

ios下JS与OC互相调用（四） -- JavaScriptCore



Haley_Wong (/u/a8cf6d63e889) [+ 关注](#)

2016.08.24 18:16* 字数 2228 阅读 9074 评论 39 喜欢 47 阅读 9074 评论 39 喜欢 47

(/u/a8cf6d63e889)

前面讲完拦截URL的方式实现JS与OC互相调用，终于到JavaScriptCore了。它是从iOS7开始加入的，用 Objective-C 把 WebKit 的 JavaScript 引擎封装了一下，提供了简单快捷的方式与JavaScript交互。

关于JavaScriptCore的使用有两篇很好的文章：

NSHipster中文版的JavaScriptCore (<https://link.jianshu.com?t=http://nshipster.cn/javascriptcore/>)

iOS7 新JavaScriptCore框架入门介绍 (<https://link.jianshu.com?t=http://blog.iderzheng.com/introduction-to-ios7-javascriptcore-framework/>)

看了上述两篇文章，对JavaScriptCore应该已经基本了解了。我就简要介绍一下，然后用代码来实际操作了。先上最终实现的效果：



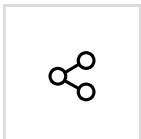
UIWebView JavaScriptCore

效果gif

1、简要介绍JavaScriptCore

JavaScriptCore 是一个iOS 7 新添加的框架，使用前需要先导入 JavaScriptCore.framework 。

然后我们在 JavaScriptCore.h 中可以看到，该框架主要的类就只有五个：



**** 1.1 JSVirtualMachine ****

JSVirtualMachine 看名字直译是JS 虚拟机，也就是说JavaScript是在一个虚拟的环境中执行，而 JSVirtualMachine 为其执行提供底层资源。

翻译这段描述：一个 JSVirtualMachine 实例，代表一个独立的 JavaScript 对象空间，并为其执行提供资源。它通过加锁虚拟机，保证 JSVirtualMachine 是线程安全的，如果要并发执行 JavaScript，那我们必须创建多个独立的 JSVirtualMachine 实例，在不同的实例中执行 JavaScript。

通过 alloc/init 就可以创建一个新的 JSVirtualMachine 对象。但是我们一般不用新建 JSVirtualMachine 对象，因为创建JSContext时，如果我们不提供一个特性的 JSVirtualMachine，内部会自动创建一个 JSVirtualMachine 对象。

**** 1.2 JSContext ****

JSContext 是为JavaScript的执行提供运行环境，所有的JavaScript的执行都必须在 JSContext 环境中。JSContext 也管理 JSVirtualMachine 中对象的生命周期。每一个 JSValue 对象都要强引用关联一个 JSContext。当与某 JSContext 对象关联的所有 JSValue 释放后，JSContext 也会被释放。
创建一个JSContext对象的方式有：

```
// 1.这种方式需要传入一个JSVirtualMachine对象，如果传nil，会导致应用崩溃的。
JSVirtualMachine *JSVM = [[JSVirtualMachine alloc] init];
JSContext *JSCtx = [[JSContext alloc] initWithVirtualMachine:JSVM];

// 2.这种方式，内部会自动创建一个JSVirtualMachine对象，可以通过JSCtx.virtualMachine
// 看其是否创建了一个JSVirtualMachine对象。
JSContext *JSCtx = [[JSContext alloc] init];

// 3. 通过webView的获取JSContext。
JSContext *context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javascriptContext"];
```

上面推荐的两篇文章以及网上介绍JavaScriptCore的文章多是通过1和2这两种方式创建JSContext，然后执行JavaScript，演示JavaScriptCore。我一直有疑问，如果不是HTML结合OC，才会使用到JavaScript，那在一个虚拟的环境里运行JS有什么意义。所以，后面我是用方式3来创建JSContext。

**** 1.3 JSValue ****

JSValue 都是通过 JSContext 返回或者创建的，并没有构造方法。JSValue 包含了每一个 JavaScript类型的值，通过 JSValue 可以将Objective-C中的类型转换为JavaScript中的类型，也可以将JavaScript中的类型转换为Objective-C中的类型。
上述两篇文章中均有OC、JSValue、JavaScript的类型对应关系表。

对应关系

** 1.4 JSManagedValue **

JSManagedValue 主要用途是解决JSValue对象在Objective-C 堆上的安全引用问题。把JSValue 保存进Objective-C 堆对象中是不正确的，这很容易引发循环引用，而导致JSContext不能释放。

这个类主要是将JSValue对象转换为JSManagedValue的API，而且也不常用，就不做具体介绍了。以后遇到使用场景再补充。

** 1.5 JSExport **

JSExport 是一个协议类，但是该协议并没有任何属性和方法。

怎么使用呢？

我们可以自定义一个协议类，继承自 JSExport 。无论我们在 JSExport 里声明的属性，实例方法还是类方法，继承的协议都会*自动的*提供给任何 JavaScript 代码。

So，我们只需要在自定义的协议类中，添加上属性和方法就可以了。

2、代码操作展示

因为该系列主要是JS与OC互调，所以主要介绍如何用JavaScriptCore实现JS与OC互调。

2.1 创建UIWebView，并加载本地HTML。

这步跟 文章（一） (<https://www.jianshu.com/p/7151987f012d>)中的步骤是一样的。

```
self.webView = [[UIWebView alloc] initWithFrame:self.view.frame];
self.webView.delegate = self;
NSURL *htmlURL = [[NSBundle mainBundle] URLForResource:@"index.html" withExtension:nil];
//  NSURL *htmlURL = [NSURL URLWithString:@"http://www.baidu.com"];
NSURLRequest *request = [NSURLRequest requestWithURL:htmlURL];

// 如果不要webView 的回弹效果
self.webView.scrollView.bounces = NO;
// UIWebView 滚动的比较慢，这里设置为正常速度
self.webView.scrollView.decelerationRate = UIScrollViewDecelerationRateNormal
;

[self.webView loadRequest:request];
[self.view addSubview:self.webView];
```

HTML的内容也大致一样，不过JS的调用有些区别，更简单了。

```
function shareClick() {
    share('测试分享的标题','测试分享的内容','url=http://www.baidu.com');
}

function shareResult(channel_id,share_channel,share_url) {
    var content = channel_id+","+share_channel+","+share_url;
    asyncAlert(content);
    document.getElementById("returnValue").value = content;
}

function locationClick() {
    getLocation();
}

function setLocation(location) {
    asyncAlert(location);
    document.getElementById("returnValue").value = location;
}
```

更详细的可以看demo中的HTML源码，demo地址在文章末。

2.2 添加JS要调用的原生OC方法。

在HMTL加载成功的回调方法 - (void)webViewDidFinishLoad:(UIWebView *)webView 中添加要调用的原生OC方法。

```
#pragma mark - UIWebViewDelegate
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
    NSLog(@"webViewDidFinishLoad");

    [self addCustomActions];
}
```

将所有要添加的功能方法，集中到一个方法 addCustomActions 中，便于维护。

```
#pragma mark - private method
- (void)addCustomActions
{
    JSContext *context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javascriptContext"];

    [self addScanWithContext:context];

    [self addLocationWithContext:context];

    [self addSetBGColorWithContext:context];

    [self addShareWithContext:context];

    [self addPayActionWithContext:context];

    [self addShakeActionWithContext:context];

    [self addGoBackWithContext:context];
}
```

然后每一个小功能独立开来，这样修改和解决Bug的时候能够快速定位到某个功能。

```
- (void)addShareWithContext:(JSContext *)context
{
    __weak typeof(self) weakSelf = self;
    context[@"share"] = ^() {
        NSArray *args = [JSContext currentArguments];

        if (args.count < 3) {
            return ;
        }

        NSString *title = [args[0] toString];
        NSString *content = [args[1] toString];
        NSString *url = [args[2] toString];
        // 在这里执行分享的操作...

        // 将分享结果返回给js
        NSString *jsStr = [NSString stringWithFormat:@"shareResult('%@','%@','%@'
)",title,content,url];
        [[JSContext currentContext] evaluateScript:jsStr];
    };
}
```

注意：

- 1.JS要调用的原生OC方法，可以在viewDidLoad webView被创建后就添加好，但最好是在网址加载成功后再添加，以避免无法预料的乱入Bug。
- 2.block 中的执行环境是在子线程中。奇怪的是竟然可以更新部分UI，例如给view设置背景色，调用webView执行js等，但是弹出原生alertView就会在控制台报子线程操作UI的错误信息。
- 3.避免循环引用，因为block 会持有外部变量，而JSContext也会强引用它所有的变量，因此在block中调用self时，要用__weak 转一下。而且在block内不要使用外部的context 以及JSValue，都会导致循环引用。如果要使用context 可以使用 [JSContext currentContext] 。当然我们可以将JSContext 和JSValue当做block的参数传进去，这样就可以使用啦。

2.3 OC调用JS方法

OC调用JS方法就有多种方式了。首先介绍使用JavaScriptCore框架的方式。

**** 方式1 ****

使用JSContext的方法 `-evaluateScript` ，可以实现OC调用JS方法。

下面是一个调用JS中 `payResult` 方法的示例代码：

```
NSString *jsStr = [NSString stringWithFormat:@"payResult('%@')",@"支付成功"];
[[JSContext currentContext] evaluateScript:jsStr];
```

**** 方式2 ****

使用JSValue的方法 `-callWithArguments` ，也可以实现OC调用JS方法。

下面这个示例代码依然是调用JS中的 `payResult`：

```
JSContext *context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javaScriptContext"];

[context[@"payResult"] callWithArguments:@[@"支付弹窗"]];
```

当然，如果是在执行原生OC方法之后，想要在OC执行完操作后，将结果回调给JS时，可以这样写：

```
- (void)addPayActionWithContext:(JSContext*)context
{
    context[@"payAction"] = ^() {
        NSArray *args = [JSContext currentArguments];

        if (args.count < 4) {
            return ;
        }

        NSString *orderNo = [args[0] toString];
        NSString *channel = [args[1] toString];
        long long amount = [[args[2] toNumber] longLongValue];
        NSString *subject = [args[3] toString];

        // 支付操作
        NSLog(@"orderNo:%@---channel:%@---amount:%lld---subject:%@", orderNo, channel, amount, subject);
        // 将支付结果返回给js
        [[JSContext currentContext] @"payResult"] callWithArguments:@[@"支付成功"];
    };
}
```

方式3

以前介绍过的，利用UIWebView的API。

```
NSString *jsStr = [NSString stringWithFormat:@"payResult('%@')",@"支付成功"];
[weakSelf.webView stringByEvaluatingJavaScriptFromString:jsStr];
```

3、补充介绍JavaScriptCore

好处：使用JavaScriptCore，JS调用Native方法时，参数的传递更方便，不用担心特殊符号的转换问题。

不好的地方：只能使用在iOS 7以上。这点我相信现在基本没有多少应用还兼容iOS 6了吧，我去年在做这个功能的时候，还要兼容iOS 6 😭😭。

先把JS与OC互调部分的介绍完了，这里再补充一些关于JavaScriptCore的相关知识。在OC中如何往JS环境中添加一个变量，便于后续在JS中使用呢？

```
JSContext *context = [[JSContext alloc] init];
[context evaluateScript:@"var arr = [3, 4, 'abc'];"];
```

而用到实际的UIWebView上，可以这样：

```
JSContext *context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javascriptContext"];
[context evaluateScript:@"var arr = [3, 4, 'abc'];"];
```

当上面这两行代码执行完后，我点击HTML中的按钮

```
<input type="button" value="输出arr" onclick="showArr()" />
function showArr(){
    asyncAlert(arr);
}

function asyncAlert(content) {
    setTimeout(function(){
        alert(content);
    },1);
}
```

直接输出arr，结果是这样的：

如果我们在OC中想要取出arr，只需要这样：

```
JSValue *value = context[@"arr"];
```

OC中的block可以传入到JavaScript中，这样就创建了一个新的JS方法。我们上面的JS调用OC方法，就是利用的这个实现的。

关于JSExport如何使用？

JSExport 主要是用于将OC中定义的Model类等引入到JavaScript中，便于在JS中使用这种对象和对象的属性、方法。

JSExport的大致使用流程是：

- 1.创建一个自定义协议 XXXExport 继承自 JSExport 。
- 2.在自定义的 XXXExport 中添加JS里需要调用的属性和方法。
- 3.在自定义的Model类中实现XXXExport中的属性的get/set方法以及定义的方法。
- 4.通过JSContext将Model类或者Model类的实例插入到JavaScript中。

当然，我们也可以给已经存在的类动态添加协议，来使其可以供JS 使用。这些示例和示例代码，在文章NSHipster中文版的JavaScriptCore (<https://link.jianshu.com?t=http://nshipster.cn/javascriptcore/>) 和 JavaScriptCore框架在iOS7中的对象交互和管理 (<https://link.jianshu.com?t=http://blog.iderzheng.com/ios7-objects-management-in-javascriptcore-framework/>)中有很详细的介绍和使用展示。

WKWebView 与 JavaScriptCore

关于WKWebView 与 JavaScriptCore，由于WKWebView 不支持通过如下的KVC的方式创建JSContext：

```
JSContext *context = [self.webView valueForKeyPath:@"documentView.webView.mainFrame.javaScriptContext"];
```

那么就不能在WKWebView中使用JavaScriptCore了。

而且，WKWebView中有OC 和JS交互的方式，更easy 、更简洁，因此也用不着使用JavaScriptCore。

WKWebView中如何实现OC与JS交互可以看前面这篇文章： iOS下JS与OC互相调用（三） --MessageHandler (<https://www.jianshu.com/p/433e59c5a9eb>)

UIWebView利用JavaScriptCore来实现交互的示例工程： JS_OC_JavaScriptCore (https://link.jianshu.com?t=https://github.com/Haley-Wong/JS_OC/tree/master/JS_OC_JavaScriptCore)

Have Fun!



Haley_Wong (/u/a8cf6d63e889)

写了 79307 字, 被 2076 人关注, 获得了 1488 个喜欢
(/u/a8cf6d63e889)写了 79307 字, 被 2076 人关注, 获得了 1488 个喜欢

+ 关注

Brighter! Brighter! Brighter!


♡ 喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button)

47



更多分享

(http://cwb.assets.jianshu.io/notes/images/5412956,





下载简书 App ▶

随时随地发现和创作内容



(/apps/download?utm_source=nbc)

被以下专题收入，发现更多相似内容

-  iOS Dev... (/c/c3b8ecefe225?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS Dev... (/c/e84a7722d673?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS网络 (/c/f2a8efbe36c0?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS (/c/5c49d63c8a07?utm_source=desktop&utm_medium=notes-included-collection)
-  [0010]O... (/c/331f76ad0d36?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS开发 (/c/f26a60846b70?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS-swift (/c/31622a1d4959?utm_source=desktop&utm_medium=notes-included-collection)

展开更多 ∨

推荐阅读

更多精彩内容 > (/)

iOS下JS与OC互相调用（六）--WKWebView + WebViewJavascriptBridg...

上一篇文章介绍了UIWebView 如何通过WebViewJavascriptBridge 来实现JS 与OC 的互相调用，这一篇来介绍一下WKWebView 又是如何通过WebViewJavascriptBridge 来实现JS 与OC 的互相调用的。

Haley_Wong (/u/a8cf6d63e889?)

utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

RunLoop 总结：RunLoop的应用场景（二）让Timer...

上一篇讲了使用RunLoop保证子线程的长时间存活，而不是执行完任务后就立刻销毁的应用场景。这一篇就讲述一下RunLoop如何保证NSTimer在视图滑动

(/p/b2d431d6fa09?)

utm_campaign=maleskine&utm_content=note&utm_

为什么你好像什么都知道，但现实中却一事无成？ (/p/86418d688f42?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

你信不信，我问你两个问题，你就会哭。“为什么你起早贪黑加班到凌晨，却依然迷茫彷徨没有出路？”“为什么你报班学习花钱如流水，却依然学无所成平庸无

曲一刀 (/u/ba44df2b6365?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

坚持早起的人打不败，早起让我的2017年赚到了！ (/p/6898c55f6e2a?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

01 我有早起的习惯，每天起来三件事：整理床铺、写晨间日记、晨跑，每天都在坚持。就拿晨跑来说，除非有雾霾或者教练说受伤不能跑休跑外，几乎每天都

晓多 (/u/fee4b4b0b89e?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

焦虑、抑郁、自杀 | 被失眠折磨的90后 (/p/6af7a496d815?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

“ 我已经整整失眠3个月了，从开学到现在，几乎没有一天睡好过，每天都是凌晨2、3点睡着，早上6、7点就醒了，半夜还要醒好几次，我快奔溃了…… ”

黑眼豆豆写字的地方 (/u/a7e0bdc7c757?utm_campaign=maleskine&utm_content=user&utm_medium=pc_all_hots&utm_source=recommendation)

(/p/ac534f508fb0?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

自动化的

安全的

高保真的

深入浅出 JavaScriptCore (/p/ac534f508fb0?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

本文由我们团队的 纠结伦 童鞋撰写。 写在前面 本篇文章是对我一次组内分享的整理，大部分图片都是直接从keynote上截图下来的，本来有很多炫酷动效的，看博客的话就全靠脑补了，多图预警：） 概览

 XcodeMen (/u/d509cc369c78?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/45f8a436aafb?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

自动化的

安全的

高保真的

深入浅出 JavaScriptCore (/p/45f8a436aafb?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

注： 本文copy自http://www.jianshu.com/p/ac534f508fb0，纯属当笔记使用。 概览JavaScriptCore 简介 Objective-C 与 JavaScript 交互JavaScript 与 Objective-C 交互内存管理多线程 ...

 以俊 (/u/70f8cf1634ce?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/3f5dc8042dfc?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

自动化的

安全的

高保真的

深入浅出 JavaScriptCore (/p/3f5dc8042dfc?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

写在前面 本篇文章是对我一次组内分享的整理，大部分图片都是直接从keynote上截图下来的，本来有很多炫酷动效的，看博客的话就全靠脑补了，多图预警：） 概览 JavaScriptCore 简介 Objective-C 与

 我叫纠结伦 (/u/7f46578eb677?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

OC与JS交互之JavaScriptCore (/p/801c977d70d7?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

OC与JS交互之JavaScriptCore 本文摘抄自：https://hjgitbook.gitbooks.io/ios/content/04-technical-research/04-javascriptcore-note.html JavaScriptCore初探...

 大冲哥 (/u/45ad704f731a?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

剖析OC与JS交互 (/p/63a5d1678d6a?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

随着H5技术的兴起，在iOS开发过程中，难免会遇到原生应用需要和H5页面交互的问题。其中会涉及方法调用及参数传值等场景。iOS原生应用和web页面的交互大致上有这几种方法：1) URL拦截协议；（兼容iOS6

 Chris_js (/u/f8c509fc37d0?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

6.15 《公主的风筝》 (/p/b15d26716c2d?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

在古代中国，有一个公主叫小小，在她父皇眼中，她的四个哥哥就像太阳，三个姐姐就像月亮，而小小就像一颗小不点的星星，因此父皇常常忽视她。可是，当皇帝被绑架，哥哥姐姐们全都束手无策时，只有小小一

 我是魏魏 (/u/520b4456d633?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

若如初见 第20章 可是我没有小兔子 (/p/5e08f894b8ee?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

文字：@青丝已绾 林桦睁开眼，首先定了定神，习惯性地回想了一下起床之后的这一天需要做什么，却在第一时间想起昨晚怕苏墨夜里胃痛起来没人照顾，便毅然留在苏墨房里照顾他。回忆了一下，自己这一晚睡得

 喇叭叔叔 (/u/a81a26bb42c7?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

男人现实还是女人现实？ (/p/44797e8d9851?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

最近跟一男性朋友聊天，他说现在的女生太现实了，好像都是喜欢有钱的男生。其实我一直在想到底男人现实还是女人现实？ 当一个男人在一无所有并且需要一个女人的时候，他可以不计较她漂不漂亮，有钱没钱，

 TODAY允儿 (/u/a3c584005fa0?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/eb7639aed934?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

反思——习惯那些事 (/p/eb7639aed934?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

曾经有人问我，实习这么久，你有什么收获？沉思片刻后，我说，成为更好的自己 此人，一直觉得我在敷衍他，然而，事实并非我在敷衍他。以前，我是一个极为不喜欢看书的人。实习后，认识了很多喜欢看书的

 bme_ritter (/u/2db65e841261?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

Back逻辑 (/p/d5133be3935c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

Back逻辑是工作一年左右的小白遇到的一个非常难搞定的问题，尤其是那些需要逛的产品。那么，Back应该按照什么方式去返回？当用户在淘宝类型的移动产品上一层一层一层一层到达某个商品详情页的时候，那么

 枫桥夜泊 (/u/ddbb01048f3c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

