

ios下JS与OC互相调用（五） -- UIWebView + WebViewJavascriptBridge



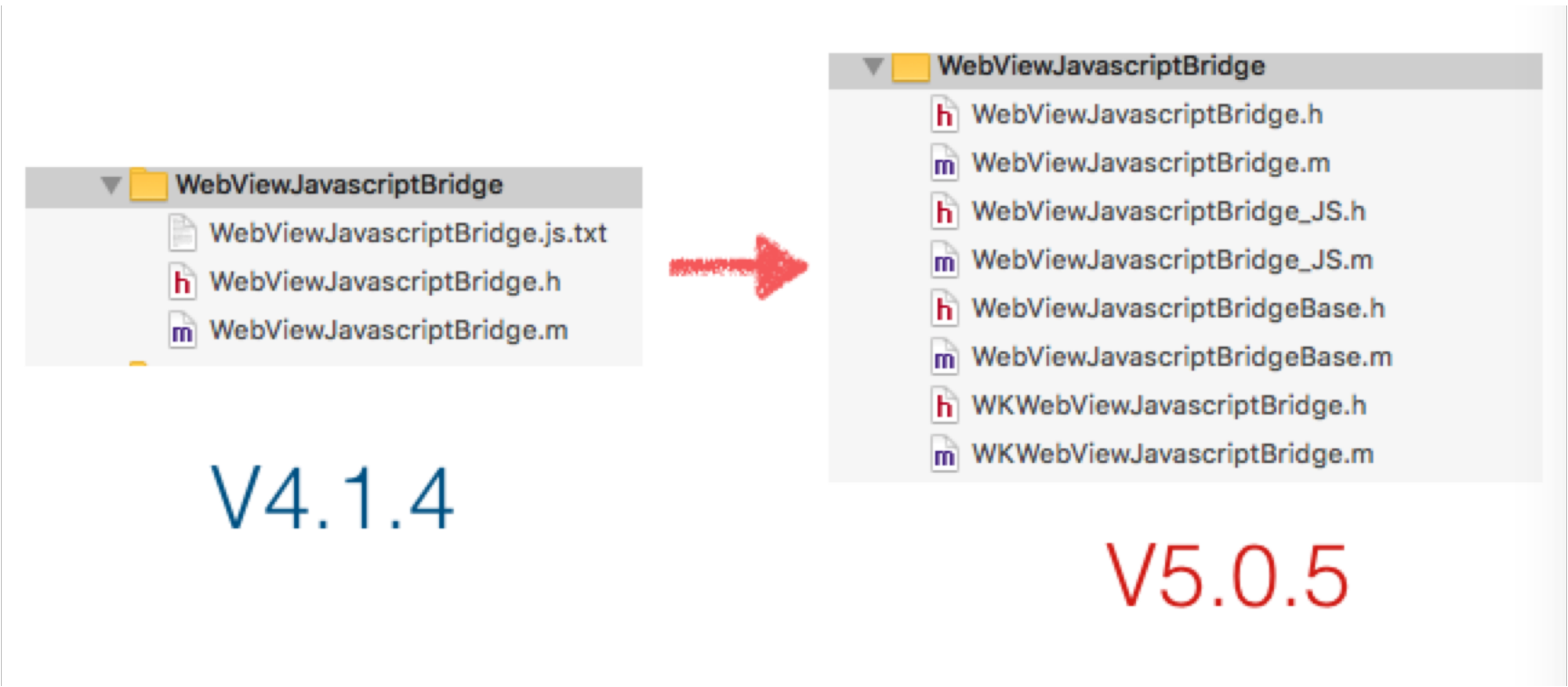
Haley_Wong (/u/a8cf6d63e889) [+ 关注](#)

2016.08.29 11:09* 字数 1962 阅读 5230 评论 16 喜欢 28 阅读 5230 评论 16 喜欢 28

(/u/a8cf6d63e889)

WebViewJavascriptBridge ([https://link.jianshu.com?](https://link.jianshu.com?t=https://github.com/marcuswestin/WebViewJavascriptBridge)

t=<https://github.com/marcuswestin/WebViewJavascriptBridge>)是一个有点年代的JS与OC交互的库，使用该库的著名应用还挺多的，目前这个库有7000+star。我去翻看了它的第一版本已经是4年前了，在版本V4.1.4以及之前，该库只有一个类和一个js的txt文件，所以旧版本的**WebViewJavascriptBridge**是很容易理解的。而最新版的**WebViewJavascriptBridge**因为也要兼容WKWebView，所以里面也加入了两个新的类，一开始看的时候，会被它里面复杂的逻辑吓到，其实仔细阅读后，它还是非常的。

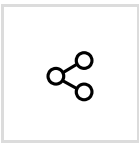


WebViewJavascriptBridge使用讲解

由于WebViewJavascriptBridge在UIWebView和WKWebView下的使用有些许差别，所以就分成两篇文章来讲解WebViewJavascriptBridge。

本文介绍的是在 UIWebView 中使用WebViewJavascriptBridge来达到JS与OC互相调用的目的。WKWebView 使用的是 WKWebViewJavascriptBridge 。接下来，我就从零开始边搭建工程边讲解WebViewJavascriptBridge。由于只研究了两天，可能有讲解的不到位，欢迎批评指正。

先上效果图：



第一步、搭建工程，将WebViewJavascriptBridge库添加到工程中。

新建工程的步骤就略过了，WebViewJavascriptBridge的github地址是：WebViewJavascriptBridge (<https://link.jianshu.com?t=https://github.com/marcuswestin/WebViewJavascriptBridge>)，我使用的是版本V5.0.5。

目前我Demo中的WebViewJavascriptBridge库在最新的iOS系统有崩溃，各位在使用该第三方库时，要先更新到最新版本。

第二步、创建UIWebView和WebViewJavascriptBridge示例。

**2.1 创建UIWebView **

创建UIWebView的代码(viewDidLoad中截选)：

```
self.webView = [[UIWebView alloc] initWithFrame:self.view.frame];
[self.view addSubview:self.webView];

NSURL *htmlURL = [[NSBundle mainBundle] URLForResource:@"index.html" withExtension:nil];
NSURLRequest *request = [NSURLRequest requestWithURL:htmlURL];

// 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];
```

这里不要为UIWebView设置代理，因为在创建WebViewJavascriptBridge的时候，UIWebView的代理已经被赋值给了WebViewJavascriptBridge。

** 2.2 创建WebViewJavascriptBridge **

因为WebViewJavascriptBridge实例，在控制器中多个地方用到，因此最好定义一个property或者实例变量存起来。

```
_webViewBridge = [WebViewJavascriptBridge bridgeForWebView:self.webView];
// {setWebViewDelegate}这个方法，可以将UIWebView的代理，从_webViewBridge中再传递出来。
// 所以如果你要在控制器中实现UIWebView的代理方法时，添加下面这样代码，否则可以不写。
[_webViewBridge setWebViewDelegate:self];
```

然后跟踪进 { - bridgeForWebView } 方法，看它具体是怎么实现的。

```
+ (instancetype)bridgeForWebView:(WVJB_WEBVIEW_TYPE*)webView {
    WebViewJavascriptBridge* bridge = [[self alloc] init];
    [bridge _platformSpecificSetup:webView];
    return bridge;
}

// 上面方法调用了这个方法
- (void) _platformSpecificSetup:(WVJB_WEBVIEW_TYPE*)webView {
    _webView = webView;
    _webView.delegate = self;
    _base = [[WebViewJavascriptBridgeBase alloc] init];
    _base.delegate = self;
}
```

其实 {-bridgeForWebView } 内部也不过是初始化了一个WebViewJavascriptBridge对象, 并为其实例变量_webView 和 _base赋值而已。

第三步、注册js 要调用的Native 功能。

先上示例代码，再分析功能：

```
#pragma mark - private method
- (void)registerNativeFunctions
{
    [self registScanFunction];

    [self registShareFunction];

    [self registLocationFunction];

    [self regitstBGColorFunction];

    [self registPayFunction];

    [self registShakeFunction];
}

- (void)registShareFunction
{
    // 所有JS 需要调用的原生功能都要先用registerHandler注册一下
    [_webViewBridge registerHandler:@"shareClick" handler:^(id data, WVJBResponse
Callback responseCallback) {
        // data 的类型与 JS中传的参数有关
        NSDictionary *tempDic = data;
        // 在这里执行分享的操作
        NSString *title = [tempDic objectForKey:@"title"];
        NSString *content = [tempDic objectForKey:@"content"];
        NSString *url = [tempDic objectForKey:@"url"];

        // 将分享的结果返回到JS中
        NSString *result = [NSString stringWithFormat:@"分享成功:%@,%@,%@",title,c
ontent,url];
        responseCallback(result);
    }];
}
```

- (void)registerHandler:(NSString *)handlerName handler:(WVJBHandler)handler 该方法有两个参数：第一个参数handlerName，是对这个功能起的一个别名；第二个参数handler，是个block，也就是Native 实现的功能。JS要调用的Native 实现其实就是block 的 {} 内的代码功能。为了便于维护，我们可以将JS要调用的Native方法都集中到一起，然后单个功能再封装一个方法。

第四步、完成HMTL必要的JS代码

由于WebViewJavascriptBridge也是拦截URL来实现的调用原生功能，所以有一些代码跟之前iOS下JS与OC互相调用（一）(<https://www.jianshu.com/p/7151987f012d>)中的HTML JS代码很相似。

HTML 中有一个必须要添加的JS 方法，然后需要自动调用一次该方法。该方法是：

```
function setupWebViewJavascriptBridge(callback) {
    if (window.WebViewJavascriptBridge) { return callback(WebViewJavascriptBridge
); }
    if (window.WVJBCallbacks) { return window.WVJBCallbacks.push(callback); }
    window.WVJBCallbacks = [callback];
    var WVJBIframe = document.createElement('iframe');
    WVJBIframe.style.display = 'none';
    WVJBIframe.src = 'wjbscheme://__BRIDGE_LOADED__';
    document.documentElement.appendChild(WVJBIframe);
    setTimeout(function() { document.documentElement.removeChild(WVJBIframe) }, 0
)
}
```

上面这个方法的参数是一个function，这个方法的作用主要是在第一次加载HTML的时候起作用，目的是加载一次 wjbscheme://__BRIDGE_LOADED__，来触发往HTML中注入一些已经写好的JS方法。

看看上面的方法与下面这个是不是很相似：

```
function loadURL(url) {
    var iFrame;
    iFrame = document.createElement("iframe");
    iFrame.setAttribute("src", url);
    iFrame.setAttribute("style", "display:none;");
    iFrame.setAttribute("height", "0px");
    iFrame.setAttribute("width", "0px");
    iFrame.setAttribute("frameborder", "0");
    document.body.appendChild(iFrame);
    // 发起请求后这个iFrame就没用了，所以把它从dom上移除掉
    iFrame.parentNode.removeChild(iFrame);
    iFrame = null;
}
```

添加完 setupWebViewJavascriptBridge 方法，需要在JS中主动调用一次该方法：

```
setupWebViewJavascriptBridge(function(bridge) {
    bridge.registerHandler('testJavascriptHandler', function(data, responseCallb
ack) {
        alert('JS方法被调用:'+data);
        responseCallback('js执行过了');
    })
})
```

Native 需要调用的 JS 功能，也是需要先注册，然后再执行的。如果Native 需要调用的 JS 功能有多个，那么这些功能都要在这里先注册，注册之后才能够被Native 调用。接下来需要好好分析一下JS 中这个方法的作用了。

这里是重点：

1、首先调用 `setupWebViewJavascriptBridge`，第一次执行的时候，由于 `window.WebViewJavascriptBridge` 和 `window.WVJBCallbacks` 都不存在，所以会继续往下执行，将参数callback（它是一个function）装进数组赋值给 `window.WVJBCallbacks`。

js 支持动态添加属性 并赋值，这里 `window.WVJBCallbacks = [callback]`；就是动态添加属性，并赋值。另外js中的全局变量都可以使用`window.xxxx`来调用;动态添加的属性也可以不加 `window.`，直接使用。

2、`WebViewJavascriptBridge` 帮助JS调用Native的url 有两种，一种是 `wvjbscheme://__BRIDGE_LOADED__`；而另一种是 `wvjbscheme://__WVJB_QUEUE_MESSAGE__`。前者只有在调用 `setupWebViewJavascriptBridge` 的时候执行一次，一般来说这个url 如果没有页面应该只会执行一次。第二种url所有js调用Native 功能时，都会使用到。

3、在拦截到自定义的url 时，`WebViewJavascriptBridge` 分了三情况，如果是 `wvjbscheme://__BRIDGE_LOADED__` ,就往HMTL 中注入已经写好的js，这个js 在 `WebViewJavascriptBridge_JS` 中；如果是 `wvjbscheme://__WVJB_QUEUE_MESSAGE__`，那就利用 `stringByEvaluatingJavaScriptFromString`，取回调用js中callHandler传进去的参数。

然后再从 `WebViewJavascriptBridge` 之前保存的Native 方法对应的block，调用对应的block。

第五步、调用Native 功能

利用之前注入的JS方法callHandler 就可以调用Native 功能了。
示例代码如下：

```
function shareClick() {
    var params = {'title':'测试分享的标题','content':'测试分享的内容','url':'http://www.baidu.com'};
    WebViewJavascriptBridge.callHandler('shareClick',params,function(response) {
        alert(response);
        document.getElementById("returnValue").value = response;
    });
}
```

这里callHandler前的 `WebViewJavascriptBridge` ,其实就是上一步注入到JS中的代码中，动态创建属性，动态赋值的属性。如下代码片段可以在 `WebViewJavascriptBridge_JS` 中找到。

```
window.WebViewJavascriptBridge = {
    registerHandler: registerHandler,
    callHandler: callHandler,
    disableJavascriptAlertBoxSafetyTimeout: disableJavascriptAlertBoxSafetyTimeout,
    _fetchQueue: _fetchQueue,
    _handleMessageFromObjC: _handleMessageFromObjC
};
```

而callHandler 内部调用了另一个js function `_doSend` ,而 `_doSend` 内部其实，就是把 handlerName和 参数data，再加上callbackId 装成键值对，然后保存到数组 `sendMessageQueue`，同时加载一次 `wvjbscheme://__WVJB_QUEUE_MESSAGE__`。

到此 利用 `WebViewJavascriptBridge` 实现JS 调用iOS Native 就完成了。

第六步、Native 调用 js 功能

Native 调用js 的功能，也需要先在js 中为要调用的功能注册一个别名。
** 6.1 js 注册Native 要调用的功能 **
示例代码：

```

setupWebViewJavascriptBridge(function(bridge) {
    bridge.registerHandler('testJSFunction', function(data, responseCallback) {
        alert('JS方法被调用:'+data);
        responseCallback('js执行过了');
    })
    // 注册其他的功能
    //bridge.regsiterHandler.....
})

```

**** 6.2 Native 调用功能的别名handlerName ****

示例代码：

```

//      // 如果不需要参数，不需要回调，使用这个
//      [_webViewBridge callHandler:@"testJSFunction"];
//      // 如果需要参数，不需要回调，使用这个
//      [_webViewBridge callHandler:@"testJSFunction" data:@"一个字符串"];
//      // 如果既需要参数，又需要回调，使用这个
//      [_webViewBridge callHandler:@"testJSFunction" data:@"一个字符串" responseCallba
ck:^(id responseData) {
    NSLog(@"调用完JS后的回调: %@", responseData);
}];

```

而callHandler 方法又是如何实现调用js 方法的呢？

callHandler 内部是将传递给js 的参数、handlerName、callbackId组合成字典，然后把字典转换成字符串，将转换后的字符串以参数的形式，通过

stringByEvaluatingJavaScriptFromString 传递给js ，js 中将传递过来的字符串转成json ，然后通过handlerName 获取对应的function执行。

关键的几个代码段：

```

// 这里是Native 调用js ，把参数转换为字符串，执行js 中的_handleMessageFromObjC方法。
- (void)_dispatchMessage:(WVJBMessage*)message {
    NSString *messageJSON = [self _serializeMessage:message pretty:NO];
    [self _log:@"SEND" json:messageJSON];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\"" withString:@"\"\\\""];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\"" withString:@"\"\\\""];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\"" withString:@"\"\\\""];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\\n" withString:@"\\n"];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\\r" withString:@"\\r"];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\\f" withString:@"\\f"];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\\u2028" withhString:@"\\u2028"];
    messageJSON = [messageJSON stringByReplacingOccurrencesOfString:@"\\u2029" withhString:@"\\u2029"];

    NSString* javascriptCommand = [NSString stringWithFormat:@"WebViewJavascriptB
ridge._handleMessageFromObjC('%@');", messageJSON];
    if ([NSThread currentThread] isMainThread) {
        [self _evaluateJavascript:javascriptCommand];
    } else {
        dispatch_sync(dispatch_get_main_queue(), ^{
            [self _evaluateJavascript:javascriptCommand];
        });
    }
}

```

下面这里是找到handlerName 对应的function，并执行function。

```

var handler = messageHandlers[message.handlerName];
if (!handler) {
    console.log("WebViewJavascriptBridge: WARNING: no handler for message from 0
bjC:", message);
} else {
    handler(message.data, responseCallback);
}

```


到这里 利用WebViewJavascriptBridge 实现Native 调用js 的功能就完成了。

提醒：

JS 有动态参数的特性，调用js 的方法，可以传0个参数，1个参数，N个参数都可以。

例如，我们在js中定义一个test()方法，我们可以调用test()，来执行这个方法；如果有参数要传进来，也可以调用test(xxx)；如果有多个参数，那么就用test(xxx,xxx)。当然如果我们定义的参数是test(a,b,c)，也可以少传参数，或者不传参数调用test()。

总结

利用WebViewJavascriptBridge来实现JS与OC的交互的优点：

1、获取参数时，更方便一些，如果参数中有一些特殊符号或者url带参数，能够很好的解析。

也有一些缺点：

- 1、做一次交互，需要执行的js 与原生的交互步骤较多，至少有两次。
- 2、需要花较多的时间，理解WebViewJavascriptBridge的原理和使用步骤。

示例工程地址：JS_OC_WebViewJavascriptBridge (https://link.jianshu.com?t=https://github.com/Haley-Wong/JS_OC/tree/master/JS_OC_WebViewJavascriptBridge)

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)

| 28







更多分享

(<http://cwb.assets.jianshu.io/notes/images/5467258>




下载简书 App ▶

随时随地发现和创作内容




(/apps/download?utm_source=nbc)

-  iOS网络 (/c/f2a8efbe36c0?utm_source=desktop&utm_medium=notes-included-collection)
-  H5-OC相互调用 (/c/2c6aeff86649?utm_source=desktop&utm_medium=notes-included-collection)
-  [0010]O... (/c/331f76ad0d36?utm_source=desktop&utm_medium=notes-included-collection)
-  JS和OC的交互 (/c/d25a473ec7ed?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS干货 (/c/1da28812a8a1?utm_source=desktop&utm_medium=notes-included-collection)
-  ios专题 (/c/9f09a5caa1f6?utm_source=desktop&utm_medium=notes-included-collection)
-  开开心心写代码 (/c/7d42d90b5b8d?utm_source=desktop&utm_medium=notes-included-collection)

展开更多 ∨

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=seo_notes&utm_source=recommendation)

(/p/6f34903be630?

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

[iOS 开发] WebViewJavascriptBridge 从原理到实战 (/p/6f34903be630...

前言：iOS 开发中，h5 和原生实现通信有多种方式，JSBridge 就是最常用的一种，各 JSBridge 类库的实现原理大同小异，这篇文章主要是针对当前使用最为广泛的 WebViewJavascriptBridge（v6.0.2），从功能

 祥龙Shannon (/u/4ef5e287fc91?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

Objective-c与js交互专题 (/p/e97a357d0688?utm_campaign=maleskin...

1 原理 在写 JavaScript 的时候，可以使用一个叫做 window 的对象，像是我们想要从现在的网页跳到另外一个网页的时候，就会去修改 window.location.href 的位置；在我们的 Objective-C 程序码中，如果我们可以

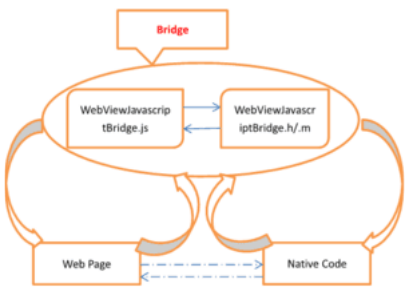
 zhengkexing (/u/edde41976524?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

iOS下JS与OC互相调用（一）--UIWebView 拦截URL (/p/7151987f012d?...

最近准备把之前用UIWebView实现的JS与原生相互调用功能，用WKWebView来替换。顺便搜索整理了一下JS 与OC 交互的方式，非常之多啊。目前我已知的JS 与 OC 交互的处理方式： 1.在JS 中做一次URL跳转，


 Haley_Wong (/u/a8cf6d63e889?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/1b4bd58b4724?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
优秀开源代码解读之JS与iOS Native Code互调的优雅实现方案 (/p/1b4bd5...


本篇为大家介绍一个优秀的开源小项目：WebViewJavascriptBridge。 它优雅地实现了在使用UIWebView时JS与ios 的ObjC nativecode之间的互调，支持消息发送、接收、消息处理器的注册与调用以及设置消息处理

 三爷、泥人的夜 (/u/6191d236b912?

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

奇世俗人 (/p/80d191d4d5c2?utm_campaign=maleskine&utm_content...


-----一个被困者悠游自在的一生 我做梦也没想到，国师居然是这样的国师。 他会写非常美妙的文字，基本上每天一篇。比如这个，他昨天写的，“白日依山尽，黄河入海流，欲穷千里目，更上一层楼

 啦啦啦啦啦啦啦啦 (/u/0fe748500ffd?

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

在线靶场 (/p/4a15e943bcf7?utm_campaign=maleskine&utm_content=...

1.DVWA DVWA是著名的OWASP开放出来的一个在线web安全教、学平台。提供了：暴力破解、命令执行、CSRF、文件包含、SQL注入、XSS学习环境，并且分： low、medium、high三种不同的安全等级，等级越

 彭向上 (/u/2be639f50f02?

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

(/p/8cea085c8de8?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
黛玉晚报170311——《江南皮革厂》 (/p/8cea085c8de8?utm_campaign...


奇思妙想·江南皮革厂 作者：林陌鹿 1初六日，雾霾。南方的雾霾不似北方，因为沙尘的关系，北方暗黄，南方谲白。很多年以前，我在一片茫然中，出现在江南皮革厂。点击阅读全文 青春·如果我们当初不添加好友，

 简黛玉 (/u/d9edcb44e2f2?

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

告别 (/p/d8ee4781ed17?utm_campaign=maleskine&utm_content=not...

行走在午夜的街头，夏日的凉风撩拨着她耳边的发丝，发丝钻进了脖子里，酥酥痒痒的。身边走着的是有些微醺的他，有淡淡的酒味儿混杂着他身上淡淡的烟草味儿被这夏夜的风送进她的嗅觉里。 午夜的街头，没有

 铃兰ft (/u/d000bfe2a1f4?

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

内存优化的建议和技巧 (/p/9761050a3ce6?utm_campaign=maleskine&...

转载自:http://www.cnblogs.com/CoderAlex/p/5264070.html性能对 iOS 应用的开发尤其重要，如果你的应用失去反应或者很慢，失望的用户会把他们的失望写满App Store的评论。然而由于iOS设备的限制，有时

 liuning_leo (/u/dadcf7a60b30?

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