JavaScriptCore 实现 JS 交互

1. 简介：

   JavaScripCore 是 webKit 的一个重要组成部分，主要对 JS 进行解析和提供执行环境。iOS7 以前对 JS 的操作只有 webView 里函数 stringByEvaluatingJavaScriptFromString，JS 对 OC 调用基本基于 URL 的拦截操作。常用的是 webViewJavaScriptBridge 和 EasyJS、webView 两个开源库。iOS7 推出了 JavaScripCore，极大方便了我们对 JS 的交互。

2. JS 调用原生 OC

   方式一：JS 发起一个假请求，然后利用 UIWebView 的代理方法拦截这次请求，然后做相应处理。firstClick://

```
-webView:shouldStartLoadWithRequest:{
NSURL *url = [request URL];
if([URL scheme] isEqualToString:@”firstClick”)){
   //取出 URL 参数(用&分隔)[url.query componentsSepar...];
   return NO;//表示不要加载这个 url
}
}
```

   方式二：采用 JavaScripCore 实现

```
#import <UIKit/UIKit.h>
#import <JavaScriptCore/JavaScriptCore.h>

@protocol TestJSExport <JSExport>
JSExportAs
(calculateForJS  /** handleFactorialCalculateWithNumber 作为js方法的别名 */,
 - (void)handleFactorialCalculateWithNumber:(NSNumber *)number
 );
- (void)pushViewController:(NSString *)view title:(NSString *)title;
@end



@interface JSCallOCViewController : UIViewController<UIWebViewDelegate,TestJSExport>

@property (weak, nonatomic) IBOutlet UIWebView *webView;
@property (strong, nonatomic) JSContext *context;

@end
```

```
- (void)webViewDidFinishLoad:(UIWebView *)webView
{
    // 以 html title 设置 导航栏 title
    self.title = [webView stringByEvaluatingJavaScriptFromString:@"document.title"];

    // 禁用 页面元素选择
    //[webView stringByEvaluatingJavaScriptFromString:@"document.documentElement.style.webkitUserSelect='none';"];

    // 禁用 长按弹出ActionSheet
    //[webView stringByEvaluatingJavaScriptFromString:@"document.documentElement.style.webkitTouchCallout='none';"];

    // Undocumented access to UIWebView's JSContext
    self.context = [webView valueForKeyPath:@"documentView.webView.mainFrame.javaScriptContext"];

    // 打印异常
    self.context.exceptionHandler =
    ^(JSContext *context, JSValue *exceptionValue)
    {
        context.exception = exceptionValue;
        NSLog(@"%@", exceptionValue);
    };

    // 以 JSExport 协议关联 native 的方法
    self.context[@"native"] = self;
```

```
    // 以 block 形式关联 JavaScript function
    self.context[@"log"] =
    ^(NSString *str)
    {
        NSLog(@"%@", str);
    };

    // 以 block 形式关联 JavaScript function
    self.context[@"alert"] =
    ^(NSString *str)
    {
        UIAlertView *alert = [[UIAlertView alloc]initWithTitle:@"msg from js" message:str delegate:nil cancelButtonTitle:@"ok" otherButtonTitles:nil, nil];
        [alert show];
    };

    __block typeof(self) weakSelf = self;
    self.context[@"addSubView"] =
    ^(NSString *viewname)
    {
        UIView *view = [[UIView alloc]initWithFrame:CGRectMake(10, 500, 300, 100)];
        view.backgroundColor = [UIColor redColor];
        UISwitch *sw = [[UISwitch alloc]init];
        [view addSubview:sw];
        [weakSelf.view addSubview:view];
    };
    //多参数
    self.context[@"mutiParams"] =
    ^(NSString *a,NSString *b,NSString *c)
    {
        NSLog(@"%@ %@ %@",a,b,c);
    };
```

```
#pragma mark - JSExport Methods

- (void)handleFactorialCalculateWithNumber:(NSNumber *)number
{
    NSLog(@"%@", number);

    NSNumber *result = [self calculateFactorialOfNumber:number];

    NSLog(@"%@", result);

    [self.context[@"showResult"] callWithArguments:@[result]];
}
```

3. OC 调用 JS

方式一：

NSString *jsStr = [NSString stringWithFormat:@”showAlert(“%@”),@”这是 JS 的 alert”]

[_webView stringByEvaluatingJavaScriptFromString:jsStr];

注意：此方法是同步方法，可能会阻塞 UI（返回了 string）

方式二：

JSContext *context = ...初始化

NSString *textJS = @”showAlert(“这是 JS 的 alert”)”;

[context evaluateScript:textJS];

补充说明：stringByEvaluatingJavaScriptFromString:容易阻塞 UI，因为是同步的。所以苹果建议使用 WKWebView 的 evaluateJavaScript:completionHandler:代理