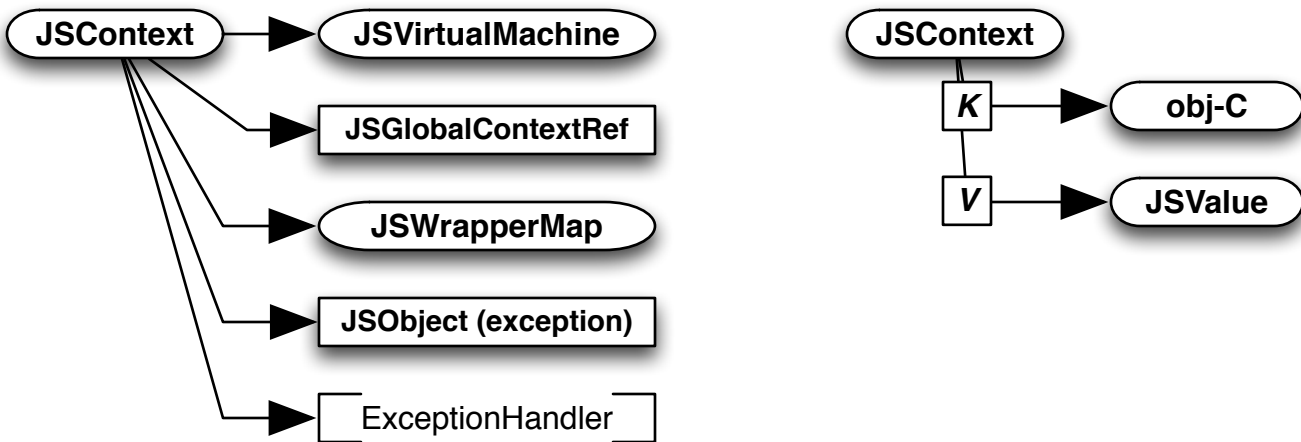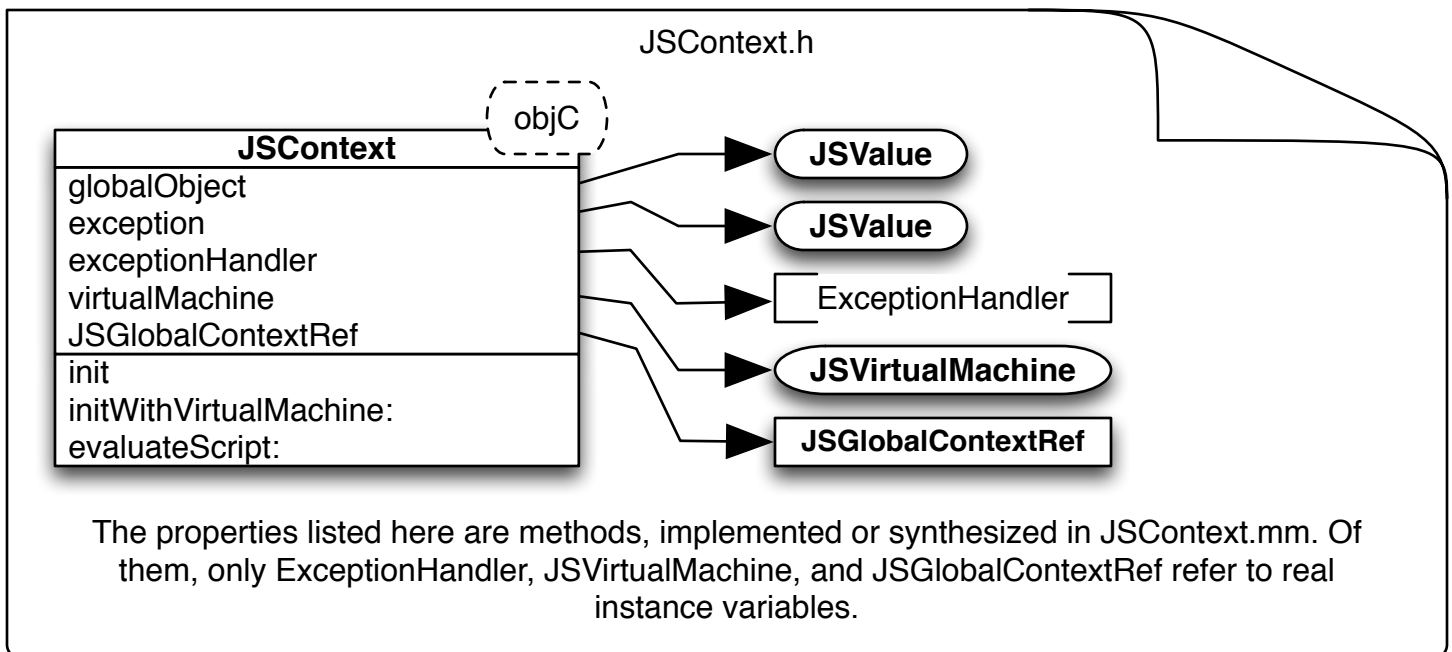# JSContext

This is a wrapper for a JSGlobalContextRef. It also tracks the owning JSVirtualMachine, a JSWrapperMap, and a JS exception. If it is not initialized with an owning JSVirtualMachine, it creates its own.

Usage

As a wrapper for a JS Context, one can execute scripts or fetch the JSValue that represents its global object. Additionally, it will store raised exceptions and accept a block exception handler. If treated as a key/value store, it fetches values from its global object.
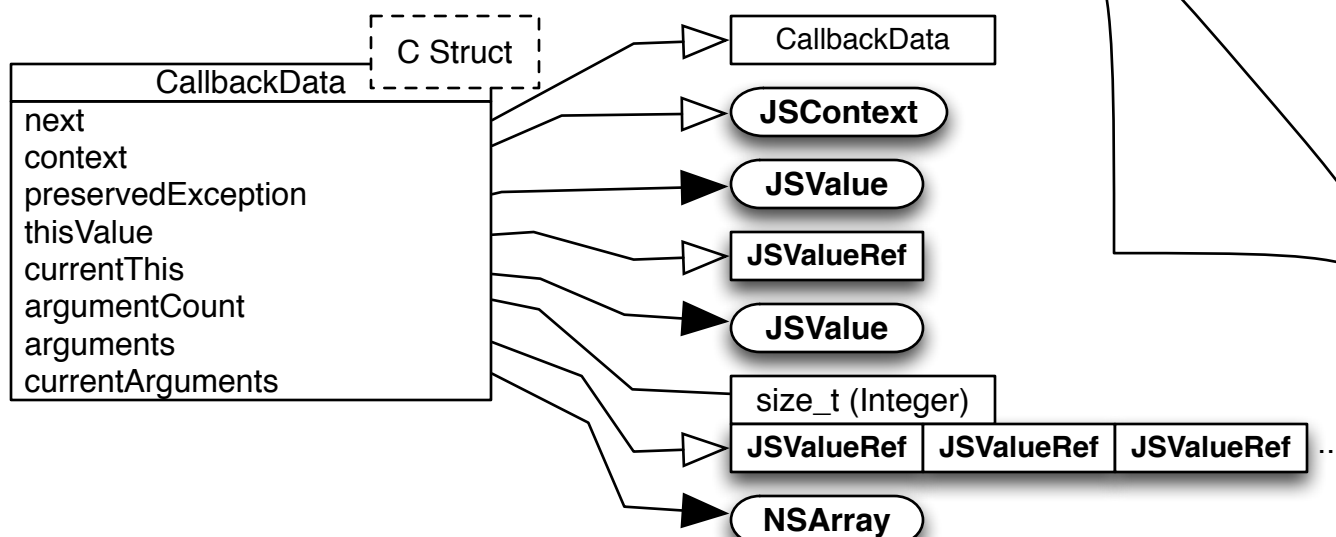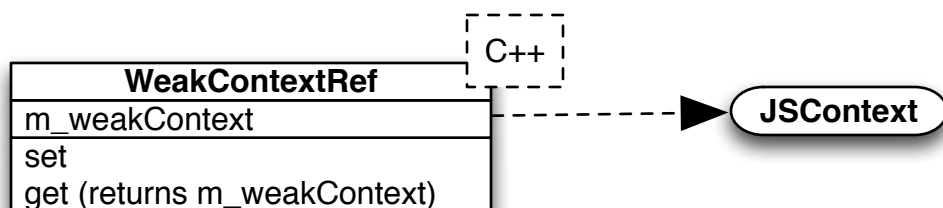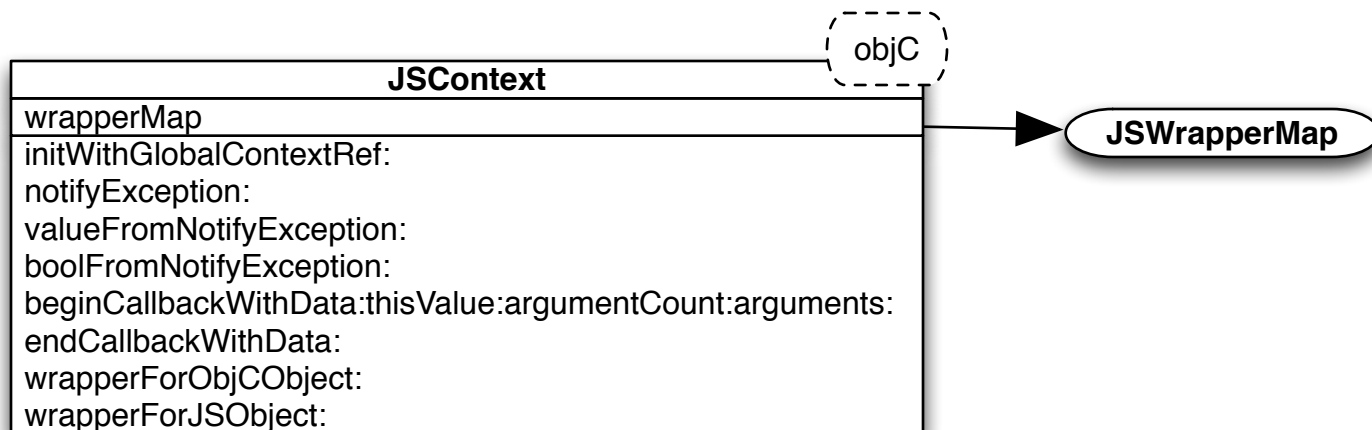
Diagram

The properties listed here are methods, implemented or synthesized in JSContext.mm. Of them, only ExceptionHandler, JSVirtualMachine, and JSGlobalContextRef refer to real instance variables.

JSContext

JSContextInternal.h

C Struct

CallbackData

CallbackData
next
context
preservedException
thisValue
currentThis
argumentCount
arguments
currentArguments

CallbackData → CallbackData

CallbackData → **JSContext**

JSValue

JSValueRef

JSValue

size_t (Integer)

| **JSValueRef** | **JSValueRef** | **JSValueRef** | ... |

NSArray

CallbackData is a temporary C struct that is a node of a linked list representing a call stack. It is used by the JSContext class methods. CurrentThis and currentArguments are generated lazily from the JSCore versions.

C++

WeakContextRef
m_weakContext
set
get (returns m_weakContext)

→ **JSContext**

WeakContextRef is used by ObjCCallbackFunction to provide a weak relationship to an Objective C object in C++.

objC

JSContext
wrapperMap
initWithGlobalContextRef:
notifyException:
valueFromNotifyException:
boolFromNotifyException:
beginCallbackWithData:thisValue:argumentCount:arguments:
endCallbackWithData:
wrapperForObjCObject:
wrapperForJSObject:

→ **JSWrapperMap**

JSContext is further declared here. WrapperMap refers to a real instance variable.

JSContext

objC

JSContext
m_virtualMachine
m_context
m_wrapperMap
m_exception
exceptionHandler
(no additional methods)

JSVirtualMachine

JSGlobalContextRef

JSWrapperMap

JSObject

ExceptionHandler

ExceptionHandler is implicitly created as an instance variable through @synthesize. Methods declared in JSContext.h and JSContextInternal.h are implemented here.

WeakContextRef's creator, destructor, and methods are implemented here, using the functions declared in ObjcRuntimeExtras.h.