/* WKWebView */

/* មាតិកា */

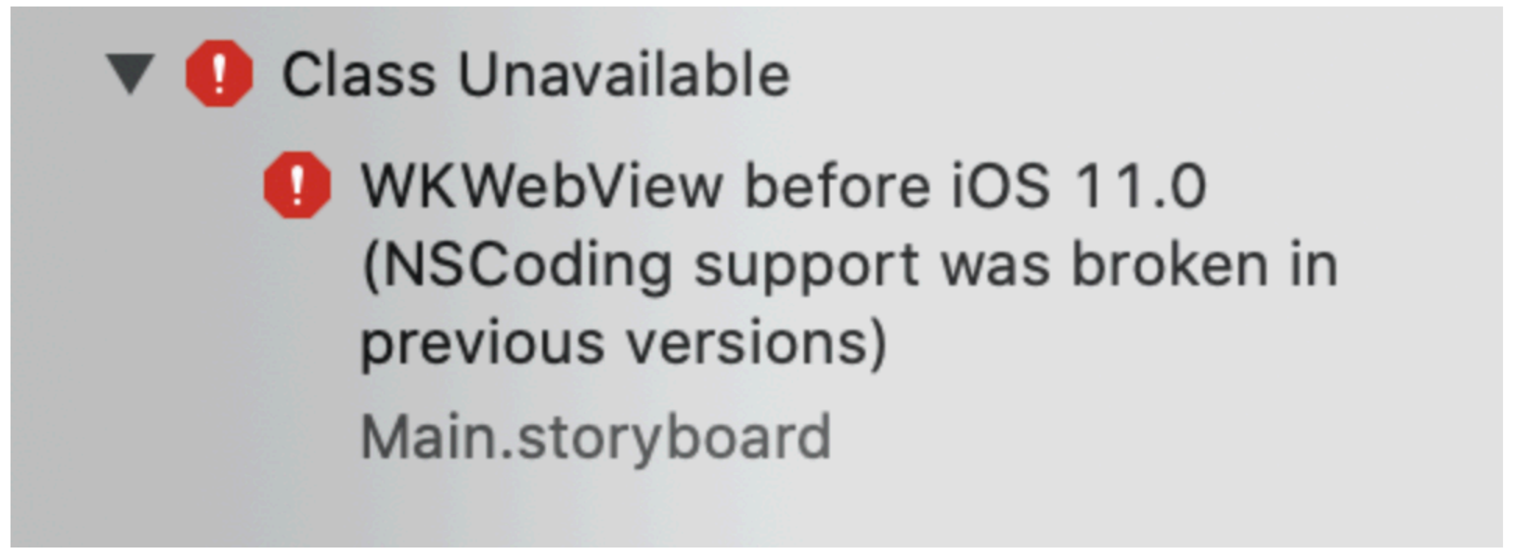
//១; ចំនុចចាប់ផ្តើម

//២; ប្រើប្រាស់Protocol

//៣; LoadURL

//៤; Cookie

//១; ចំនុចចាប់ផ្តើម

- UIWebView(iOS2.0 - 12.0 Deprecated)
- WKWebView(iOS8.0+)
- 

Class Unavailable

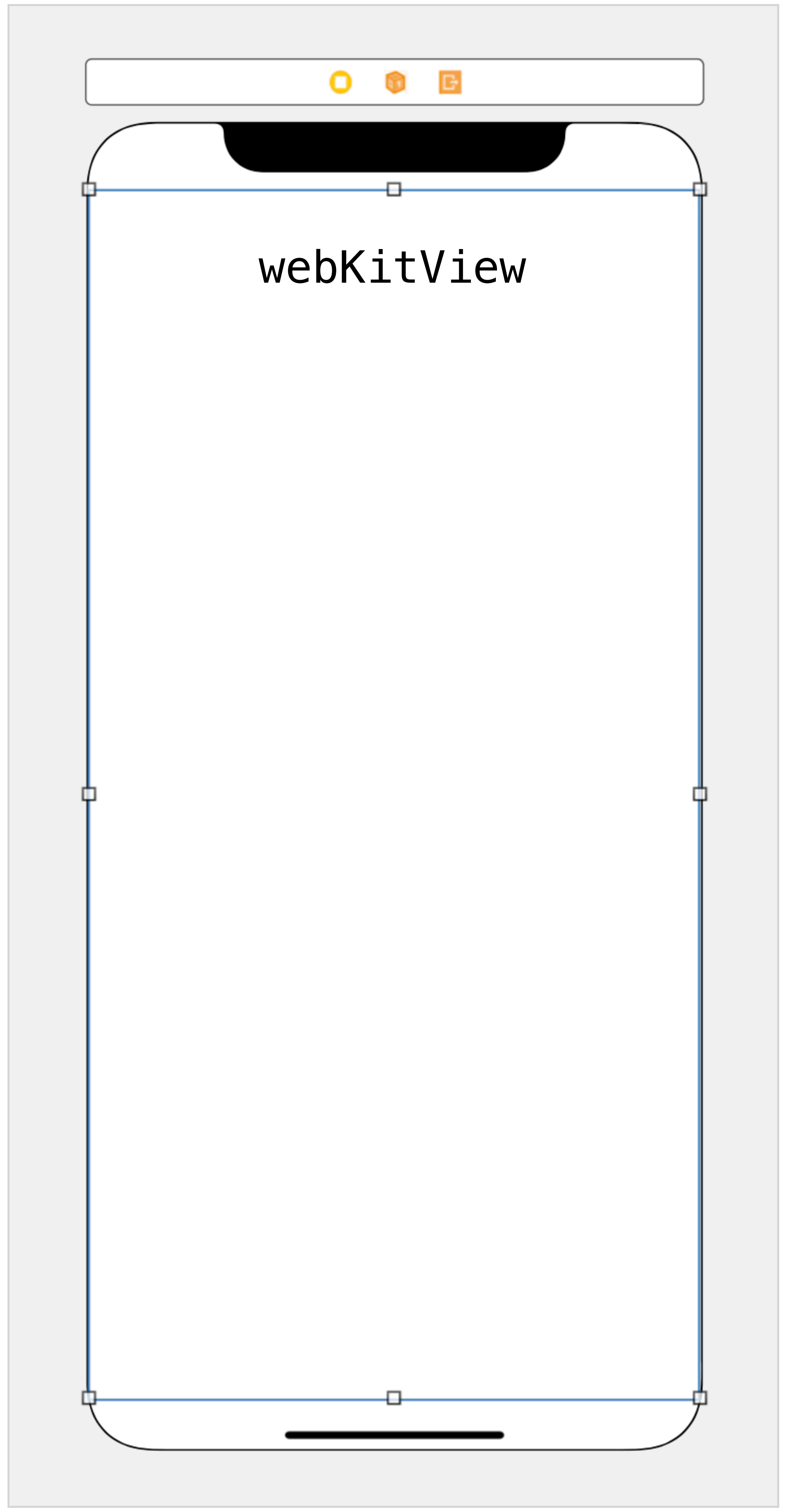
WKWebView before iOS 11.0 (NSCoding support was broken in previous versions)

Main.storyboard

//១; ចំនុចចាប់ផ្តើម

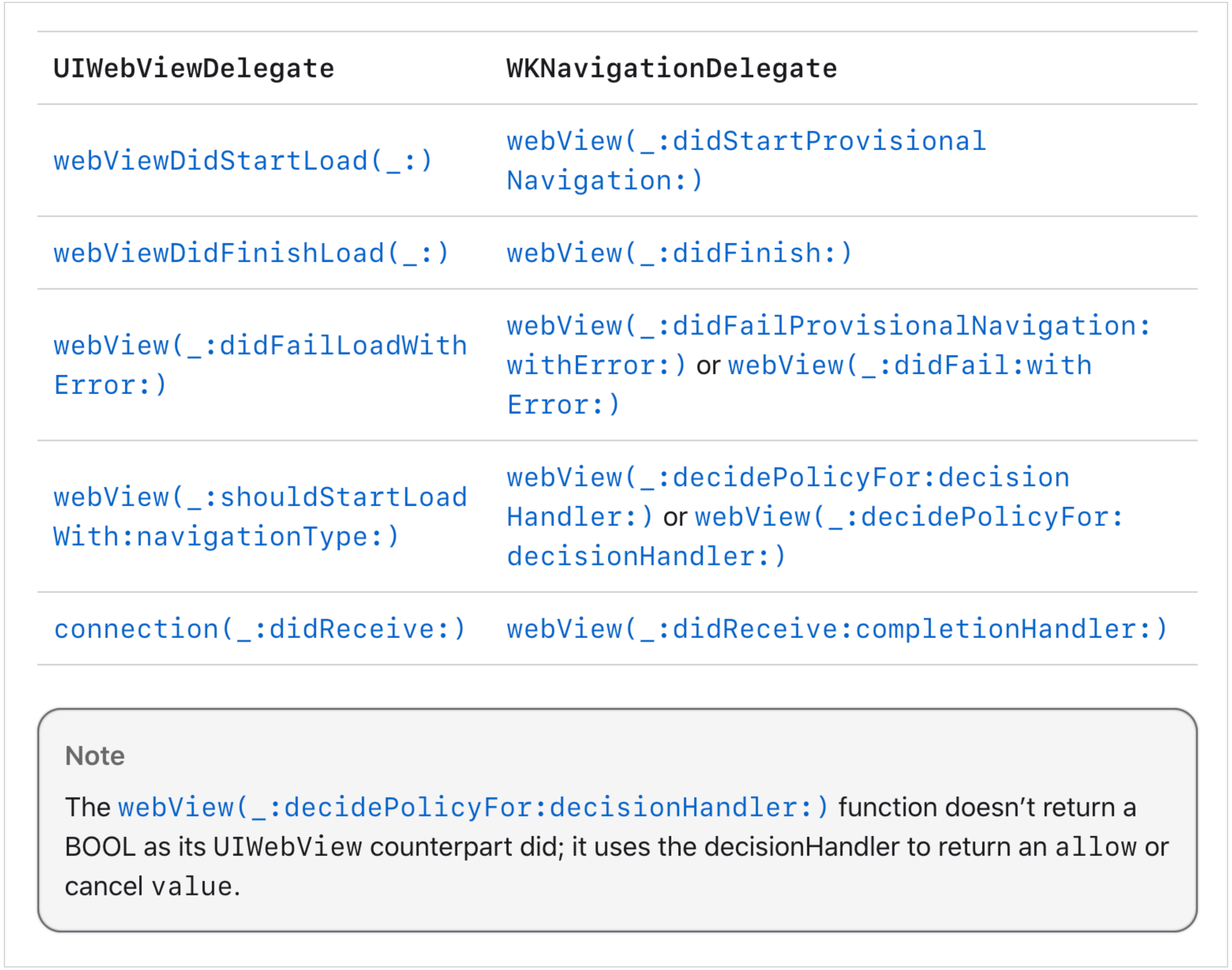
- Setup web view ដោយប្រើប្រាស់ SubView និង Constraint

```
    private func setupBaseWebview() {
        self.webKit = BaseWebView(andURL: self.urlStr, user_agent: user_agent )

        self.webKitView.addSubview(webKit)

        self.webKit.translatesAutoresizingMaskIntoConstraints = false
        NSLayoutConstraint.activate([
            webKit.widthAnchor.constraint(equalTo: webKitView.widthAnchor),
            webKit.heightAnchor.constraint(equalTo: webKitView.heightAnchor)
                ])

        self.webKit.delegate        = self
        self.webKit.param           = self.param

        self.webKit.loadURL()
    }
```



//២; ការប្រើប្រាស់Protocol

- ប្រៀបធៀបរវាង UIWebViewDelegate ជាមួយនឹង WKNavigationDelegate



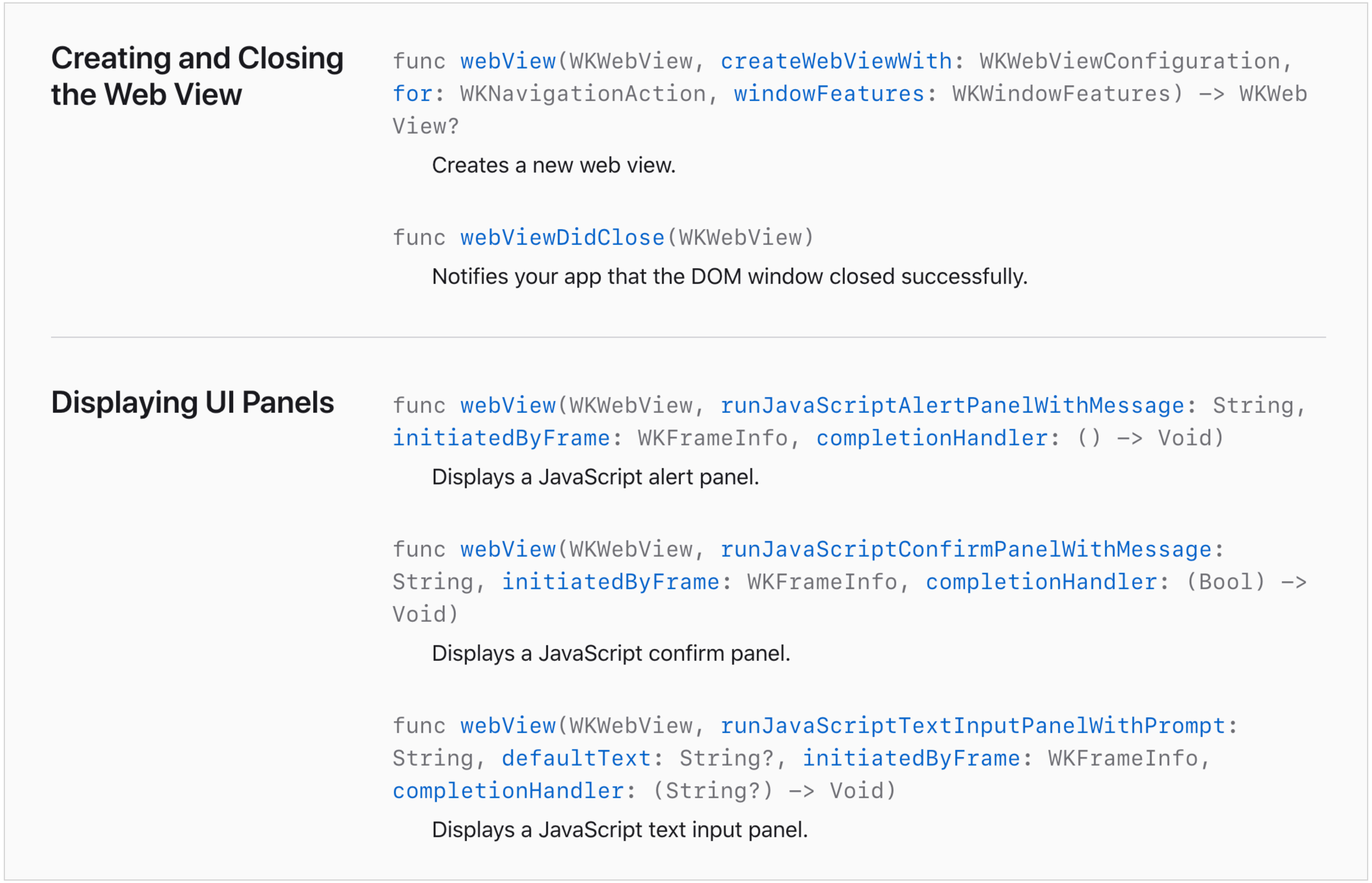
UIWebViewDelegate	WKNavigationDelegate
webViewDidStartLoad(_:)	webView(_:didStartProvisionalNavigation:)
webViewDidFinishLoad(_:)	webView(_:didFinish:)
webView(_:didFailLoadWithError:)	webView(_:didFailProvisionalNavigation:withError:) or webView(_:didFail:withError:)
webView(_:shouldStartLoadWith:navigationType:)	webView(_:decidePolicyFor:decisionHandler:) or webView(_:decidePolicyFor:decisionHandler:)
connection(_:didReceive:)	webView(_:didReceive:completionHandler:)

Note

The webView(_:decidePolicyFor:decisionHandler:) function doesn't return a BOOL as its UIWebView counterpart did; it uses the decisionHandler to return an allow or cancel value.

//២; ការប្រើប្រាស់Protocol

- WKUIDelegate



Creating and Closing the Web View

```
func webView(WKWebView, createWebViewWith: WKWebViewConfiguration,
for: WKNavigationAction, windowFeatures: WKWindowFeatures) -> WKWeb
View?
```

Creates a new web view.

```
func webViewDidClose(WKWebView)
```

Notifies your app that the DOM window closed successfully.

Displaying UI Panels

```
func webView(WKWebView, runJavaScriptAlertPanelWithMessage: String,
initiatedByFrame: WKFrameInfo, completionHandler: () -> Void)
```

Displays a JavaScript alert panel.

```
func webView(WKWebView, runJavaScriptConfirmPanelWithMessage:
String, initiatedByFrame: WKFrameInfo, completionHandler: (Bool) ->
Void)
```

Displays a JavaScript confirm panel.

```
func webView(WKWebView, runJavaScriptTextInputPanelWithPrompt:
String, defaultText: String?, initiatedByFrame: WKFrameInfo,
completionHandler: (String?) -> Void)
```

Displays a JavaScript text input panel.

//២; ការប្រើប្រាស់Protocol

– ការបើកនិងបិទweb view

```
    func webView(_ webView: WKWebView, createWebViewWith configuration: WKWebViewConfiguration, for
navigationAction: WKNavigationAction, windowFeatures: WKWindowFeatures) -> WKWebView? {

        //파라미터로 받은 configuration
        createWebView = WKCookieWebView(frame: webView.frame, configuration: configuration,
useRedirectCookieHandling: true)
        createWebView!.navigationDelegate = self
        createWebView!.uiDelegate = self

        //오토레이아웃 처리
        createWebView!.autoresizingMask = [.flexibleWidth, .flexibleHeight]

        self.addSubview(createWebView!)

        return createWebView!
    }

    func webViewDidClose(_ webView: WKWebView) {

        webView.removeFromSuperview()
        createWebView = nil
    }
```

```
<a href="https://www.bizplay.co.kr" target="_blank">Bizplay</a>
```

//២; ការប្រើប្រាស់Protocol

- ដើម្បីបង្ហាញAlert Popup ពី web view

```
func webView(_ webView: WKWebView, runJavaScriptAlertPanelWithMessage message: String, initiatedByFrame
frame: WKFrameInfo, completionHandler: @escaping () -> Void) {
    <#code#>
}
```

- alert(“alert”);

```
func webView(_ webView: WKWebView, runJavaScriptConfirmPanelWithMessage message: String, initiatedByFrame
frame: WKFrameInfo, completionHandler: @escaping (Bool) -> Void) {
    <#code#>
}
```

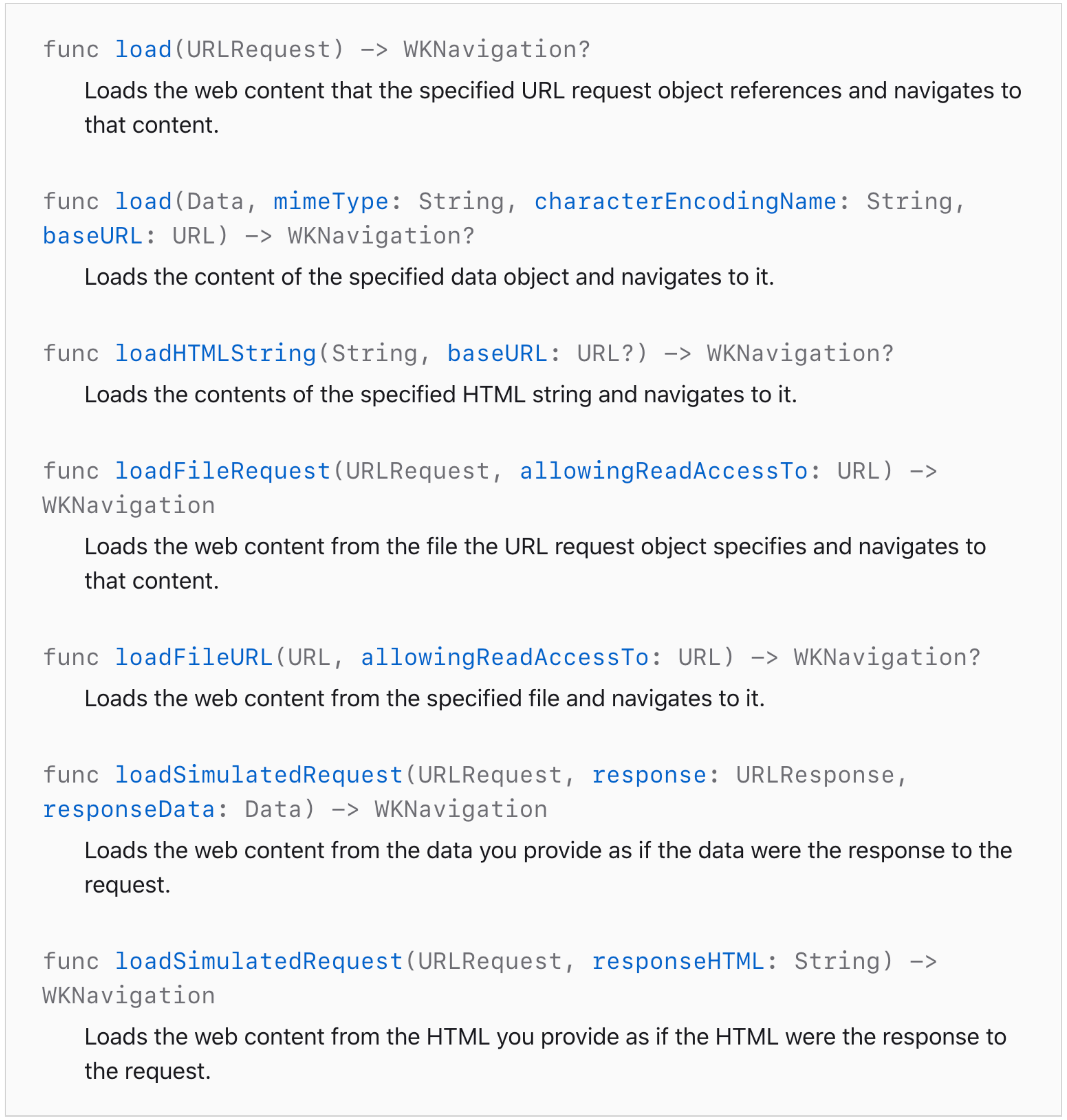
- confirm(“confirm”);

```
func webView(_ webView: WKWebView, runJavaScriptTextInputPanelWithPrompt prompt: String, defaultText:
String?, initiatedByFrame frame: WKFrameInfo, completionHandler: @escaping (String?) -> Void) {
    <#code#>
}
```

- prompt(“prompt”, "default Text");

//៣; LoadURL

– វិធីប្រើប្រាស់ Load



```
func load(URLRequest) -> WKNavigation?
```
Loads the web content that the specified URL request object references and navigates to that content.

```
func load(Data, mimeType: String, characterEncodingName: String,
baseURL: URL) -> WKNavigation?
```
Loads the content of the specified data object and navigates to it.

```
func loadHTMLString(String, baseURL: URL?) -> WKNavigation?
```
Loads the contents of the specified HTML string and navigates to it.

```
func loadFileRequest(URLRequest, allowingReadAccessTo: URL) ->
WKNavigation
```
Loads the web content from the file the URL request object specifies and navigates to that content.

```
func loadFileURL(URL, allowingReadAccessTo: URL) -> WKNavigation?
```
Loads the web content from the specified file and navigates to it.

```
func loadSimulatedRequest(URLRequest, response: URLResponse,
responseData: Data) -> WKNavigation
```
Loads the web content from the data you provide as if the data were the response to the request.

```
func loadSimulatedRequest(URLRequest, responseHTML: String) ->
WKNavigation
```
Loads the web content from the HTML you provide as if the HTML were the response to the request.

//៣; LoadURL

- វិធីប្រើប្រាស់ Load with URLRequest (Simple)

```
func load(URLRequest) -> WKNavigation?
```

Loads the web content that the specified URL request object references and navigates to that content.

```
var urlRequest = URLRequest(url: URL(string: urlString)!)

//- setup header
if let extraHeader = self.addOnHeader {
    if let authorization = extraHeader["Authorization"] {
        urlRequest.addValue(authorization, forHTTPHeaderField: "Authorization")
    }
}

//- setup param
if let myParam = self.param {
    urlRequest.httpMethod = "POST"
    urlRequest.httpBody = myParam.data(using: .utf8)
}

 _ = webview.load(urlRequest)
```

//៣; LoadURL

- វិធីប្រើប្រាស់ Load with URLRequest ដោយប្រើ URLSession ដូច DataAccess

```
    override func load(_ request: URLRequest) -> WKNavigation? {

        guard (request.httpBody != nil) else {
            return super.load(request)
        }

        requestWithCookieHandling(request, success: { (newRequest , response, data) in
            DispatchQueue.main.async {
                self.syncCookiesInJS()
                if let data = data, let response = response {
                    let _ = self.webViewLoad(data: data, response: response)
                } else {
                    self.syncCookies(newRequest, nil, { (cookieRequest) in
                        let _ = super.load(cookieRequest)
                    })
                }
            }
        }, failure: {
            // let WKWebView handle the network error
            DispatchQueue.main.async {
                self.syncCookies(request, nil, { (newRequest) in
                    let _ = super.load(newRequest)
                })
            }
        })
       return nil
    }
```

```
- private func requestWithCookieHandling(_ request: URLRequest, success: @escaping (URLRequest, HTTPURLResponse?,
Data?) -> Void, failure: @escaping () -> Void) {
```

//ണ; LoadURL

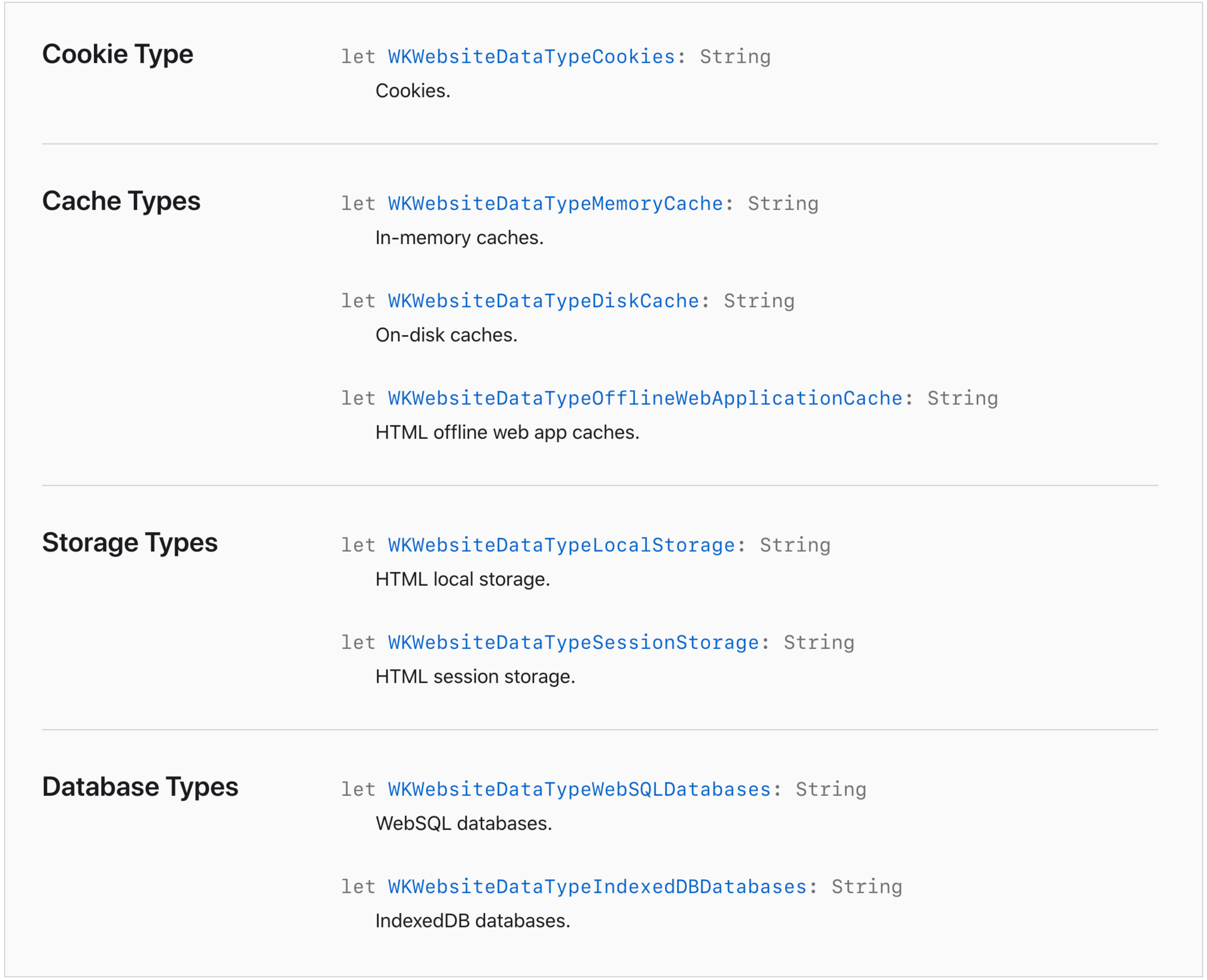
```swift
    private func requestWithCookieHandling(_ request: URLRequest, success: @escaping (URLRequest, HTTPURLResponse?, Data?) -> Void, failure: @escaping
() -> Void) {
        let sessionConfig = URLSessionConfiguration.default
        let session = URLSession(configuration: sessionConfig, delegate: self, delegateQueue: nil)
        let task = session.dataTask(with: request) { (data, response, error) in
            if let _ = error {
                failure()
            } else {
                if let response = response as? HTTPURLResponse {

                    let code = response.statusCode
                    if code == 200 {
                        // for code 200 return data to load data directly
                        success(request, response, data)

                    } else if code >= 300 && code <  400  {
                        // for redirect get location in header,and make a new URLRequest
                        guard let location = response.allHeaderFields["Location"] as? String, let redirectURL = URL(string: location) else {
                            failure()
                            return
                        }

                        let request = URLRequest(url: redirectURL, cachePolicy: .reloadIgnoringLocalAndRemoteCacheData, timeoutInterval: 5)
                        success(request, nil, nil)

                    } else {
                        success(request, response, data)
                    }
                }
            }
        }
        task.resume()
    }
```

//៩; Cookie

- WKWebsiteDataStore
- WKWebsiteDataStore ប្រើសម្រាប់ គ្រប់គ្រង data store ទាំងអស់របស់ web view
- WKHTTPCookieStore
- WKHTTPCookieStore ប្រើប្រាស់សម្រាប់គ្រប់គ្រងតែលើ Cookie

//៥; Cookie

- Data Store Record Types



Cookie Type

```
let WKWebsiteDataTypeCookies: String
```
Cookies.

Cache Types

```
let WKWebsiteDataTypeMemoryCache: String
```
In-memory caches.

```
let WKWebsiteDataTypeDiskCache: String
```
On-disk caches.

```
let WKWebsiteDataTypeOfflineWebApplicationCache: String
```
HTML offline web app caches.

Storage Types

```
let WKWebsiteDataTypeLocalStorage: String
```
HTML local storage.

```
let WKWebsiteDataTypeSessionStorage: String
```
HTML session storage.

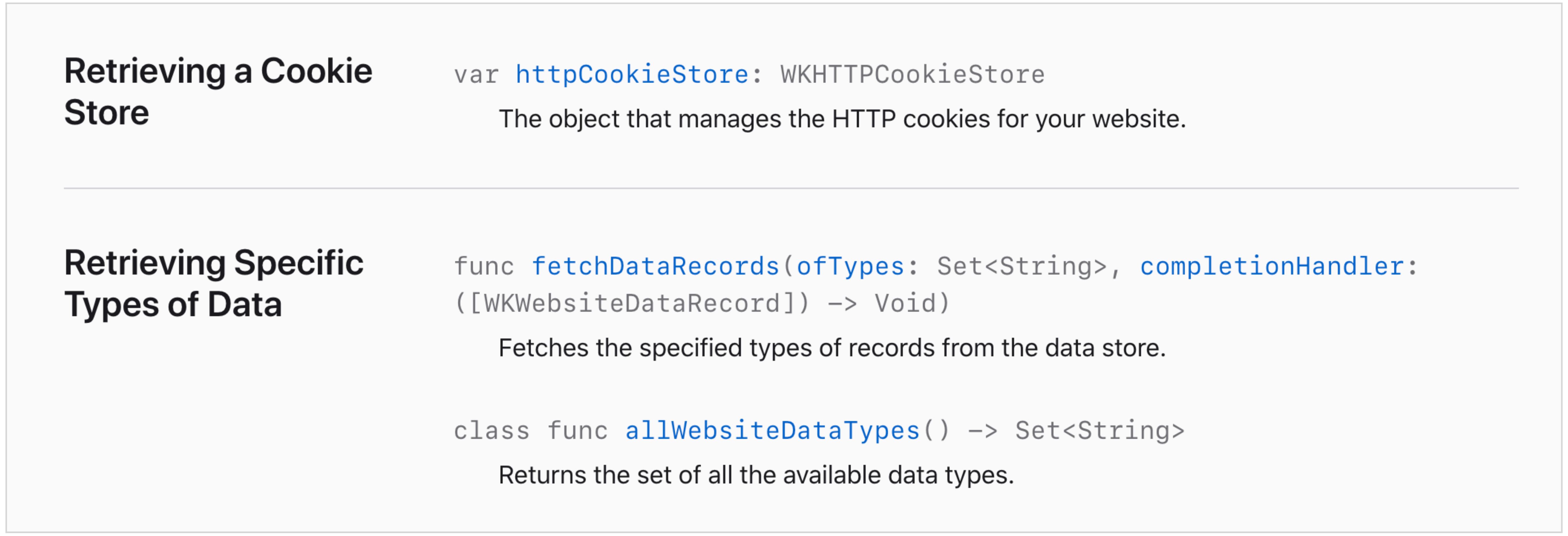
Database Types

```
let WKWebsiteDataTypeWebSQLDatabases: String
```
WebSQL databases.

```
let WKWebsiteDataTypeIndexedDBDatabases: String
```
IndexedDB databases.

//៥; Cookie

- វិធីទាយយក web data ដោយប្រើ WKWebsiteDataStore



Retrieving a Cookie Store

```
var httpCookieStore: WKHTTPCookieStore
```

The object that manages the HTTP cookies for your website.

Retrieving Specific Types of Data

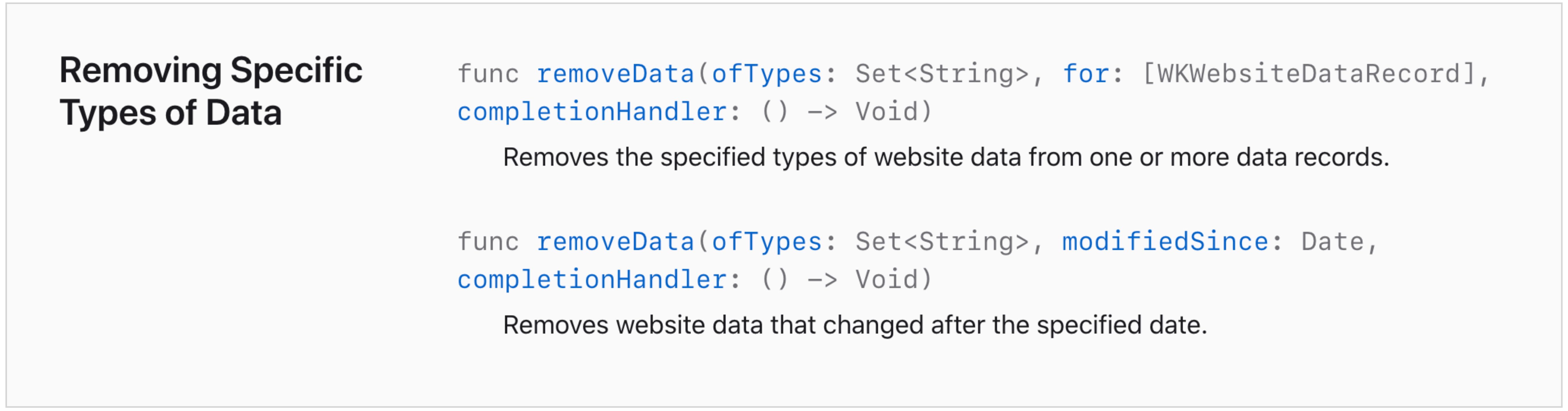
```
func fetchDataRecords(ofTypes: Set<String>, completionHandler:
([WKWebsiteDataRecord]) -> Void)
```

Fetches the specified types of records from the data store.

```
class func allWebsiteDataTypes() -> Set<String>
```

Returns the set of all the available data types.

- វិធីលុប web data ដោយប្រើ WKWebsiteDataStore



Removing Specific Types of Data

```
func removeData(ofTypes: Set<String>, for: [WKWebsiteDataRecord],
completionHandler: () -> Void)
```

Removes the specified types of website data from one or more data records.

```
func removeData(ofTypes: Set<String>, modifiedSince: Date,
completionHandler: () -> Void)
```

Removes website data that changed after the specified date.

//៤; Cookie

- វិធីទាញយក web data ដោយប្រើ fetchDataRecords

```
WKWebsiteDataStore.default().fetchDataRecords(ofTypes: Set<String>) { <#[WKWebsiteDataRecord]#> in
        <#code#>
}

WKWebsiteDataStore.allWebsiteDataTypes()
```

- វិធីទាញយក Cookie ដោយប្រើ httpCookieStore.getAllCookies

```
WKWebsiteDataStore.default().httpCookieStore.getAllCookies { <#[HTTPCookie]#> in
       <#code#>
}
```

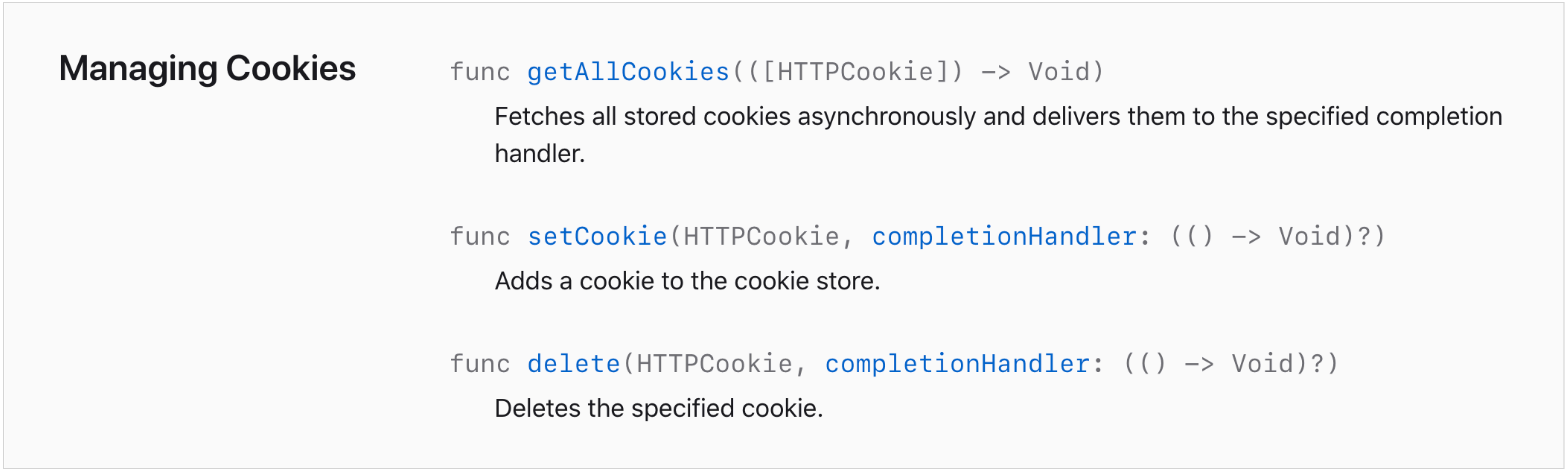
//៩; Cookie

- វិធីលុប web data

```
WKWebsiteDataStore.default().fetchDataRecords(ofTypes: WKWebsiteDataStore.allWebsiteDataTypes()) {
records in
            records.forEach { record in
                WKWebsiteDataStore.default().removeData(ofTypes: record.dataTypes, for: [record],
completionHandler: {})
            }
        }
```

//៩; Cookie

- ការគ្រប់គ្រង Cookie ដោយប្រើ WKHTTPCookieStore



Managing Cookies

```
func getAllCookies(([HTTPCookie]) -> Void)
```

Fetches all stored cookies asynchronously and delivers them to the specified completion handler.

```
func setCookie(HTTPCookie, completionHandler: (() -> Void)?)
```

Adds a cookie to the cookie store.

```
func delete(HTTPCookie, completionHandler: (() -> Void)?)
```

Deletes the specified cookie.

//៤; Cookie

- វិធីទាញយក Cookie

```
webview.configuration.websiteDataStore.httpCookieStore.getAllCookies { <#[HTTPCookie]#> in
        <#code#>
}
```

- វិធីរក្សាទុក Cookie

```
webview.configuration.websiteDataStore.httpCookieStore.setCookie(<#HTTPCookie#>)
```

- វិធីលុប Cookie

```
webview.configuration.websiteDataStore.httpCookieStore.delete(<#HTTPCookie#>)
```

//៩; Cookie

- ការប្រើប្រាស់ Cookie ដោយប្រើ WKWebsiteDataStore

```
webView.configuration.websiteDataStore.httpCookieStore.getAllCookies { cookies in
    let cookieDict = HTTPCookie.requestHeaderFields(with: cookies)
    if let cookieStr = cookieDict["Cookie"] {
        request.addValue(cookieStr, forHTTPHeaderField: "Cookie")
    }
}
```

- ការប្រើប្រាស់ Cookie ដោយប្រើ HTTPCookieStorage

```
HTTPCookieStorage.shared.getCookiesFor(task, completionHandler: { (cookies) in
    if let cookies = cookies {

        let cookieDict = HTTPCookie.requestHeaderFields(with: cookies)

        if let cookieStr = cookieDict["Cookie"] {
            request.addValue(cookieStr, forHTTPHeaderField: "Cookie")
        }
    }

})
```

//៤; Cookie

– ការយល់ខុស

```swift
    private func syncCookiesInJS(for request: URLRequest? = nil) {
        if let url = request?.url, let cookies = HTTPCookieStorage.shared.cookies(for: url) {
            let script = jsCookiesString(for: cookies)
            let cookieScript = WKUserScript(source: script, injectionTime: .atDocumentStart, forMainFrameOnly:
false)
            self.configuration.userContentController.addUserScript(cookieScript)
            /* set cookie to WKWebViewConfiguration */
            for cookie in cookies {
                self.configuration.websiteDataStore.httpCookieStore.setCookie(cookie,completionHandler: nil)
            }

        } else if let cookies = HTTPCookieStorage.shared.cookies {
            let script = jsCookiesString(for: cookies)
            let cookieScript = WKUserScript(source: script, injectionTime: .atDocumentStart, forMainFrameOnly:
false)
            self.configuration.userContentController.addUserScript(cookieScript)
            /* set cookie to WKWebViewConfiguration */
            for cookie in cookies {
                self.configuration.websiteDataStore.httpCookieStore.setCookie(cookie,completionHandler: nil)
            }
        }
    }
```

//ថ; Cookie

```
private func jsCookiesString(for cookies: [HTTPCookie]) -> String {
        var result = ""
        let dateFormatter = DateFormatter()
        dateFormatter.timeZone = TimeZone(abbreviation: "UTC")
        dateFormatter.dateFormat = "EEE, d MMM yyyy HH:mm:ss zzz"

        for cookie in cookies {
            result += "document.cookie='\(cookie.name)=\(cookie.value); domain=\(cookie.domain); path=\
(cookie.path); "
            if let date = cookie.expiresDate {
                result += "expires=\(dateFormatter.string(from: date)); "
            }
            if (cookie.isSecure) {
                result += "secure; "
            }
            result += "'; "
        }
        return result
    }
```

//[illegible]; Cookie

```
private func syncCookies(_ request: URLRequest, _ task: URLSessionTask? = nil, _ completion: @escaping (URLRequest) -> Void) {

        var request = request
        var cookiesArray = [HTTPCookie]()

        if let task = task { /** Old way set Cookies */
            HTTPCookieStorage.shared.getCookiesFor(task, completionHandler: { (cookies) in
                if let cookies = cookies {
                    cookiesArray.append(contentsOf: cookies)

                    let cookieDict = HTTPCookie.requestHeaderFields(with: cookiesArray)
                    if let cookieStr = cookieDict["Cookie"] {
                        request.addValue(cookieStr, forHTTPHeaderField: "Cookie")
                    }
                }
                completion(request)
            }) /* End */
        } else  if let url = request.url {
            /** New way set Cookies */
            DispatchQueue.main.async {
                self.configuration.websiteDataStore.httpCookieStore.getAllCookies { cookies in
                    let cookieDict = HTTPCookie.requestHeaderFields(with: cookies)
                    if let cookieStr = cookieDict["Cookie"] {
                        request.addValue(cookieStr, forHTTPHeaderField: "Cookie")
                    }
                    completion(request)
                }
            } /* End */
        }
    }
```

/* សូមចប់ */