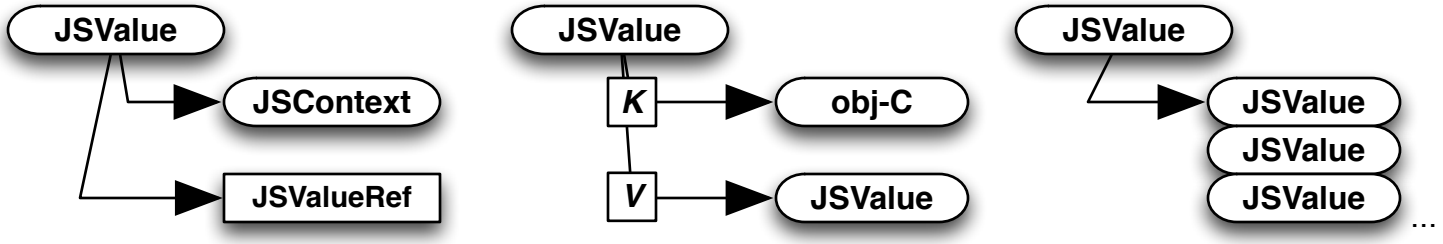
JSValue

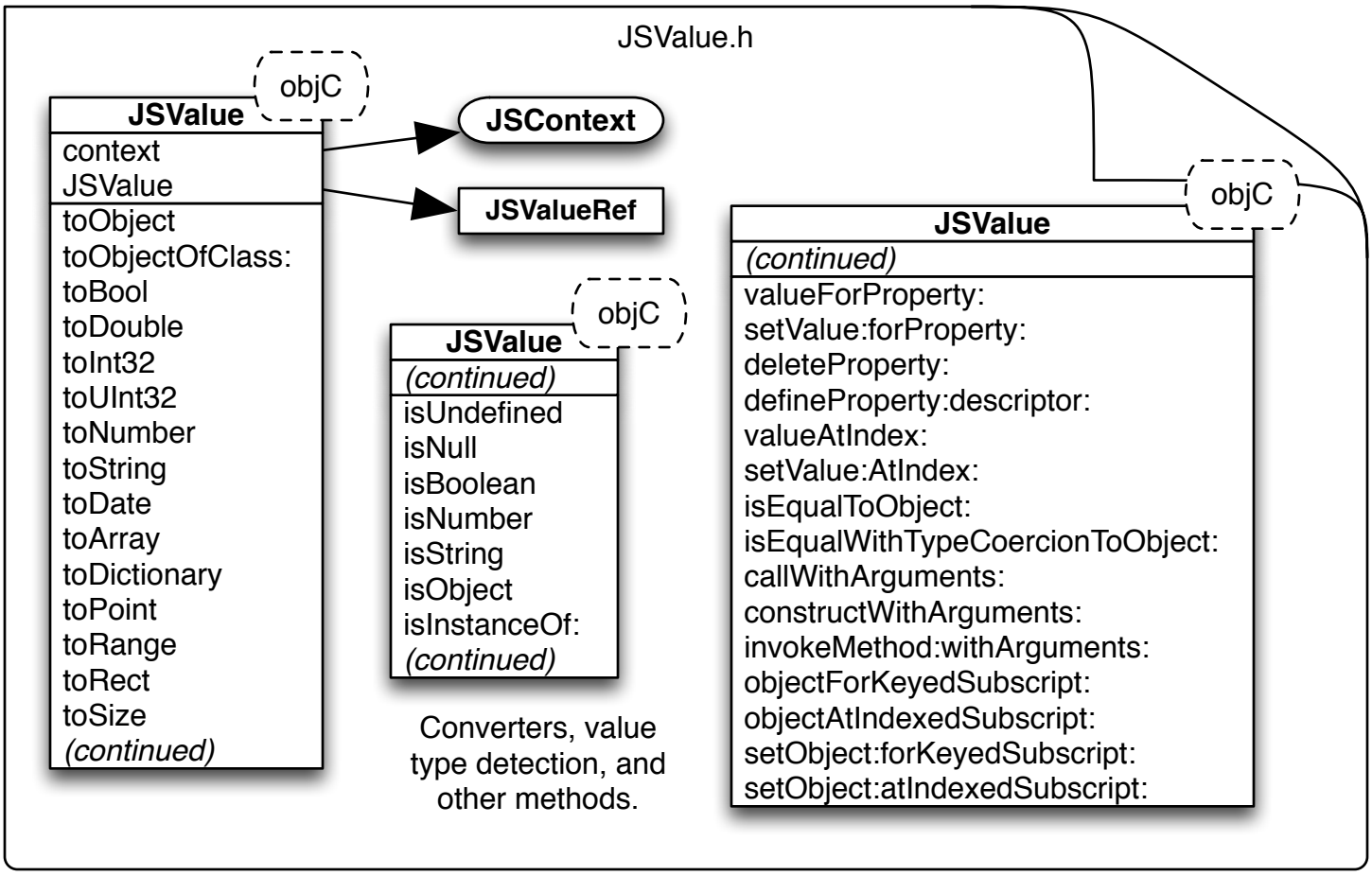
This is a wrapper for any Javascript value, including numeric, strings, booleans, and even those that wrapper native objects. JSValue provides a wealth of accessors and conversion methods, as well as a means by which a developer can easily extend to non-object struct data types.

Usage

As the JSValue.h attests, there are a lot of methods on JSValue, in order to represent the various ways to use the underlying objects. At a base level, it strongly holds said object as well as the context for safety reasons. It has both array and map accessors. Creation is done via class methods.

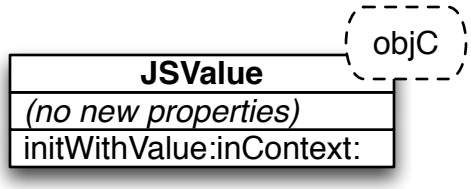


Diagram



JSValue

JSValueInternal.h



valueInternalValue

valueToObject valueToNumber valueToDate

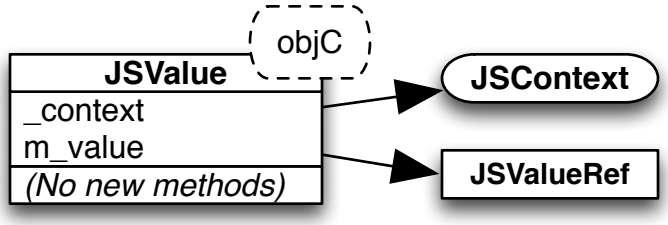
objectToValue valueToString valueToArray valueToDictionary

typeToValueInvocationFor valueToTypeInvocationFor

Beyond two class methods, all that's added to JSValue itself is the internal initializer. Oddly, valueInternalValue is provided as a convenience function with minimal added value.

Due to mappings, valueToObject and objectToValue use the wrapper JSContext to convert between a native object and a JSValueRef, while ValueToNumber, -Date, -String, -Array, and -Dictionary use the JSC JSContextRef as the intermediary. The invocation methods are used by ObjCCallbackFunction.mm to take advantage of any conversion methods added to JSValue.

JSValue.mm



_context is implicitly created, and m_value is the private variable of the JSValue property.

A large portion of JSValue.mm is comprised of JSValue's implementation, many of them trivial wrapper calls to JSC's C api or functions listed in JSValueInternal.h. However, a large portion of the code base is dedicated to converting between JSValueRef and native objects, hinted at by JSValueInternal.

isArray isDate

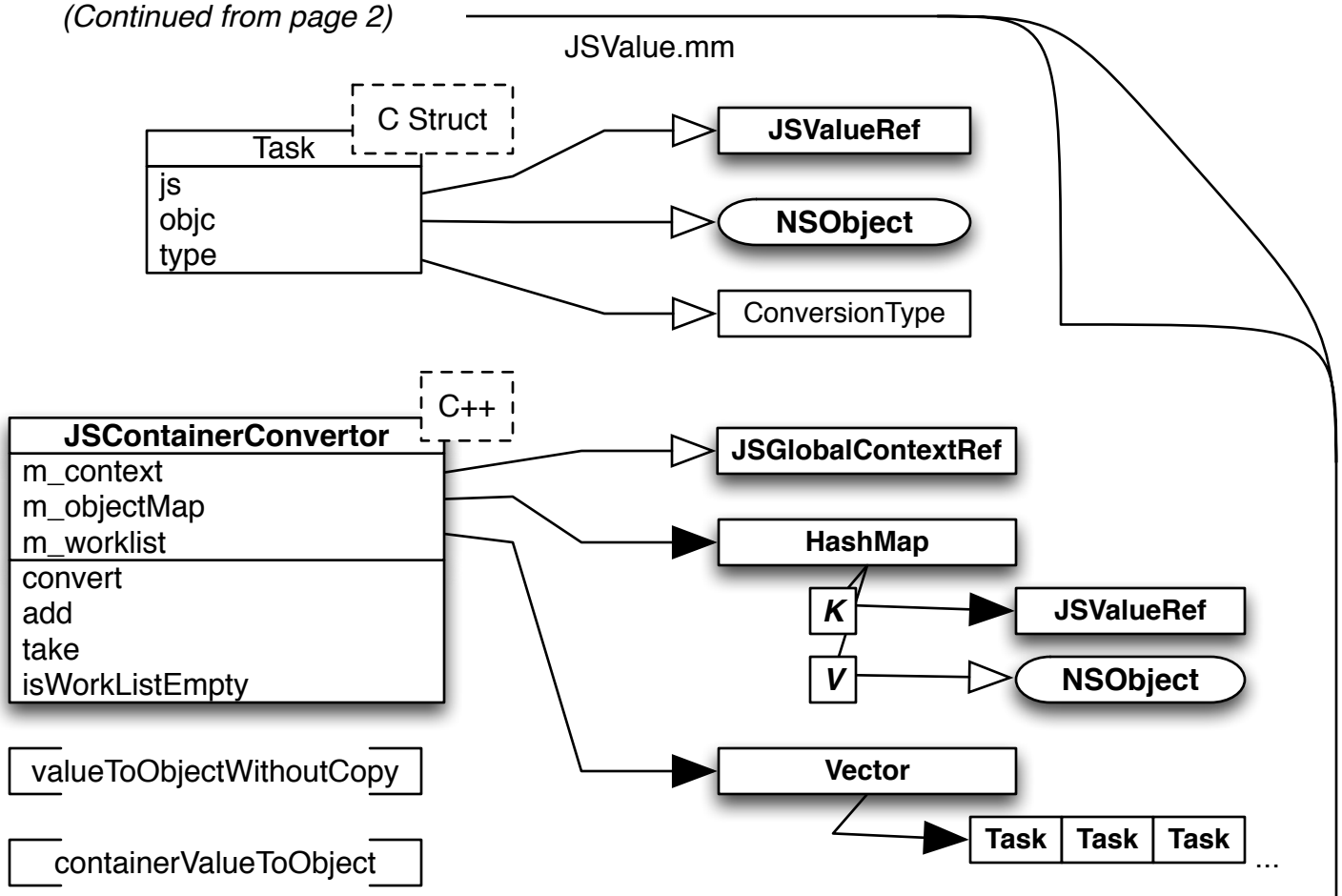
isNSBoolean

isArray, isDate, and isNSBoolean are inline helper functions used to detect datatypes during conversion. The first two are for specific JSValueRef types, the latter for native.

(Continued on page 3)

JSValue

(Continued from page 2)



JSContainerConverter::Task is a simple structure that serves as a tuple between a JSValueRef, native object, and an enum specifying them to be maps, arrays, or neither.

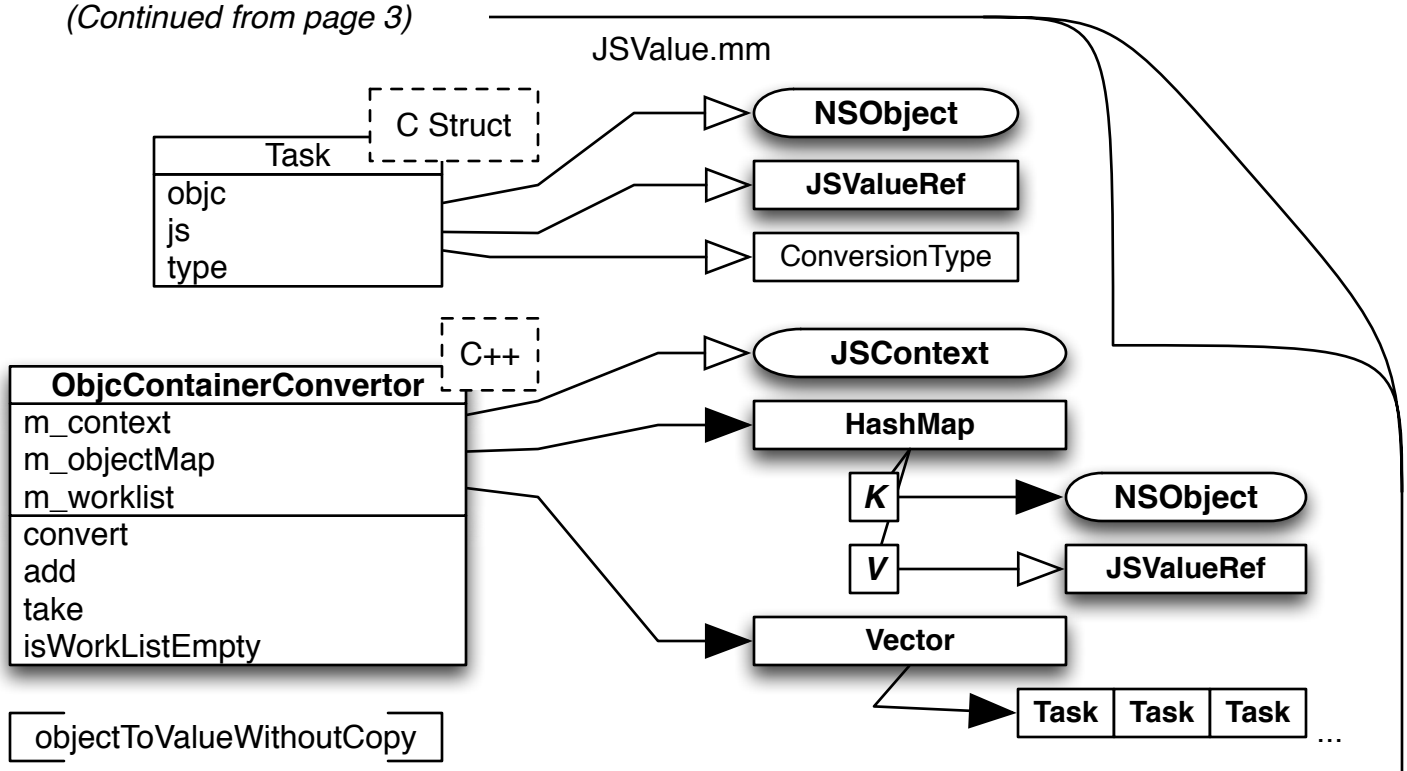
JSContainerConverter itself is an advanced breadth-first JSValueRef to native object converter, compensating for circular ownerships by breaking up container creation to make the empty object before going deeper and having a hash map so that other objects can refer to it before it's finished being built.

Both valueToObjectWithoutCopy and containerValueToObject use this class, those functions themselves used by valueTo____ mentioned in JSValueInternal.h

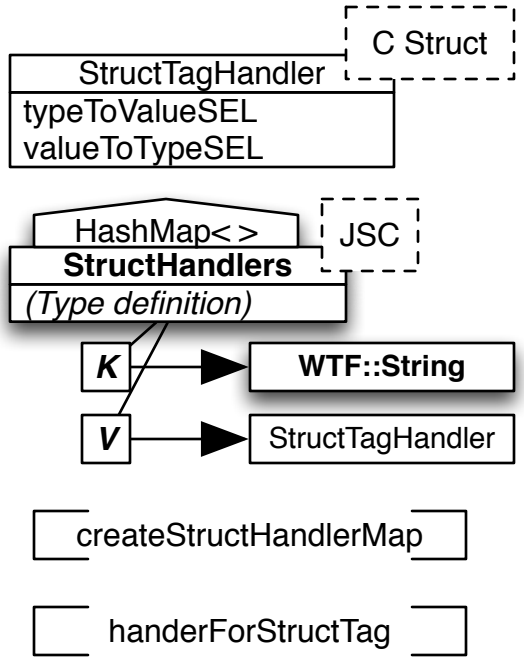
(Continued on page 4)

JSValue

(Continued from page 3)



ObjcContainerConverter and its Task are mirror reflections of JSContainerConverter, and converts from a native object into a JSValueRef. ObjectToValueWithoutCopy similarly reflects the inverse function of valueToObjectWithoutCopy.



Finally, JSValue provides support for translating between data types. This is done by extending the native class (via categories) to create pairs of functions that provide conversion to and from C structure data types. These pairs are stored in the StructHandlers hash map as StructTagHandler values.

This data is generated with createStructHandlerMap, and is stored within handlerForStructTag, which is used by the two functions, typeToValueInvocationFor and valueToTypeInvocationFor listed in JSValueInternals, and used by ObjCCallbackFunction.