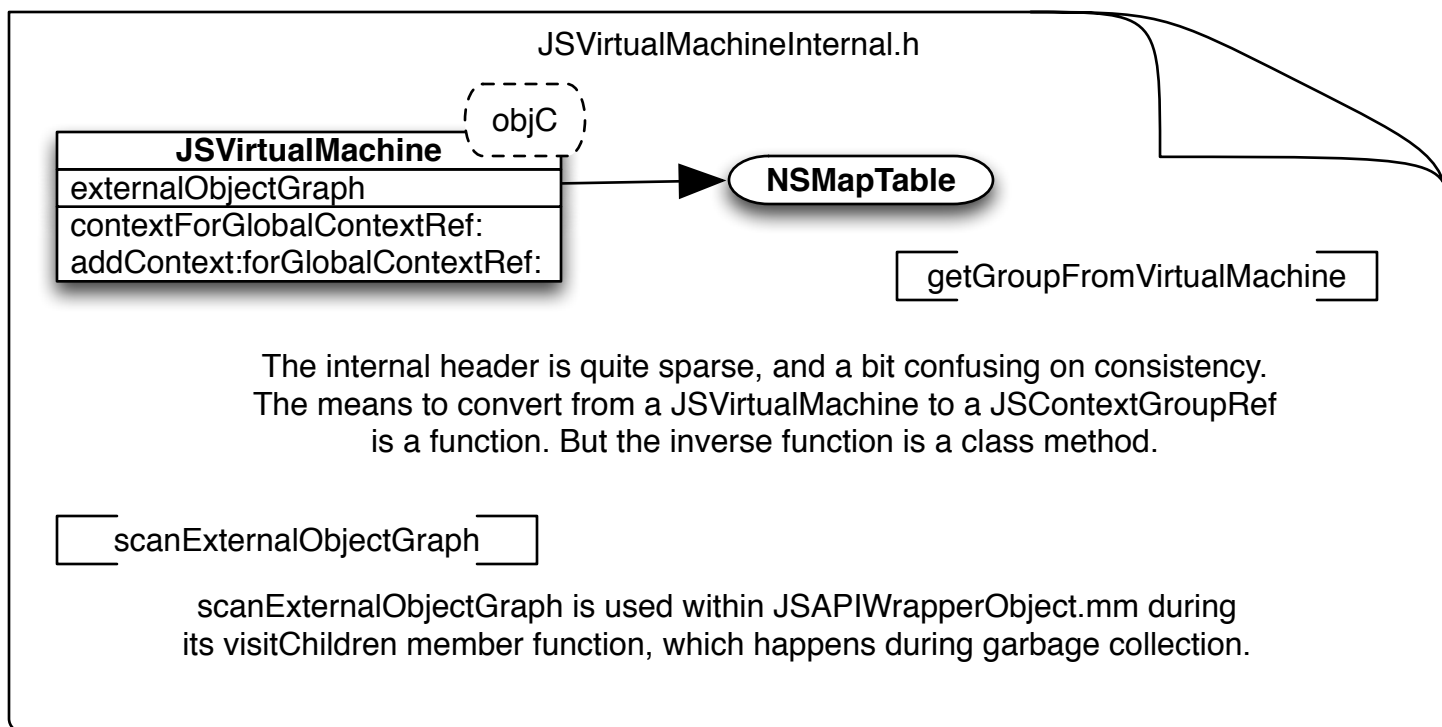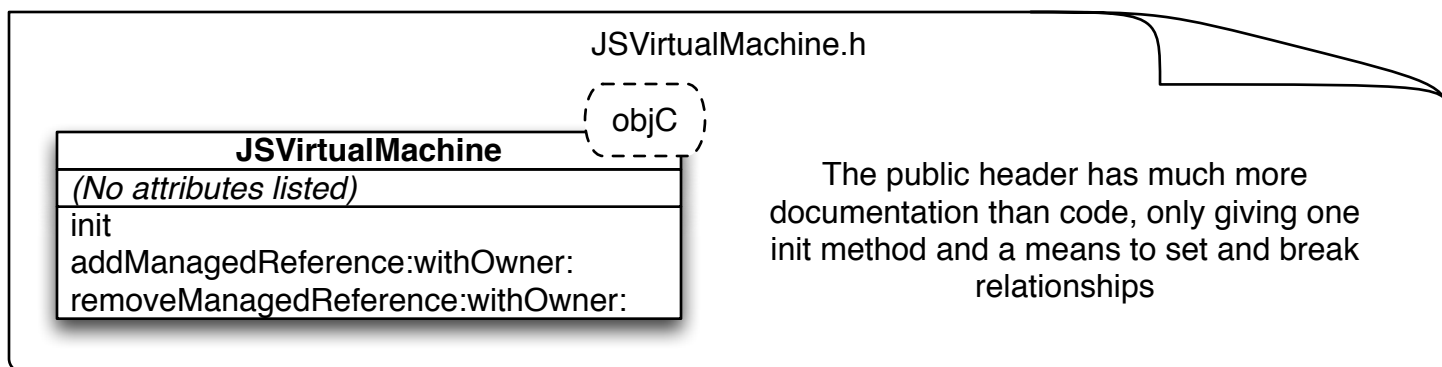# JSVirtualMachine

This is a wrapper for a JSContextGroupRef. It provides a means to explicitly tell the garbage collector of relationships between native objects. Beyond this, however, little is exposed.

Usage

The public API, as mentioned above, only allows for creating and breaking relationships between native objects, but even these relationships are not given a public accessor.

(JSVirtualMachine)

Diagram

JSVirtualMachine.h

(objC)

JSVirtualMachine
(No attributes listed)
init addManagedReference:withOwner: removeManagedReference:withOwner:

The public header has much more documentation than code, only giving one init method and a means to set and break relationships

JSVirtualMachineInternal.h

(objC)

JSVirtualMachine
externalObjectGraph contextForGlobalContextRef: addContext:forGlobalContextRef:

→ (NSMapTable)

getGroupFromVirtualMachine

The internal header is quite sparse, and a bit confusing on consistency. The means to convert from a JSVirtualMachine to a JSContextGroupRef is a function. But the inverse function is a class method.

scanExternalObjectGraph

scanExternalObjectGraph is used within JSAPIWrapperObject.mm during its visitChildren member function, which happens during garbage collection.

JSVirtualMachine

JSVirtualMachine.mm

NSMapTable

K ▷ **JSContextGroupRef**

V ┅▶ **JSVirtualMachine**

initWrapperCache

wrapperCacheLock

wrapperCache

objC

JSVMWrapperCache
(no instance attributes)
(no instance methods)
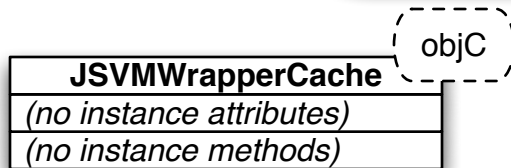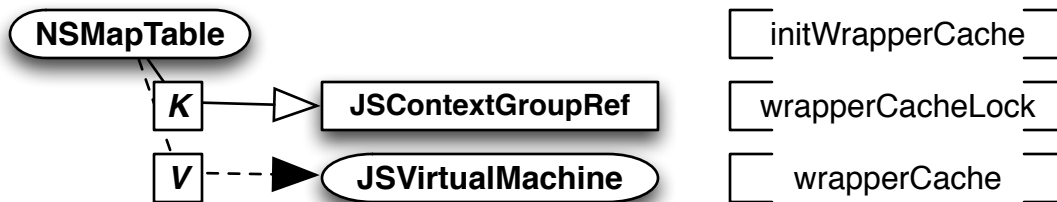
The wrapper cache is an odd setup. An NSMapTable called globalWrapperCache is wrapped by three C functions. Those C functions are only used inside an Objective C class which consists only of two class methods: One to add a pairing, another to get it.

getInternalObjcObject

addManagedReference and removeManagedReference use getInternalObjcObject to extract values from JSValue and JSManagedValue

objC

JSVirtualMachine
m_group
m_contextCache
m_externalObjectGraph
initWithContextGroupRef:

→ **JSContextGroupRef**

→ **NSMapTable**

K ▷ **JSGlobalContextRef**

V ┅▶ **JSContext**

m_group and m_contextCache are straightforward: the first is the wrapping, the second is a reverse lookup table pairing JSGlobalContextRef with its wrapper.

→ **NSMapTable**

m_externalObjectGraph is more complicated. It can be seen as storing the reference relationship count between an owner and a reference, holding both objects weakly.

Owner
K ┅▶ **obj-C**

V ▶ **NSMapTable**

The count is incremented and decremented via addManagedReference and removeManagedReference. During scanExternalObjectGraph, these relationships are traversed to stop garbage collection on reachable objects.

Reference
K ┅▶ **obj-C**

Count
V ── Integer