WKWebView API介绍

WKWebView的头文件声明

```
// webview 配置，具体看下面
@property (nonatomic, readonly, copy)
WKWebViewConfiguration *configuration;

// 导航代理
@property (nullable, nonatomic, weak) id
<WKNavigationDelegate> navigationDelegate;

// 用户交互代理
@property (nullable, nonatomic, weak) id <WKUIDelegate>
UIDelegate;

// 页面前进、后退列表
@property (nonatomic, readonly, strong) WKBackForwardList
*backForwardList;

// 默认构造器
- (instancetype)initWithFrame:(CGRect)frame
configuration:(WKWebViewConfiguration *)configuration
NS_DESIGNATED_INITIALIZER;

// 已不再使用
- (instancetype)initWithCoder:(NSCoder *)coder
NS_UNAVAILABLE;

// 与UIWebView一样的加载请求API
- (nullable WKNavigation *)loadRequest:(NSURLRequest
*)request;
```

```
// 加载URL
- (nullable WKNavigation *)loadFileURL:(NSURL *)URL
allowingReadAccessToURL:(NSURL *)readAccessURL
NS_AVAILABLE(10_11, 9_0);

// 直接加载HTML
- (nullable WKNavigation *)loadHTMLString:(NSString
*)string baseURL:(nullable NSURL *)baseURL;

// 直接加载data
- (nullable WKNavigation *)loadData:(NSData *)data
MIMEType:(NSString *)MIMEType characterEncodingName:
(NSString *)characterEncodingName baseURL:(NSURL
*)baseURL NS_AVAILABLE(10_11, 9_0);

// 前进或者后退到某一页面
- (nullable WKNavigation *)goToBackForwardListItem:
(WKBackForwardListItem *)item;

// 页面的标题，这昆支持KVO的
@property (nullable, nonatomic, readonly, copy) NSString
*title;

// 当前请求的URL，它是支持KVO的
@property (nullable, nonatomic, readonly, copy) NSURL
*URL;

// 标识当前是否正在加载内容中，它是支持KVO的
@property (nonatomic, readonly, getter=isLoading) BOOL
loading;

// 当前加载的进度，范围为[0, 1]
@property (nonatomic, readonly) double estimatedProgress;
```

```
// 标识页面中的所有资源是否通过安全加密连接来加载，它是支持KVO的
@property (nonatomic, readonly) BOOL
hasOnlySecureContent;

// 当前导航的证书链，支持KVO
@property (nonatomic, readonly, copy) NSArray
*certificateChain NS_AVAILABLE(10_11, 9_0);

// 是否可以招待goback操作，它是支持KVO的
@property (nonatomic, readonly) BOOL canGoBack;

// 是否可以执行gofarward操作，支持KVO
@property (nonatomic, readonly) BOOL canGoForward;

// 返回上一页面，如果不能返回，则什么也不干
- (nullable WKNavigation *)goBack;

// 进入下一页面，如果不能前进，则什么也不干
- (nullable WKNavigation *)goForward;

// 重新载入页面
- (nullable WKNavigation *)reload;

// 重新从原始URL载入
- (nullable WKNavigation *)reloadFromOrigin;

// 停止加载数据
- (void)stopLoading;

// 执行JS代码
```

```
- (void)evaluateJavaScript:(NSString *)javaScriptString
completionHandler:(void (^ __nullable)(__nullable id,
NSError * __nullable error))completionHandler;

// 标识是否支持左、右swipe手势是否可以前进、后退
@property (nonatomic) BOOL
allowsBackForwardNavigationGestures;

// 自定义user agent，如果没有则为nil
@property (nullable, nonatomic, copy) NSString
*customUserAgent NS_AVAILABLE(10_11, 9_0);

// 在iOS上默认为NO，标识不允许链接预览
@property (nonatomic) BOOL allowsLinkPreview
NS_AVAILABLE(10_11, 9_0);
```

WKWebViewConfiguration配置

```
WKWebViewConfiguration *config = [[WKWebViewConfiguration
alloc] init];
```

WKPreferences偏好设置

```
// 设置偏好设置
config.preferences = [[WKPreferences alloc] init];
// 默认为0
config.preferences.minimumFontSize = 10;
// 默认认为YES
config.preferences.javaScriptEnabled = YES;
// 在iOS上默认为NO，表示不能自动通过窗口打开
config.preferences.javaScriptCanOpenWindowsAutomatically
= NO;
```

WKProcessPool内容处理池

WKProcessPool并没有公开任何的属性或者方法，不需要配置：

```
config.processPool = [[WKProcessPool alloc] init];
```

WKUserContentController内容交互控制器

我们要通过JS与webview内容交互，就需要到这个类了，它的所有属性及方法说明如下：

```
// 只读属性，所有添加的WKUserScript都在这里可以获取到
@property (nonatomic, readonly, copy)
NSArray<WKUserScript *> *userScripts;

// 注入JS
- (void)addUserScript:(WKUserScript *)userScript;

// 移除所有注入的JS
- (void)removeAllUserScripts;

// 添加scriptMessageHandler到所有的frames中，则都可以通过
//
window.webkit.messageHandlers.<name>.postMessage(<message
Body>)
// 发送消息
// 比如，JS要调用我们原生的方法，就可以通过这种方式了
- (void)addScriptMessageHandler:(id
<WKScriptMessageHandler>)scriptMessageHandler name:
(NSString *)name;

// 根据name移除所注入的scriptMessageHandler
- (void)removeScriptMessageHandlerForName:(NSString
*)name;
```

WKUserScript

在WKUserContentController中，所有使用到WKUserScript。WKUserContentController是用于与JS交互的类，而所注入的JS是WKUserScript对象。它的所有属性和方法如下：

```
// JS源代码
@property (nonatomic, readonly, copy) NSString *source;

// JS注入时间
@property (nonatomic, readonly) WKUserScriptInjectionTime
injectionTime;

// 只读属性，表示JS是否应该注入到所有的frames中还是只有main
frame.
@property (nonatomic, readonly,
getter=isForMainFrameOnly) BOOL forMainFrameOnly;

// 初始化方法，用于创建WKUserScript对象
// source：JS源代码
// injectionTime：JS注入的时间
// forMainFrameOnly：是否只注入main frame
- (instancetype)initWithSource:(NSString *)source
injectionTime:(WKUserScriptInjectionTime)injectionTime
forMainFrameOnly:(BOOL)forMainFrameOnly;
```

WKUserScriptInjectionTime

它是一个枚举类型，只有在文档开始加载时注入和加载结束时注入。

```
typedef NS_ENUM(NSInteger, WKUserScriptInjectionTime) {
    WKUserScriptInjectionTimeAtDocumentStart,
    WKUserScriptInjectionTimeAtDocumentEnd
} NS_ENUM_AVAILABLE(10_10, 8_0);
```

WKWebsiteDataStore存储的Web内容

iOS9.0以后才能使用这个类。它是代表webview不同的数据类型，包括cookies、disk、memory caches、WebSQL、IndexedDB数据库和本地存储。
从这里看，要优化Webview好像可以通过它来实现，不过要求iOS9.0以上才能使用。现在6.0都没有抛弃的我，从来不能考虑这种最新的。
WKProcessPool并没有公开任何的属性或者方法，不需要配置：

```
// 默认数据存储
+ (WKWebsiteDataStore *)defaultDataStore;

// 返回非持久化存储，数据不会写入文件系统
+ (WKWebsiteDataStore *)nonPersistentDataStore;

// 已经不可用
- (instancetype)init NS_UNAVAILABLE;

// 只读属性，表示是否是持久化存储
@property (nonatomic, readonly, getter=isPersistent) BOOL
persistent;

// 获取所有web内容的数据存储类型集，比如cookies、disk等
+ (NSSet<NSString *> *)allWebsiteDataTypes;

// 获取某些指定数据存储类型的数据
- (void)fetchDataRecordsOfTypes:(NSSet<NSString *>
*)dataTypes completionHandler:(void (^)
(NSArray<WKWebsiteDataRecord *> *))completionHandler;

// 删除某些指定类型的数据
```

```
- (void)removeDataOfTypes:(NSSet<NSString *> *)dataTypes
forDataRecords:(NSArray<WKWebsiteDataRecord *>
*)dataRecords completionHandler:(void (^)
(void))completionHandler;

// 删除某些指定类型的数据且修改日期是指定的日期
- (void)removeDataOfTypes:(NSSet<NSString *>
*)websiteDataTypes modifiedSince:(NSDate *)date
completionHandler:(void (^)(void))completionHandler;
```

所有的dataTypes是下面这些系统所定义的:

```
WK_EXTERN NSString * const WKWebsiteDataTypeDiskCache
NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const WKWebsiteDataTypeMemoryCache
NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const
WKWebsiteDataTypeOfflineWebApplicationCache
NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const WKWebsiteDataTypeCookies
NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const
WKWebsiteDataTypeSessionStorage NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const WKWebsiteDataTypeLocalStorage
NS_AVAILABLE(10_11, 9_0);

WK_EXTERN NSString * const
WKWebsiteDataTypeWebSQLDatabases NS_AVAILABLE(10_11,
9_0);
```

```
WK_EXTERN NSString * const
WKWebsiteDataTypeIndexedDBDatabases NS_AVAILABLE(10_11,
9_0);
```

WKWebsiteDataRecord

iOS9.0以后才可用。
website的数据存储记录类型，它只有两个属性：

```
// 通常是域名
@property (nonatomic, readonly, copy) NSString
*displayName;

// 存储的数据类型集
@property (nonatomic, readonly, copy) NSSet<NSString *>
*dataTypes;
```

WKSelectionGranularity选择粒度

它表示在webview上选择内容的粒度，只有下面这两种类型：

```
typedef NS_ENUM(NSInteger, WKSelectionGranularity) {
    WKSelectionGranularityDynamic,
    WKSelectionGranularityCharacter,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

它是用于webview内容交互时选择内容的粒度类型设置。比如说，当使用WKSelectionGranularityDynamic时，而所选择的内容是单个块，这时候granularity可能会是单个字符；当所选择的web内容不限制于某个块时，granularity可能会是单个块。

WKNavigationDelegate

```
@protocol WKNavigationDelegate <NSObject>
```

```
@optional

// 决定导航的动作，通常用于处理跨域的链接能否导航。WebKit对跨域进行了安全检查限制，不允许跨域，因此我们要对不能跨域的链接
// 单独处理。但是，对于Safari是允许跨域的，不用这么处理。
// 这个是决定是否Request
- (void)webView:(WKWebView *)webView decidePolicyForNavigationAction:(WKNavigationAction *)navigationAction decisionHandler:(void (^)(WKNavigationActionPolicy))decisionHandler;

// 决定是否接收响应
// 这个是决定是否接收response
// 要获取response，通过WKNavigationResponse对象获取
- (void)webView:(WKWebView *)webView decidePolicyForNavigationResponse:(WKNavigationResponse *)navigationResponse decisionHandler:(void (^)(WKNavigationResponsePolicy))decisionHandler;

// 当main frame的导航开始请求时，会调用此方法
- (void)webView:(WKWebView *)webView didStartProvisionalNavigation:(null_unspecified WKNavigation *)navigation;

// 当main frame接收到服务重定向时，会回调此方法
- (void)webView:(WKWebView *)webView didReceiveServerRedirectForProvisionalNavigation:(null_unspecified WKNavigation *)navigation;

// 当main frame开始加载数据失败时，会回调
```

```
- (void)webView:(WKWebView *)webView
didFailProvisionalNavigation:(null_unspecified
WKNavigation *)navigation withError:(NSError *)error;

// 当main frame的web内容开始到达时，会回调
- (void)webView:(WKWebView *)webView didCommitNavigation:
(null_unspecified WKNavigation *)navigation;

// 当main frame导航完成时，会回调
- (void)webView:(WKWebView *)webView didFinishNavigation:
(null_unspecified WKNavigation *)navigation;

// 当main frame最后下载数据失败时，会回调
- (void)webView:(WKWebView *)webView didFailNavigation:
(null_unspecified WKNavigation *)navigation withError:
(NSError *)error;

// 这与用于授权验证的API，与AFN、UIWebView的授权验证API是一样的
- (void)webView:(WKWebView *)webView
didReceiveAuthenticationChallenge:
(NSURLAuthenticationChallenge *)challenge
completionHandler:(void (^)
(NSURLSessionAuthChallengeDisposition disposition,
NSURLCredential *__nullable
credential))completionHandler;

// 当web content处理完成时，会回调
- (void)webViewWebContentProcessDidTerminate:(WKWebView
*)webView NS_AVAILABLE(10_11, 9_0);

@end
```

WKNavigationActionPolicy

导航动作决定策略：

```
typedef NS_ENUM(NSInteger, WKNavigationActionPolicy) {
    WKNavigationActionPolicyCancel,
    WKNavigationActionPolicyAllow,
} NS_ENUM_AVAILABLE(10_10, 8_0);
```

它是枚举类型，只有Cancel和Allow这两种。设置为Cancel就是不允许导航，就不会跳转链接。

WKNavigationResponse

WKNavigationResponse是导航响应类，通过它可以获取相关响应的信息：

```
NS_CLASS_AVAILABLE(10_10, 8_0)
@interface WKNavigationResponse : NSObject

// 是否是main frame
@property (nonatomic, readonly, getter=isForMainFrame)
BOOL forMainFrame;

// 获取响应response
@property (nonatomic, readonly, copy) NSURLResponse
*response;

// 是否显示MIMEType
@property (nonatomic, readonly) BOOL canShowMIMEType;

@end
```

只有接收响应与不接收响应两种。

WKNavigationAction

WKNavigationAction对象包含关于导航的action的信息，用于make policy decisions。它只有以下几个属性：

```
// 正在请求的导航的frame
@property (nonatomic, readonly, copy) WKFrameInfo
*sourceFrame;

// 目标frame，如果这是新的window，它会是nil
@property (nullable, nonatomic, readonly, copy)
WKFrameInfo *targetFrame;

// 导航类型，如下面的小标题WKNavigationType
@property (nonatomic, readonly) WKNavigationType
navigationType;

// 导航的请求
@property (nonatomic, readonly, copy) NSURLRequest
*request;
```

WKNavigationType

WKNavigationType类型是枚举类型，它的可选值如下：

```
typedef NS_ENUM(NSInteger, WKNavigationType) {
// 链接已经点击
    WKNavigationTypeLinkActivated,
    // 表单提交
    WKNavigationTypeFormSubmitted,
    // 前进、后退
    WKNavigationTypeBackForward,
    // 重新载入
    WKNavigationTypeReload,
```

```
    // 表单重新提交
    WKNavigationTypeFormResubmitted,
    // 其它
    WKNavigationTypeOther = -1,
} NS_ENUM_AVAILABLE(10_10, 8_0);
```

WKUIDelegate

```
@protocol WKUIDelegate <NSObject>

@optional

// 创建新的webview
// 可以指定配置对象、导航动作对象、window特性
- (nullable WKWebView *)webView:(WKWebView *)webView
createWebViewWithConfiguration:(WKWebViewConfiguration
*)configuration forNavigationAction:(WKNavigationAction
*)navigationAction windowFeatures:(WKWindowFeatures
*)windowFeatures;

// webview关闭时回调
- (void)webViewDidClose:(WKWebView *)webView
NS_AVAILABLE(10_11, 9_0);

// 调用JS的alert()方法
- (void)webView:(WKWebView *)webView
runJavaScriptAlertPanelWithMessage:(NSString *)message
initiatedByFrame:(WKFrameInfo *)frame completionHandler:
(void (^)(void))completionHandler;

// 调用JS的confirm()方法
```

```
- (void)webView:(WKWebView *)webView
runJavaScriptConfirmPanelWithMessage:(NSString *)message
initiatedByFrame:(WKFrameInfo *)frame completionHandler:
(void (^)(BOOL result))completionHandler;

// 调用JS的prompt()方法
- (void)webView:(WKWebView *)webView
runJavaScriptTextInputPanelWithPrompt:(NSString *)prompt
defaultText:(nullable NSString *)defaultText
initiatedByFrame:(WKFrameInfo *)frame completionHandler:
(void (^)(NSString * __nullable
result))completionHandler;

@end
```

WKBackForwardList

WKBackForwardList表示webview中可以前进或者后退的页面列表。其声明如下：

```
NS_CLASS_AVAILABLE(10_10, 8_0)
@interface WKBackForwardList : NSObject

// 当前正在显示的item（页面）
@property (nullable, nonatomic, readonly, strong)
WKBackForwardListItem *currentItem;

// 后一页，如果没有就是nil
@property (nullable, nonatomic, readonly, strong)
WKBackForwardListItem *backItem;

// 前一页，如果没有就是nil
```

```
@property (nullable, nonatomic, readonly, strong)
WKBackForwardListItem *forwardItem;

// 根据下标获取某一个页面的item
- (nullable WKBackForwardListItem *)itemAtIndex:
(NSInteger)index;

// 可以进行goback操作的页面列表
@property (nonatomic, readonly, copy)
NSArray<WKBackForwardListItem *> *backList;

// 可以进行goforward操作的页面列表
@property (nonatomic, readonly, copy) NSArray
*forwardList;

@end
```

WKBackForwardListItem

页面导航前进、后退列表项：

```
@interface WKBackForwardListItem : NSObject

// 该页面的URL
@property (readonly, copy) NSURL *URL;

// 该页面的title
@property (nullable, readonly, copy) NSString *title;

// 初始请求该item的请求的URL
@property (readonly, copy) NSURL *initialURL;
```

```
@end
```

结束