WebScripting Protocol Reference

Cocoa > User Experience

2009-04-08

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

WebScripting Protocol Reference 5

Overview 5
Tasks 6
- Getting Attributes 6
- Invoking Methods 6
- Finalizing 6

Class Methods 6
- isKeyExcludedFromWebScript: 6
- isSelectorExcludedFromWebScript: 7
- webScriptNameForKey: 7
- webScriptNameForSelector: 8

Instance Methods 8
- finalizeForWebScript 8
- invokeDefaultMethodWithArguments: 9
- invokeUndefinedMethodFromWebScript:withArguments: 9

Document Revision History 11

Index 13

WebScripting Protocol Reference

(informal protocol)

Framework	/System/Library/Frameworks/WebKit.framework
Availability	Available in Mac OS X v10.3.9 and later.
Companion guide	WebKit Objective-C Programming Guide
Declared in	WebScriptObject.h

Overview

WebScripting is an informal protocol that defines methods that classes can implement to export their interfaces to a WebScript environment such as JavaScript.

Not all properties and methods are exported to JavaScript by default. The object needs to implement the class methods described below to specify the properties and methods to export. Furthermore, a method is not exported if its return type and all its parameters are not Objective-C objects or scalars.

Method argument and return types that are Objective-C objects will be converted to appropriate types for the scripting environment. For example:

- `nil` is converted to undefined.
- `NSNumber` objects will be converted to JavaScript numbers.
- `NSString` objects will be converted to JavaScript strings.
- `NSArray` objects will be mapped to special read-only arrays.
- `NSNull` will be converted to JavaScript's `null`.
- `WebUndefined` will be converted to undefined.
- `WebScriptObject` instances will be unwrapped for the scripting environment.

Instances of all other classes will be wrapped before being passed to the script, and unwrapped as they return to Objective-C. Primitive types such as `int` and `char` are cast to a numeric in JavaScript.

Access to an object's attributes, such as instance variables, is managed by key-value coding (KVC). The KVC methods `setValue:forKey:` and `valueForKey:` are used to access the attributes of an object from the scripting environment. Additionally, the scripting environment can attempt any number of attribute requests or method invocations that are not exported by your class. You can manage these requests by overriding the `setValue:forUndefinedKey:` and `valueForUndefinedKey:` methods from the key-value coding protocol.

Exceptions can be raised from the scripting environment by sending a `throwException:` message to the relevant `WebScriptObject` instance. The method raising the exception must be within the scope of the script invocation.

Tasks

Getting Attributes

`+ webScriptNameForKey:` (page 7)
Returns the scripting environment name for an attribute specified by a key.

`+ webScriptNameForSelector:` (page 8)
Returns the scripting environment name for a selector.

`+ isSelectorExcludedFromWebScript:` (page 7)
Returns whether a selector should be hidden from the scripting environment.

`+ isKeyExcludedFromWebScript:` (page 6)
Returns whether a key should be hidden from the scripting environment.

Invoking Methods

`- invokeDefaultMethodWithArguments:` (page 9)
Executes when a script attempts to invoke a method on an exposed object directly.

`- invokeUndefinedMethodFromWebScript:withArguments:` (page 9)
Handles undefined method invocation from the scripting environment.

Finalizing

`- finalizeForWebScript` (page 8)
Performs cleanup when the scripting environment is reset.

Class Methods

isKeyExcludedFromWebScript:

Returns whether a key should be hidden from the scripting environment.

```
+ (BOOL)isKeyExcludedFromWebScript:(const char *)name
```

Parameters

name
The name of the attribute.

Return Value
`YES` if the attribute specified by `name` should be hidden from the scripting environment; otherwise, `NO`.

Discussion
The default value is `YES`.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

isSelectorExcludedFromWebScript:

Returns whether a selector should be hidden from the scripting environment.

```
+ (BOOL)isSelectorExcludedFromWebScript:(SEL)aSelector
```

Parameters

aSelector
The selector.

Return Value
`YES` if the selector specified by `aSelector` should be hidden from the scripting environment; otherwise, `NO`.

Discussion
Only methods with valid parameters and return types are exported to the WebKit JavaScript environment. The valid types are Objective-C objects and scalars. The default value is `YES`.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

webScriptNameForKey:

Returns the scripting environment name for an attribute specified by a key.

```
+ (NSString *)webScriptNameForKey:(const char *)name
```

Parameters

name
The name of the attribute.

Return Value
The name used to represent the attribute in the scripting environment.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

webScriptNameForSelector:

Returns the scripting environment name for a selector.

```
+ (NSString *)webScriptNameForSelector:(SEL)aSelector
```

Parameters

aSelector

The selector.

Return Value

The name used to represent the selector in the scripting environment.

Discussion

It is your responsibility to ensure that the returned name is unique to the script invoking this method. If this method returns `nil` or you do not implement it, the default name for the selector is constructed as follows:

- A colon (":") in the Objective-C selector is replaced by an underscore ("_").
- An underscore in the Objective-C selector is prefixed with a dollar sign ("$").
- A dollar sign in the Objective-C selector is prefixed with another dollar sign.

The following table shows examples of how the default name is constructed:

Objective-C selector	Default script name for selector
`setFlag:`	`setFlag_`
`setFlag:forKey: withAttributes:`	`setFlag_forKey_withAttributes_`
`propertiesForExample_Object:`	`propertiesForExample$_Object_`
`set_$_forKey: withDictionary:`	`set$_$$_$_forKey_withDictionary_`

Since the default construction for a method name can be confusing depending on its Objective-C name, you should implement this method and return a more human-readable name.

Availability

Available in Mac OS X v10.3.9 and later.

Declared In

`WebScriptObject.h`

Instance Methods

finalizeForWebScript

Performs cleanup when the scripting environment is reset.

```
- (void)finalizeForWebScript
```

Discussion
This method is invoked on objects exposed to the scripting environment just before the scripting environment is reset. After invocation, the receiving object will no longer be referenced by the scripting environment. Further references to `WebScriptObject` instances created by the exposed object will be invalid and may produce unpredictable results.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

invokeDefaultMethodWithArguments:

Executes when a script attempts to invoke a method on an exposed object directly.

```
- (id)invokeDefaultMethodWithArguments:(NSArray *)args
```

Parameters

args
The arguments to be passed to the default method.

Return Value
The result of invoking the default method.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

invokeUndefinedMethodFromWebScript:withArguments:

Handles undefined method invocation from the scripting environment.

```
- (id)invokeUndefinedMethodFromWebScript:(NSString *)name withArguments:(NSArray
    *)args
```

Parameters

name
The name of the undefined method.

args
The arguments passed to the undefined method.

Return Value
The result of invoking the undefined method.

Discussion
This method is invoked when a script attempts to invoke a method not directly exported to the scripting environment. You should return the result of the invocation, converted appropriately for the scripting environment.

Availability
Available in Mac OS X v10.3.9 and later.

Declared In
`WebScriptObject.h`

Document Revision History

This table describes the changes to *WebScripting Protocol Reference*.

Date	Notes
2009-04-08	Added separate parameter and return value descriptions.
2008-06-19	Updated per Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index

F

`finalizeForWebScript` <NSObject> instance method 8

I

`invokeDefaultMethodWithArguments:` protocol instance method 9
`invokeUndefinedMethodFromWebScript:withArguments:` protocol instance method 9
`isKeyExcludedFromWebScript:` protocol class method 6
`isSelectorExcludedFromWebScript:` protocol class method 7

W

`webScriptNameForKey:` protocol class method 7
`webScriptNameForSelector:` protocol class method 8