Karlsruhe Institute of Technology

Software Engineering Practice

WINTER TERM 2015/2016

rootJS

Node.js bindings for ROOT 6

Jonas Schwabe
Theo Beffart
Sachin Rajgopal
Christoph Wolff
Christoph Haas
Maximilian Früh

supervised by
Dr. Marek Szuba

Contents

1 CallbackHandler **2**
1.1 ctorCallback . . . 2
1.2 staticCtorCallback . . . 3
1.3 memberGetterCallback . . . 4
1.4 memberSetterCallback . . . 5
1.5 memberFunctionCallback . . . 6
1.6 staticGetterCallback . . . 7
1.7 staticSetterCallback . . . 8
1.8 staticFunctionCallback . . . 9

2 NodeHandler **10**
2.1 getExports . . . 10

3 NodeApplication **11**
3.1 NodeApplication . . . 11

4 TemplateFactory **12**
4.1 createTemplate . . . 12

5 Proxy **13**
5.1 Proxy . . . 13
5.2 setAddress . . . 14
5.3 getAddress . . . 15
5.4 getType . . . 16
5.5 getScope . . . 17
5.6 isGlobal . . . 18
5.7 isTemplate . . . 19
5.8 isConst . . . 20
5.9 isStatic . . . 21

6 FunctionProxyFactory **22**
6.1 createFunctionProxy . . . 22
6.2 fromArgs . . . 23

7 FunctionProxy **24**
7.1 getCallFunc . . . 24
7.2 getMethodsFromName . . . 25
7.3 FunctionProxy . . . 26
7.4 getType . . . 27
7.5 validateArgs . . . 28
7.6 call . . . 29

8 ObjectProxyFactory **30**
8.1 createObjectProxy . . . 30

9 ObjectProxy **31**
9.1 ObjectProxy . . . 31
9.2 getType . . . 32
9.3 set . . . 33
9.4 get . . . 34
9.5 setProxy . . . 35
9.6 getProxy . . . 36
9.7 isPrimitive . . . 37

1. CallbackHandler

describe class CallbackHandler here

1.1. ctorCallback

Name	`CallbackHandler::ctorCallback(args: FunctionCallbackInfo<Value>)`
Visibility	public
Parameters	*args: FunctionCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.2. staticCtorCallback

Name	`CallbackHandler::staticCtorCallback(args: FunctionCallbackInfo<Value>)`
Visibility	public
Parameters	*args: FunctionCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.3. memberGetterCallback

Name	`CallbackHandler::memberGetterCallback(property: Local<String>, info: PropertyCallbackInfo<Value>)`
Visibility	public
Parameters	*property: Local<String>, info: PropertyCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.4. memberSetterCallback

Name	`CallbackHandler::memberSetterCallback(property: Local<String>, value: Local<Value>, info: PropertyCallbackInfo<Value>)`
Visibility	public
Parameters	*property: Local<String>, value: Local<Value>, info: PropertyCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.5. memberFunctionCallback

Name	`CallbackHandler::memberFunctionCallback(args: FunctionCallbackInfo<Value>)`
Visibility	public
Parameters	*args: FunctionCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.6. staticGetterCallback

Name	`CallbackHandler::staticGetterCallback(property: Local<String>, info: PropertyCallbackInfo<Value>)`
Visibility	public
Parameters	*property: Local<String>, info: PropertyCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.7. staticSetterCallback

Name	`CallbackHandler::staticSetterCallback(property: Local<String>, value: Local<Value>, info: PropertyCallbackInfo<Value>)`
Visibility	public
Parameters	*property: Local<String>, value: Local<Value>, info: PropertyCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

1.8. staticFunctionCallback

Name	`CallbackHandler::staticFunctionCallback(args: FunctionCallbackInfo<Value>)`
Visibility	public
Parameters	*args: FunctionCallbackInfo<Value>*
Return value	**none**
behavior	describe beahviour

2. NodeHandler

describe class NodeHandler here

2.1. getExports

Name	`NodeHandler::getExports()`
Visibility	public
Parameters	*none*
Return value	**Local**<**Object**> describe return value
behavior	describe beahviour

3. NodeApplication

describe class NodeApplication here

3.1. NodeApplication

Name	`NodeApplication::NodeApplication(acn: char*, argc: int*, argv: char**)`
Visibility	public
Parameters	*acn: char*, argc: int*, argv: char***
Return value	**«constructor»** describe return value
behavior	describe beahviour

4. TemplateFactory

describe class TemplateFactory here

4.1. createTemplate

Name	`TemplateFactory::createTemplate(clazz: TClassRef)`
Visibility	public
Parameters	*clazz: TClassRef*
Return value	**Local**<**FunctionTemplate**> describe return value
behavior	describe beahviour

5. Proxy

describe class Proxy here

5.1. Proxy

Name	`Proxy::Proxy(address: void*, type: TObject, scope: TClassRef)`
Visibility	protected
Parameters	*address: void*, type: TObject, scope: TClassRef*
Return value	**«constructor»** describe return value
behavior	describe beahviour

5.2. setAddress

Name	`Proxy::setAddress(address: void*)`
Visibility	public
Parameters	*address: void**
Return value	**none**
behavior	describe beahviour

5.3. getAddress

Name	`Proxy::getAddress()`
Visibility	public
Parameters	*none*
Return value	**void*** describe return value
behavior	describe beahviour

5.4. getType

Name	`Proxy::getType()`
Visibility	public
Parameters	*none*
Return value	**TObject** describe return value
behavior	describe beahviour

5.5. getScope

Name	`Proxy::getScope()`
Visibility	public
Parameters	*none*
Return value	**TClassRef** describe return value
behavior	describe beahviour

5.6. isGlobal

Name	`Proxy::isGlobal()`
Visibility	public
Parameters	*none*
Return value	**bool** describe return value
behavior	describe beahviour

5.7. isTemplate

Name	`Proxy::isTemplate()`
Visibility	public
Parameters	*none*
Return value	**bool** describe return value
behavior	describe beahviour

5.8. isConst

Name	`Proxy::isConst()`
Visibility	public
Parameters	*none*
Return value	**bool** describe return value
behavior	describe beahviour

5.9. isStatic

Name	`Proxy::isStatic()`
Visibility	public
Parameters	*none*
Return value	**bool** describe return value
behavior	describe beahviour

6. FunctionProxyFactory

describe class FunctionProxyFactory here

6.1. createFunctionProxy

Name	`FunctionProxyFactory::createFunctionProxy(function: TFunction, scope: TClassRef)`
Visibility	public
Parameters	*function: TFunction, scope: TClassRef*
Return value	**ProxyFunciton** describe return value
behavior	describe beahviour

6.2. fromArgs

Name	`FunctionProxyFactory::fromArgs(name: string, scope: TClassRef, args: FunctionCallbackInfo)`
Visibility	public
Parameters	*name: string, scope: TClassRef, args: FunctionCallbackInfo*
Return value	**FunctionProxy** describe return value
behavior	describe beahviour

7. FunctionProxy

Acts as a proxy for a ROOT callable (i.e. function or class method). It provides methods to execute such a callable and validate its arguments. It also maintains a map of `TFunction` - `CallFunc` entries to cache already used functions.

7.1. getCallFunc

Name	`FunctionProxy::getCallFunc(method: TFunction*)`
Visibility	public
Parameters	*method: TFunction**: pointer to the ROOT function for which a proxy is to be created
Return value	**CallFunc*** a pointer to the CallFunc object provied by kling
behavior	gets a pointer to a CallFunc object, which encapsulates the provided TFunction in storage (CallFunc is made available by cling) to which is used during this class' instanciation

7.2. getMethodsFromName

Name	`FunctionProxy::getMethodsFromName(scope: TClassRef, name: string)`
Visibility	public
Parameters	*scope: TClassRef, name: string*
Return value	**vector<TFunction*>** describe return value
behavior	describe beahviour

7.3. FunctionProxy

Name	`FunctionProxy::FunctionProxy(address: void*, function: TFunction, scope: TClassRef)`
Visibility	public
Parameters	*address: void*, function: TFunction, scope: TClassRef*
Return value	**«constructor»** describe return value
behavior	describe beahviour

7.4. getType

Name	`FunctionProxy::getType()`
Visibility	public
Parameters	*none*
Return value	**TFunction** describe return value
behavior	describe beahviour

7.5. validateArgs

Name	`FunctionProxy::validateArgs(args: FunctionCallbackInfo)`
Visibility	public
Parameters	*args: FunctionCallbackInfo*
Return value	**ObjectProxy[]** describe return value
behavior	describe beahviour

7.6. call

Name	`FunctionProxy::call(args: ObjectProxy[])`
Visibility	public
Parameters	*args: ObjectProxy[]*
Return value	**ObjectProxy** describe return value
behavior	describe beahviour

8. ObjectProxyFactory

describe class ObjectProxyFactory here

8.1. createObjectProxy

Name	`ObjectProxyFactory::createObjectProxy(type: TDataMember, scope: TClassRef, holder: ObjectProxy)`
Visibility	public
Parameters	*type: TDataMember, scope: TClassRef, holder: ObjectProxy*
Return value	**ObjectProxy** describe return value
behavior	describe beahviour

9. ObjectProxy

describe class ObjectProxy here

9.1. ObjectProxy

Name	`ObjectProxy::ObjectProxy(type: TDataMember, scope: TClassRef)`
Visibility	public
Parameters	*type: TDataMember, scope: TClassRef*
Return value	**«constructor»** describe return value
behavior	describe beahviour

9.2. getType

Name	`ObjectProxy::getType()`
Visibility	public
Parameters	*none*
Return value	**TDataMember** describe return value
behavior	describe beahviour

9.3. set

Name	`ObjectProxy::set(value: ObjectProxy)`
Visibility	public
Parameters	*value: ObjectProxy*
Return value	**none**
behavior	describe beahviour

9.4. get

Name	`ObjectProxy::get()`
Visibility	public
Parameters	*none*
Return value	**Local**<**Value**> describe return value
behavior	describe beahviour

9.5. setProxy

Name	`ObjectProxy::setProxy(proxy: Local<Object>)`
Visibility	public
Parameters	*proxy: Local<Object>*
Return value	**none**
behavior	describe beahviour

9.6. getProxy

Name	`ObjectProxy::getProxy()`
Visibility	public
Parameters	*none*
Return value	**Local**<**Object**> describe return value
behavior	describe beahviour

9.7. isPrimitive

Name	`ObjectProxy::isPrimitive()`
Visibility	public
Parameters	*none*
Return value	**bool** describe return value
behavior	describe beahviour