6

Wed Feb 10 2016 20:47:34

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

Public Member Functions

- **AsyncRunner** (AsyncFunction func, void ∗param, v8::Persistent< v8::Function, v8::CopyablePersistent-Traits< 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)

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/AsyncRunner.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/AsyncRunner.cc

3.2 rootJS::BooleanProxy Class Reference

`#include <BooleanProxy.h>`

Inherits **rootJS::PrimitiveProxy**.

Public Member Functions

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

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.

3.2.2 Constructor & Destructor Documentation

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

Create a new **BooleanProxy** (p. 5).

Parameters

info	the type of the excapsulated object
scope	the scope of the encapsulated object

3.2.3 Member Function Documentation

3.2.3.1 void rootJS::BooleanProxy::backup ( ) [virtual]

Saves the value to the heap

Reimplemented from **rootJS::ObjectProxy** (p. 32).

3.2.3.2 ObjectProxy * rootJS::BooleanProxy::boolConstruct (MetaInfo & *info,* TClass * *scope*) [static]

Creates a **BooleanProxy** (p. 5), can be derefered to be added to a map

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.2.3.3 v8::Local< v8::Value > rootJS::BooleanProxy::get ( ) [virtual]

Returns a v8 Boolean depending on **MetaInfo** (p. 21)

Returns

boolean depending on **MetaInfo** (p. 21)

Reimplemented from **rootJS::ObjectProxy** (p. 32).

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

Checks if the boolean is backed up. Check if the type is a boolean type.

Parameters

type	the type to be checked

Returns

if the type is a boolean type

3.2.3.5 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** (p. 31).

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/BooleanProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/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 Member Function Documentation

3.3.1.1 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** (p. 31). As result the **ObjectProxy** (p. 31)'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

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

TODO: fill in description

Parameters

info	

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

TODO: fill in description

Parameters

property	
info	

3.3.1.4 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	

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

TODO: fill in description

Parameters

info	

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

TODO: fill in description

Parameters

property	

info	

3.3.1.7 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	

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

TODO: fill in description

Parameters

name	
proxy	

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

TODO: fill in description

Parameters

name	
proxy	

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

TODO: fill in description

Parameters

info	

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

TODO: fill in description

Parameters

property	

info	

3.3.1.12 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	

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/CallbackHandler.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/CallbackHandler.cc

3.4 rootJS::FunctionInfo Class Reference

`#include <FunctionInfo.h>`

Inherits **rootJS::MetaInfo**.

Public Member Functions

- **FunctionInfo** (TFunction *__func__)
- virtual bool **isGlobal** ()
- virtual Long_t **GetOffset** ()
- virtual bool **isConst** ()
- virtual bool **isStatic** ()
- virtual const char * **getTypeName** ()
- virtual **MetaInfo** * **clone** ()

Protected Attributes

- TFunction * **func**

3.4.1 Detailed Description

This class contains the info for a TFunction

3.4.2 Member Function Documentation

3.4.2.1 virtual MetaInfo* rootJS::FunctionInfo::clone ( ) [inline], [virtual]

Makes a clone of the **MetaInfo** (p. 21) instance.

Returns

Pointer to the cloned **MetaInfo** (p. 21) instance

Implements **rootJS::MetaInfo** (p. 22).

3.4.2.2 virtual const char* rootJS::FunctionInfo::getTypeName ( ) [inline], [virtual]

Returns the typename of the TObject.

Returns

Typename of the TObject

Implements **rootJS::MetaInfo** (p. 23).

3.4.2.3 virtual bool rootJS::FunctionInfo::isConst ( ) [inline], [virtual]

Checks if the TObject is a constant.

Returns

If the TObject is a constant

Implements **rootJS::MetaInfo** (p. 23).

3.4.2.4 virtual bool rootJS::FunctionInfo::isGlobal ( ) [inline], [virtual]

Checks if the TObject is global.

Returns

If the TObject is global

Reimplemented from **rootJS::MetaInfo** (p. 23).

3.4.2.5 virtual bool rootJS::FunctionInfo::isStatic ( ) [inline], [virtual]

Checks if the TObject is static.

Returns

If the TObject is static

Implements **rootJS::MetaInfo** (p. 23).

3.4.3 Member Data Documentation

3.4.3.1 TFunction* rootJS::FunctionInfo::func [protected]

The function the **FunctionInfo** (p. 11) is holding.

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/FunctionInfo.h

3.5 rootJS::FunctionProxy Class Reference

`#include <FunctionProxy.h>`

Inherits **rootJS::Proxy**.

Public Member Functions

- **FunctionProxy** (void ∗address, **FunctionInfo** &mode, TFunction ∗function, TClass ∗**scope**)
- std::vector< **ObjectProxy** ∗ > **validateArgs** (v8::FunctionCallbackInfo< v8::Value > args)
- void **prepareCall** (const v8::Local< v8::Array > &args)
- **ObjectProxy** ∗ **call** ()
- virtual bool **isConst** ()
- virtual bool **isGlobal** ()
- virtual bool **isStatic** ()
- virtual bool **isTemplate** ()
- bool **determineOverload** (const v8::Local< v8::Array > &**info**)
- void **setSelfAddress** (void ∗addr)
- **FunctionProxy** ∗ **clone** ()

Static Public Member Functions

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

Additional Inherited Members

3.5.1 Detailed Description

Represents a ROOT callable and provides functionality to invoke those callables. Also acts as a static cache for already created **FunctionProxy** (p. 12) objects.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 rootJS::FunctionProxy::FunctionProxy (void ∗ *address,* FunctionInfo & *mode,* TFunction ∗ *function,* TClass ∗ *scope*)

Create a new **FunctionProxy** (p. 12).

Parameters

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

3.5.3 Member Function Documentation

3.5.3.1 ObjectProxy ∗ rootJS::FunctionProxy::call ( )

Invokes the proxied function.

Parameters

args	the arguments for the function call.

Returns

the function's return value encasulated in an **ObjectProxy** (p. 31)

3.5.3.2 FunctionProxy * rootJS::FunctionProxy::clone ( )

Makes a clone of the current **FunctionProxy** (p. 12)

Returns

A pointer to the clone

3.5.3.3 bool rootJS::FunctionProxy::determineOverload (const v8::Local< v8::Array > & *info*)

Determines which overloaded function is wanted

Parameters

info	The info of the overloaded function

Returns

true if the overloaded function is found

3.5.3.4 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

3.5.3.5 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

3.5.3.6 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** (p. 39).

3.5.3.7 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** (p. 39).

3.5.3.8 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** (p. 39).

3.5.3.9 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** (p. 39).

3.5.3.10 void rootJS::FunctionProxy::setSelfAddress (void * *addr*) [inline]

Sets the address of the function

Parameters

addr	The address the function will be set to

3.5.3.11 std::vector< ObjectProxy * > rootJS::FunctionProxy::validateArgs (v8::FunctionCallbackInfo< v8::Value > *args*)

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

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/FunctionProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/FunctionProxy.cc

3.6 rootJS::FunctionProxyFactory Class Reference

Static Public Member Functions

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

3.6.1 Member Function Documentation

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

Create a new **FunctionProxy** (p. 12) 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** (p. 12)

3.6.1.2 void ∗ rootJS::FunctionProxyFactory::createInstance (std::string *name,* TClass ∗ *scope,* v8::Local< v8::Array > *args*) [static]

Create a new instance of the specified type using the constructor suitable to the supplied arguments.

Parameters

name	the name of the constructor function
scope	the type of the instance that will be created
args	the arguments to call the constructor with

Returns

the address to the newly created instance of the specified type or nullptr if no suitable constructor was found

3.6.1.3 TFunction ∗ rootJS::FunctionProxyFactory::determineFunction (std::string *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

3.6.1.4 FunctionProxy * rootJS::FunctionProxyFactory::fromArgs (std::string *name,* TClass * *scope,* 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

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/FunctionProxyFactory.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/FunctionProxyFactory.cc

3.7 rootJS::GlobalInfo Class Reference

`#include <GlobalInfo.h>`

Inherits **rootJS::MetaInfo**.

Public Member Functions

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

Public Attributes

- const TGlobal & **currentObject**

Additional Inherited Members

3.7.1 Detailed Description

This class contains the info for a TGlobal

3.7.2 Member Function Documentation

3.7.2.1 virtual MetaInfo* rootJS::GlobalInfo::clone ( ) [inline], [virtual]

Makes a clone of the **MetaInfo** (p. 21) instance.

Returns

Pointer to the cloned **MetaInfo** (p. 21) instance

Implements **rootJS::MetaInfo** (p. 22).

3.7.2.2 const char * rootJS::GlobalInfo::getTypeName ( ) [virtual]

Returns the typename of the TObject.

Returns

Typename of the TObject

Implements **rootJS::MetaInfo** (p. 23).

3.7.2.3 bool rootJS::GlobalInfo::isConst ( ) [virtual]

Checks if the TObject is a constant.

Returns

If the TObject is a constant

Implements **rootJS::MetaInfo** (p. 23).

3.7.2.4 bool rootJS::GlobalInfo::isGlobal ( ) [virtual]

Checks if the TObject is global.

Returns

If the TObject is global

Reimplemented from **rootJS::MetaInfo** (p. 23).

3.7.2.5 virtual bool rootJS::GlobalInfo::isStatic ( ) [inline], [virtual]

Checks if the TObject is static.

Returns

If the TObject is static

Implements **rootJS::MetaInfo** (p. 23).

3.7.3 Member Data Documentation

3.7.3.1 const TGlobal& rootJS::GlobalInfo::currentObject

The type the **GlobalInfo** (p. 18) is holding.

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/GlobalInfo.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/GlobalInfo.cc

3.8 rootJS::MemberInfo Class Reference

`#include <MemberInfo.h>`

Inherits **rootJS::MetaInfo**.

Public Member Functions

- **MemberInfo** (const TDataMember &, void ∗**baseAddress**)
- virtual bool **isGlobal** ()
- virtual Long_t **getOffset** ()
- virtual bool **isConst** ()
- virtual bool **isStatic** ()
- virtual const char ∗ **getTypeName** ()
- virtual **MetaInfo** ∗ **clone** ()

Public Attributes

- const TDataMember & **currentObject**

Additional Inherited Members

3.8.1 Detailed Description

This class contains the info for a TDataMember

3.8.2 Member Function Documentation

3.8.2.1 virtual MetaInfo∗ rootJS::MemberInfo::clone ( ) [inline], [virtual]

Makes a clone of the **MetaInfo** (p. 21) instance.

Returns

Pointer to the cloned **MetaInfo** (p. 21) instance

Implements **rootJS::MetaInfo** (p. 22).

3.8.2.2 Long_t rootJS::MemberInfo::getOffset ( ) [virtual]

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

Returns

The offset

Reimplemented from **rootJS::MetaInfo** (p. 23).

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

Returns the typename of the TObject.

Returns

Typename of the TObject

Implements **rootJS::MetaInfo** (p. 23).

3.8.2.4 bool rootJS::MemberInfo::isConst ( ) [virtual]

Checks if the TObject is a constant.

Returns

If the TObject is a constant

Implements **rootJS::MetaInfo** (p. 23).

3.8.2.5 bool rootJS::MemberInfo::isGlobal ( ) [virtual]

Checks if the TObject is global.

Returns

If the TObject is global

Reimplemented from **rootJS::MetaInfo** (p. 23).

3.8.2.6 bool rootJS::MemberInfo::isStatic ( ) [virtual]

Checks if the TObject is static.

Returns

If the TObject is static

Implements **rootJS::MetaInfo** (p. 23).

3.8.3 Member Data Documentation

3.8.3.1 const TDataMember& rootJS::MemberInfo::currentObject

The type the **MemberInfo** (p. 19) is holding.

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/MemberInfo.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/MemberInfo.cc

3.9 rootJS::MetaInfo Class Reference

`#include <MetaInfo.h>`

Inherited by **rootJS::FunctionInfo**, **rootJS::GlobalInfo**, **rootJS::MemberInfo**, and **rootJS::PointerInfo**.

Public Member Functions

- **MetaInfo** (const TObject &foo, void ∗**baseAddress**)
- virtual bool **isGlobal** ()
- virtual Long_t **getOffset** ()
- virtual bool **isConst** ()=0
- virtual bool **isStatic** ()=0

- virtual const char * **getTypeName** ()=0
- virtual void * **getBaseAddress** ()
- virtual void * **getAddress** ()
- virtual **MetaInfo** * **clone** ()=0

Protected Attributes

- void * **baseAddress**

3.9.1 Detailed Description

This class encapsulates the differences in behaviour between TMember and TGlobal

3.9.2 Constructor & Destructor Documentation

3.9.2.1 rootJS::MetaInfo::MetaInfo (const TObject & *foo,* void * *baseAddress*) [inline]

Creates **MetaInfo** (p. 21) with a specific TObject and its base address

3.9.3 Member Function Documentation

3.9.3.1 virtual MetaInfo* rootJS::MetaInfo::clone ( ) [pure virtual]

Makes a clone of the **MetaInfo** (p. 21) instance.

Returns

Pointer to the cloned **MetaInfo** (p. 21) instance

Implemented in **rootJS::FunctionInfo** (p. 11), **rootJS::PointerInfo** (p. 36), **rootJS::MemberInfo** (p. 20), and **rootJS::GlobalInfo** (p. 18).

3.9.3.2 virtual void* rootJS::MetaInfo::getAddress ( ) [inline], [virtual]

Returns the address of the TObject.

Returns

Address of the TObject

3.9.3.3 virtual void* rootJS::MetaInfo::getBaseAddress ( ) [inline], [virtual]

Returns the base address of the TObject.

Returns

Base address of the TObject

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

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

Returns

The offset

Reimplemented in **rootJS::MemberInfo** (p. 20).

3.9.3.5 virtual const char* rootJS::MetaInfo::getTypeName ( ) [pure virtual]

Returns the typename of the TObject.

Returns

Typename of the TObject

Implemented in **rootJS::FunctionInfo** (p. 12), **rootJS::PointerInfo** (p. 36), **rootJS::MemberInfo** (p. 20), and **rootJS::GlobalInfo** (p. 19).

3.9.3.6 virtual bool rootJS::MetaInfo::isConst ( ) [pure virtual]

Checks if the TObject is a constant.

Returns

If the TObject is a constant

Implemented in **rootJS::FunctionInfo** (p. 12), **rootJS::MemberInfo** (p. 21), **rootJS::PointerInfo** (p. 36), and **rootJS::GlobalInfo** (p. 19).

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

Checks if the TObject is global.

Returns

If the TObject is global

Reimplemented in **rootJS::MemberInfo** (p. 21), **rootJS::FunctionInfo** (p. 12), **rootJS::GlobalInfo** (p. 19), and **rootJS::PointerInfo** (p. 36).

3.9.3.8 virtual bool rootJS::MetaInfo::isStatic ( ) [pure virtual]

Checks if the TObject is static.

Returns

If the TObject is static

Implemented in **rootJS::FunctionInfo** (p. 12), **rootJS::MemberInfo** (p. 21), **rootJS::PointerInfo** (p. 36), and **rootJS::GlobalInfo** (p. 19).

3.9.4 Member Data Documentation

3.9.4.1 void* rootJS::MetaInfo::baseAddress [protected]

The base address of the specific TObject

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/MetaInfo.h

3.10 rootJS::NodeApplication Class Reference

`#include <NodeApplication.h>`

Inherits TApplication.

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** ()

3.10.1 Detailed Description

NodeApplication (p. 24) is used to handle ROOT GUIs

3.10.2 Constructor & Destructor Documentation

3.10.2.1 rootJS::NodeApplication::NodeApplication (const char * *acn*, Int_t * *argc*, char ** *argv*)

Constructor for **NodeApplication** (p. 24) Accepts commandline arguments

Parameters

acn	Application name
argc	number of parameters
argv	actual parameters

3.10.2.2 virtual rootJS::NodeApplication::~NodeApplication ( ) [inline], [virtual]

Destructor for **NodeApplication** (p. 24)

3.10.3 Member Function Documentation

3.10.3.1 Bool_t rootJS::NodeApplication::CreateNodeApplication ( ) [static]

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

3.10.3.2 Bool_t rootJS::NodeApplication::InitROOTGlobals ( ) [static]

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

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NodeApplication.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NodeApplication.cc

3.11 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 >)

3.11.1 Detailed Description

The **NodeHandler** (p. 25) is the main entry point when you require rootJS

3.11.2 Member Function Documentation

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

The method which starts rootJS.

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NodeHandler.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NodeHandler.cc

3.12 rootJS::NumberProxy Class Reference

`#include <NumberProxy.h>`

Inherits **rootJS::PrimitiveProxy**.

Public Member Functions

- **NumberProxy** (**MetaInfo** &**info**, TClass *__scope**)
- virtual void **backup** ()
- 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

3.12.1 Detailed Description

The **NumberProxy** (p. 25) is the proxy between C++ numbers and JavaScript number. The **NumberProxy** (p. 25) 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.

3.12.2 Constructor & Destructor Documentation

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

Create a new **NumberProxy** (p. 25).

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.12.3 Member Function Documentation

3.12.3.1 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

3.12.3.2 void rootJS::NumberProxy::backup ( ) [virtual]

Saves the value to the heap

Reimplemented from **rootJS::ObjectProxy** (p. 32).

3.12.3.3 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

3.12.3.4 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

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

Return the encapsulating javascript value.

Returns

the encapsulating javascript value

Reimplemented from **rootJS::ObjectProxy** (p. 32).

3.12.3.6 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

3.12.3.7 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

3.12.3.8 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

3.12.3.9 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

3.12.3.10 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

3.12.3.11 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** (p. 31).

3.12.3.12 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

3.12.3.13 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

3.12.3.14 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

3.12.3.15 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

3.12.3.16 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

3.12.3.17 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

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NumberProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/NumberProxy.cc

3.13 rootJS::ObjectProxy Class Reference

`#include <ObjectProxy.h>`

Inherits **rootJS::Proxy**.

Inherited by **rootJS::PrimitiveProxy**.

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** ()
- virtual void **backup** ()

Protected Attributes

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

Additional Inherited Members

3.13.1 Detailed Description

The **ObjectProxy** (p. 31) class is used to represent ROOT objects. It differentiates between primitive and non-primitive object types.

3.13.2 Constructor & Destructor Documentation

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

Create a new **ObjectProxy** (p. 31) of a TObject.

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.13.3 Member Function Documentation

3.13.3.1 void rootJS::ObjectProxy::backup ( ) [virtual]

Saves the value to the heap

Reimplemented in **rootJS::StringProxy** (p. 41), **rootJS::NumberProxy** (p. 26), and **rootJS::BooleanProxy** (p. 6).

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

Return the encapsulating javascript value.

Returns

the encapsulating javascript value

Reimplemented in **rootJS::NumberProxy** (p. 28), **rootJS::StringProxy** (p. 41), and **rootJS::BooleanProxy** (p. 6).

3.13.3.3 Long_t rootJS::ObjectProxy::getOffset ( )

Get the offset

Returns

the offset

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

Return the encapsulating javascript object.

Returns

the encapsulating javascript object

3.13.3.5 const char * rootJS::ObjectProxy::getTypeName ( )

Return the name of the type

Returns

the name of the type

3.13.3.6 bool rootJS::ObjectProxy::isConst ( ) [virtual]

Check if this proxy encapsulates a constant.

Returns

true if this ProxyObject encapsulates a constant

Implements **rootJS::Proxy** (p. 39).

3.13.3.7 bool rootJS::ObjectProxy::isGlobal ( ) [virtual]

Check if this proxy encapsulates a global.

Returns

true if this ProxyObject encapsulates a global

Implements **rootJS::Proxy** (p. 39).

3.13.3.8 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** (p. 37).

3.13.3.9 bool rootJS::ObjectProxy::isStatic ( ) [virtual]

Check if this proxy encapsulates a static.

Returns

true if this ProxyObject encapsulates a static

Implements **rootJS::Proxy** (p. 39).

3.13.3.10 bool rootJS::ObjectProxy::isTemplate ( ) [virtual]

Check if this proxy encapsulates a template.

Returns

true if this ProxyObject encapsulates a template

Implements **rootJS::Proxy** (p. 39).

3.13.3.11 void rootJS::ObjectProxy::set (ObjectProxy & *value*) [virtual]

Assign the specified value to this **ObjectProxy** (p. 31).

Parameters

value	the value to assign to this **ObjectProxy** (p. 31)

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

Set the encapsulating javascript object.

Parameters

proxy	the encapsulating javascript object

3.13.4 Member Data Documentation

3.13.4.1 v8::Persistent<v8::Object> rootJS::ObjectProxy::proxy [protected]

the exposed javascript object

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/ObjectProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/ObjectProxy.cc

3.14 rootJS::ObjectProxyFactory Class Reference

Static Public Member Functions

- static **ObjectProxy** * **createObjectProxy** (TGlobal &global)
- static **ObjectProxy** * **createObjectProxy** (TDataMember const &type, TClass *scope, **ObjectProxy** &holder)
- static **ObjectProxy** * **createObjectProxy** (**MetaInfo** &info, TClass *scope)

- static **ObjectProxy** ∗ **createObjectProxy** (void ∗address, TClass ∗type, v8::Local< v8::Object > proxy)
- static **ObjectProxy** ∗ **determineProxy** (**MetaInfo** &info, TClass ∗clazz)
- static void **initializeProxyMap** (void)

3.14.1 Member Function Documentation

3.14.1.1 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** (p. 31) holding the specified JavaScript Object for exposure

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/ObjectProxyFactory.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/ObjectProxyFactory.cc

3.15 rootJS::PointerInfo Class Reference

`#include <PointerInfo.h>`

Inherits **rootJS::MetaInfo**.

Public Member Functions

- **PointerInfo** (void ∗baseAddr, const char ∗**typeName**)
- virtual bool **isGlobal** ()
- virtual Long_t **GetOffset** ()
- virtual bool **isConst** ()
- virtual bool **isStatic** ()
- virtual const char ∗ **getTypeName** ()
- virtual **MetaInfo** ∗ **clone** ()

Protected Attributes

- const char ∗ **typeName**

3.15.1 Detailed Description

This class contains the info for a pointer

3.15.2 Member Function Documentation

3.15.2.1 virtual MetaInfo* rootJS::PointerInfo::clone ( ) [inline], [virtual]

Makes a clone of the **MetaInfo** (p. 21) instance.

Returns

Pointer to the cloned **MetaInfo** (p. 21) instance

Implements **rootJS::MetaInfo** (p. 22).

3.15.2.2 virtual const char* rootJS::PointerInfo::getTypeName ( ) [inline], [virtual]

Returns the typename of the TObject.

Returns

Typename of the TObject

Implements **rootJS::MetaInfo** (p. 23).

3.15.2.3 virtual bool rootJS::PointerInfo::isConst ( ) [inline], [virtual]

Checks if the TObject is a constant.

Returns

If the TObject is a constant

Implements **rootJS::MetaInfo** (p. 23).

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

Checks if the TObject is global.

Returns

If the TObject is global

Reimplemented from **rootJS::MetaInfo** (p. 23).

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

Checks if the TObject is static.

Returns

If the TObject is static

Implements **rootJS::MetaInfo** (p. 23).

3.15.3 Member Data Documentation

3.15.3.1 const char* rootJS::PointerInfo::typeName [protected]

Type of the pointer

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/PointerInfo.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/PointerInfo.cc

3.16 rootJS::PrimitiveProxy Class Reference

`#include <PrimitiveProxy.h>`

Inherits **rootJS::ObjectProxy**.

Inherited by **rootJS::BooleanProxy**, **rootJS::NumberProxy**, and **rootJS::StringProxy**.

Public Member Functions

- **PrimitiveProxy** (**MetaInfo** &type, TClass *__scope__)
- virtual bool **isPrimitive** ()

Additional Inherited Members

3.16.1 Detailed Description

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

3.16.2 Constructor & Destructor Documentation

3.16.2.1 rootJS::PrimitiveProxy::PrimitiveProxy (MetaInfo & *type,* TClass * *scope*)

Create a new **PrimitiveProxy** (p. 37).

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.16.3 Member Function Documentation

3.16.3.1 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** (p. 33).

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/PrimitiveProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/PrimitiveProxy.cc

3.17 rootJS::Proxy Class Reference

`#include <Proxy.h>`

Inherited by **rootJS::FunctionProxy**, and **rootJS::ObjectProxy**.

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**

3.17.1 Detailed Description

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

3.17.2 Member Function Documentation

3.17.2.1 void * rootJS::Proxy::getAddress ( ) [virtual]

get the address of the encapsulated object

Returns

the encapsulated object's address

3.17.2.2 TClass * rootJS::Proxy::getScope ( )

get meta information about the encapsulated objcet's scope

Returns

meta information about the scope

3.17.2.3 MetaInfo * rootJS::Proxy::getTypeInfo ( ) [virtual]

get meta information about the encapsulated objcet's type

Returns

meta information about the type

3.17.2.4 virtual bool rootJS::Proxy::isConst ( ) [pure virtual]

check if the encapsulated object is constant

Returns

if the encapsulated object is constant

Implemented in **rootJS::ObjectProxy** (p. 33), and **rootJS::FunctionProxy** (p. 14).

3.17.2.5 virtual bool rootJS::Proxy::isGlobal ( ) [pure virtual]

check if the encapsulated object is global

Returns

if the encapsulated object is global

Implemented in **rootJS::ObjectProxy** (p. 33), and **rootJS::FunctionProxy** (p. 15).

3.17.2.6 virtual bool rootJS::Proxy::isStatic ( ) [pure virtual]

check if the encapsulated object is static

Returns

if the encapsulated object is static

Implemented in **rootJS::ObjectProxy** (p. 34), and **rootJS::FunctionProxy** (p. 15).

3.17.2.7 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** (p. 15), and **rootJS::ObjectProxy** (p. 34).

3.17.2.8 void rootJS::Proxy::setAddress (void * *address*) [virtual]

set the address this proxy points to

Parameters

address	the new address

3.17.3 Member Data Documentation

3.17.3.1 MetaInfo* rootJS::Proxy::info [protected]

type meta information of encapsulated object

3.17.3.2 TClassRef rootJS::Proxy::scope [protected]

scope meta information of encapsulated object

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/Proxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/Proxy.cc

3.18 rootJS::StringProxy Class Reference

`#include <StringProxy.h>`

Inherits **rootJS::PrimitiveProxy**.

Public Member Functions

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

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

3.18.1 Detailed Description

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

3.18.2 Constructor & Destructor Documentation

3.18.2.1 rootJS::StringProxy::StringProxy (MetaInfo & *info,* TClass * *scope*)

Checks if the string is backed up. Create a new **StringProxy** (p. 40).

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.18.3 Member Function Documentation

3.18.3.1 void rootJS::StringProxy::backup ( ) [virtual]

Saves the value to the heap

Reimplemented from **rootJS::ObjectProxy** (p. 32).

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

Creates a **StringProxy** (p. 40) based on a const char*, nullterminated string

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

3.18.3.3 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** (p. 21) object

Reimplemented from **rootJS::ObjectProxy** (p. 32).

3.18.3.4 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.

3.18.3.5 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** (p. 31).

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

Creates a **StringProxy** (p. 40) based on a const std::string, nullterminated string

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

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

Creates a **StringProxy** (p. 40) based on a const TString, nullterminated string

Parameters

info	the type of the encapsulated object
scope	the scope of the encapsulated object

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/StringProxy.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/StringProxy.cc

3.19 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::FunctionTemplate > **createClassTemplate** (TClass *clazz) throw (std::invalid_argument)
- static v8::Local < v8::FunctionTemplate > **createStructTemplate** (TClass *clazz) throw (std::invalid_argument)

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/TemplateFactory.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/TemplateFactory.cc

3.20 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 **log** (const std::string &message)

Static Public Attributes

- static const int **INTERNAL_FIELD_COUNT** = 2

3.20.1 Detailed Description

Utility class for various purposes.

3.20.2 Member Enumeration Documentation

3.20.2.1 enum rootJS::Toolbox::InternalFieldData

Enumerates the internal fields of v8::Objects.

3.20.3 Member Function Documentation

3.20.3.1 void rootJS::Toolbox::log (const std::string & *message*) [static]

Log the specified message.

Parameters

the	message to log

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

Throws a new v8 exception.

Parameters

message	the exception message

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

- /home/sachin/Documents/KIT/PSE/git/rootjs/src/Toolbox.h
- /home/sachin/Documents/KIT/PSE/git/rootjs/src/Toolbox.cc