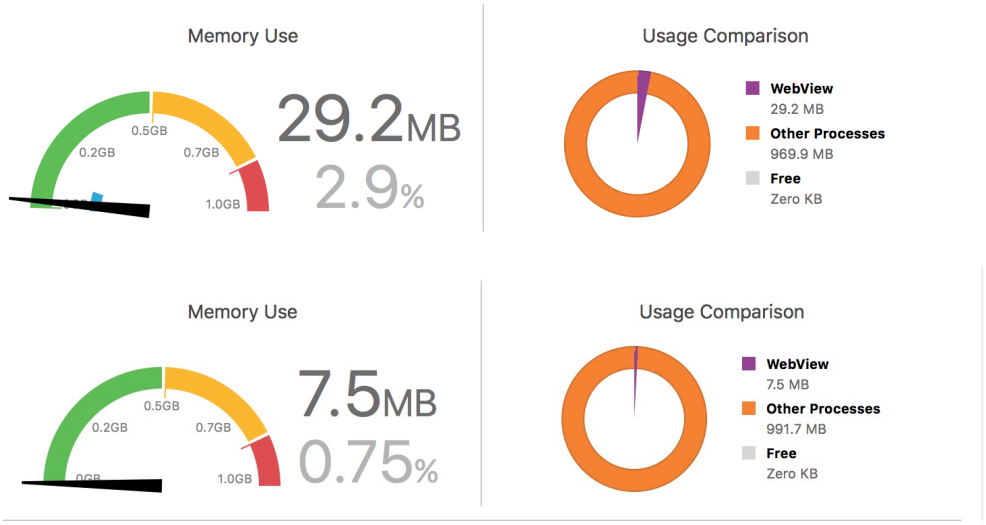
记录WKWebView的一些坑

前言

在项目不支持iOS7之后，我们果断地使用WKWebView替换了原先的UIWebView。先说说相比于UIWebView，WKWebView所具有的优势吧。

占用内存显著减少



同样是打开https://m.baidu.com，上面是使用UIWebView的内存，下面是WKWebView的内存，后者的内存消耗只有前者的1/4。

性能的显著提升

真的是谁用谁知道，从加载效率到滚动帧数再到手势交互，相比于UIWebView都是能够明显感受到质的飞跃。由于使用了和Safari一样的JS引擎，JS性能也得到大幅提升。最最关键的是解决了UIWebView一直存在的内存泄漏问题。

坑

废话说了这么多，也该看看替换WKWebView的过程中，到底遇到了那些坑吧。

Cookie

UIWebView和NSURLConnection都会自动将Cookie存储在NSCookieStorage中，所以我们基本不需要去关心太多的Cookie，网络请求和WebView共用同一套Cookie。但是这个在WKWebView中就行不通了，Cookie不再是通过NSCookieStorage去获取并且自动携带，而是需要手动设置。

创建的时候在WKUserScript中设置Cookie

```
WKWebViewConfiguration *webConfig = [[WKWebViewConfiguration alloc] init];
    //
    webConfig.preferences = [[WKPreferences alloc] init];
    // 0
    webConfig.preferences.minimumFontSize = 10;
    // YES
    webConfig.preferences.javaScriptEnabled = YES;
    // iOSNO
    webConfig.preferences.javaScriptCanOpenWindowsAutomatically = NO;

    // web
    webConfig.processPool = [[WKProcessPool alloc] init];
    // cookiedocument.cookie = 'key=value';
    #warning ()
    NSString *cookieValue = @"document.cookie =
'fromapp=ios';document.cookie = 'channel=appstore';";

    // cookieh5ios
    WKUserContentController* userContentController =
WKUserContentController.new;
    WKUserScript * cookieScript = [[WKUserScript alloc]
                                   initWithSource: cookieValue

injectionTime:WKUserScriptInjectionTimeAtDocumentStart
forMainFrameOnly:NO];
    [userContentController addUserScript:cookieScript];
    webConfig.userContentController = userContentController;

    WKWebView *wkWebView = [[WKWebView alloc] initWithFrame:frame
configuration:webConfig];

    wkWebView.UIDelegate = wkWebView;
    wkWebView.navigationDelegate = wkWebView;
```

加载某个URL的时候添加Cookie

如果WKWebView在加载url的时候需要添加cookie，需要先手动获取当前NSHTTPCookieStorage中的所有cookie，然后将cookie放到NSMutableURLRequest请求头中。

```
- (void)loadRequestWithUrlString:(NSString *)urlString {

    // cookie
    NSMutableDictionary *cookieDic = [NSMutableDictionary dictionary];

    NSMutableString *cookieValue = [NSMutableString stringWithFormat:@""];
    NSHTTPCookieStorage *cookieJar = [NSHTTPCookieStorage
sharedHTTPCookieStorage];
    // cookie
    for (NSHTTPCookie *cookie in [cookieJar cookies]) {
        [cookieDic setObject:cookie.value forKey:cookie.name];
    }

    for (NSString *key in cookieDic) {
        NSString *appendString = [NSString stringWithFormat:@"%@=%@;", key,
[cookieDic valueForKey:key]];
        [cookieValue appendString:appendString];
    }

    NSMutableURLRequest *request = [NSMutableURLRequest
requestWithURL:[NSURL URLWithString:urlString]];
    [request addValue:cookieValue forHTTPHeaderField:@"Cookie"];

    [self loadRequest:request];
}
```

POST请求的参数

在WKWebView中，如果使用NSURLRequest发送一个POST请求，会丢失请求中的HTTPBody，这个实在是无语了……

解决的办法，就是用JS写POST请求，再调用JS。比如先在main bundle中加入一个这样的html文件：

```
<html>
 <head>
     <script>
         // post('URL', {"key": "value"});
         function post(path, params) {
             var method = "post";
             var form = document.createElement("form");
             form.setAttribute("method", method);
             form.setAttribute("action", path);

             for(var key in params) {
                 if(params.hasOwnProperty(key)) {
                     var hiddenField = document.createElement("input");
                     hiddenField.setAttribute("type", "hidden");
                     hiddenField.setAttribute("name", key);
                     hiddenField.setAttribute("value", params[key]);

                     form.appendChild(hiddenField);
                 }
             }
             document.body.appendChild(form);
             form.submit();
         }
     </script>
 </head>
 <body>
 </body>
</html>
```

在发送POST请求的时候，先用webView加载这个文件：

```
// JS
 NSString *path = [[NSBundle mainBundle] pathForResource:@"JSPOST"
ofType:@"html"];
 // html
 NSString *html = [[NSString alloc] initWithContentsOfFile:path
encoding:NSUTF8StringEncoding error:nil];
 // js
 [self.webView loadHTMLString:html baseURL:[[NSBundle mainBundle]
bundleURL]];
```

在页面加载完成的时候，调用JS函数来发送POST请求：

```
- (void)webView:(WKWebView *)webView didFinishNavigation:(WKNavigation
*)navigation {
    //
    if (self.needLoadJSPOST) {
        // JSPOST
        [self postRequestWithJS];
        // FlagNO
        self.needLoadJSPOST = NO;
    }
 }

 // JSPOST
 - (void)postRequestWithJS {
    // POST
    NSString *postData = @"\"username\":\"aaa\",\"password\":\"123\"";
    //
    NSString *urlStr = @"http://www.postexample.com";
    // JavaScript
    NSString *jscript = [NSString stringWithFormat:@"post('%@', {%@});",
urlStr, postData];

    // NSLog(@"Javascript: %@", jscript);
    // JS
    [self.webView evaluateJavaScript:jscript completionHandler:^(id
object, NSError * _Nullable error) {

    }];
 }
```

加载本地HTML

iOS 9的话，我们可以用loadFileURL:allowingReadAccessToURL:方法来加载本地HTML，但是iOS 8就不行了，只能使用用loadHTMLString:baseURL，但是这有一个很大很大的坑：js、css、image这些资源文件必须被baseURL指定在tmp目录下，关键是tmp目录是会被系统清空的，这就非常蛋疼了。iOS 8还有更蛋疼的：baeURL必须指定在tmp/www目录下。