

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

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

1.2. staticCtorCallback

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

1.3. memberGetterCallback

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

1.4. memberSetterCallback

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

1.5. memberFunctionCallback

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

1.6. staticGetterCallback

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

1.7. staticSetterCallback

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

1.8. staticFunctionCallback

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

2. NodeHandler

describe class NodeHandler here

2.1. getExports

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

3. NodeApplication

describe class NodeApplication here

3.1. NodeApplication

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

4. TemplateFactory

describe class TemplateFactory here

4.1. createTemplate

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

5. Proxy

describe class Proxy here

5.1. Proxy

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

5.2. setAddress

<i>Name</i>	<code>Proxy::setAddress(address: void*)</code>
<i>Visibility</i>	public
<i>Parameters</i>	<code>address: void*</code>
<i>Return value</i>	none
<i>behavior</i>	describe beahviour

5.3. getAddress

<i>Name</i>	<code>Proxy::getAddress()</code>
<i>Visibility</i>	<code>public</code>
<i>Parameters</i>	<code>none</code>
<i>Return value</i>	void* describe return value
<i>behavior</i>	describe beahviour

5.4. getType

<i>Name</i>	<code>Proxy::getType()</code>
<i>Visibility</i>	public
<i>Parameters</i>	none
<i>Return value</i>	TObject describe return value
<i>behavior</i>	describe beahviour

5.5. getScope

<i>Name</i>	<code>Proxy::getScope()</code>
<i>Visibility</i>	public
<i>Parameters</i>	none
<i>Return value</i>	TClassRef describe return value
<i>behavior</i>	describe beahviour

5.6. isGlobal

<i>Name</i>	<code>Proxy::isGlobal()</code>
<i>Visibility</i>	<code>public</code>
<i>Parameters</i>	<code>none</code>
<i>Return value</i>	<code>bool</code> describe return value
<i>behavior</i>	describe beahviour

5.7. isTemplate

<i>Name</i>	<code>Proxy::isTemplate()</code>
<i>Visibility</i>	<code>public</code>
<i>Parameters</i>	<code>none</code>
<i>Return value</i>	<code>bool</code> describe return value
<i>behavior</i>	describe beahviour

5.8. isConst

<i>Name</i>	<code>Proxy::isConst()</code>
<i>Visibility</i>	<code>public</code>
<i>Parameters</i>	<code>none</code>
<i>Return value</i>	<code>bool</code> describe return value
<i>behavior</i>	describe beahviour

5.9. isStatic

<i>Name</i>	<code>Proxy::isStatic()</code>
<i>Visibility</i>	public
<i>Parameters</i>	none
<i>Return value</i>	bool describe return value
<i>behavior</i>	describe beahviour

6. FunctionProxyFactory

describe class FunctionProxyFactory here

6.1. createFunctionProxy

<i>Name</i>	<code>FunctionProxyFactory::createFunctionProxy(function: TFunction, scope: TClassRef)</code>
<i>Visibility</i>	public
<i>Parameters</i>	<i>function: TFunction, scope: TClassRef</i>
<i>Return value</i>	ProxyFunciton describe return value
<i>behavior</i>	describe beahviour

6.2. fromArgs

<i>Name</i>	<code>FunctionProxyFactory::fromArgs(name: string, scope: TClassRef, args: FunctionCallbackInfo)</code>
<i>Visibility</i>	public
<i>Parameters</i>	<i>name: string, scope: TClassRef, args: FunctionCallbackInfo</i>
<i>Return value</i>	FunctionProxy describe return value
<i>behavior</i>	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

<i>Name</i>	FunctionProxy::getCallFunc(method: TFunction*)
<i>Visibility</i>	public
<i>Parameters</i>	<i>method: TFunction*</i> : pointer to the ROOT function for which a proxy is to be created
<i>Return value</i>	CallFunc* a pointer to the CallFunc object provied by kling
<i>behavior</i>	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' instantiation

7.2. getMethodsFromName

<i>Name</i>	<code>FunctionProxy::getMethodsFromName(scope: TClassRef, name: string)</code>
<i>Visibility</i>	public
<i>Parameters</i>	<p><i>scope: TClassRef</i> a reference to the class which is checked for methods with the specified name</p> <p><i>name: string</i> the name of the overloaded methods which shall be returned</p>
<i>Return value</i>	<code>vector<TFunction*></code> all methods that match the specified name
<i>Behavior</i>	Gets a reference to a class and a method name string. It returns all methods of the class with the specified name. This is needed since JavaScript does not support method overloading.

7.3. FunctionProxy

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

7.4. getType

<i>Name</i>	<code>FunctionProxy::getType()</code>
<i>Visibility</i>	public
<i>Parameters</i>	<i>none</i>
<i>Return value</i>	TFunction describe return value
<i>behavior</i>	describe beahviour

7.5. validateArgs

<i>Name</i>	<code>FunctionProxy::validateArgs(args: FunctionCallbackInfo)</code>
<i>Visibility</i>	public
<i>Parameters</i>	<i>args: FunctionCallbackInfo</i>
<i>Return value</i>	ObjectProxy[] describe return value
<i>behavior</i>	describe beahviour

7.6. call

<i>Name</i>	<code>FunctionProxy::call(args: ObjectProxy[])</code>
<i>Visibility</i>	public
<i>Parameters</i>	<code>args: ObjectProxy[]</code>
<i>Return value</i>	ObjectProxy describe return value
<i>behavior</i>	describe beahviour

8. ObjectProxyFactory

describe class ObjectProxyFactory here

8.1. createObjectProxy

<i>Name</i>	ObjectProxyFactory::createObjectProxy(type: TDataMember, scope: TClassRef, holder: ObjectProxy)
<i>Visibility</i>	public
<i>Parameters</i>	type: TDataMember, scope: TClassRef, holder: ObjectProxy
<i>Return value</i>	ObjectProxy describe return value
<i>behavior</i>	describe beahviour

9. ObjectProxy

describe class ObjectProxy here

9.1. ObjectProxy

<i>Name</i>	ObjectProxy::ObjectProxy(type: TDataMember, scope: TClassRef)
<i>Visibility</i>	public
<i>Parameters</i>	<i>type: TDataMember, scope: TClassRef</i>
<i>Return value</i>	« constructor » describe return value
<i>behavior</i>	describe beahviour

9.2. getType

<i>Name</i>	<code>ObjectProxy::getType()</code>
<i>Visibility</i>	public
<i>Parameters</i>	none
<i>Return value</i>	TDataMember describe return value
<i>behavior</i>	describe beahviour

9.3. set

<i>Name</i>	<code>ObjectProxy::set(value: ObjectProxy)</code>
<i>Visibility</i>	<code>public</code>
<i>Parameters</i>	<i>value: ObjectProxy</i>
<i>Return value</i>	none
<i>behavior</i>	describe beahviour

9.4. get

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

9.5. setProxy

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

9.6. getProxy

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

9.7. isPrimitive

<i>Name</i>	<code>ObjectProxy::isPrimitive()</code>
<i>Visibility</i>	public
<i>Parameters</i>	<i>none</i>
<i>Return value</i>	bool describe return value
<i>behavior</i>	describe beahviour