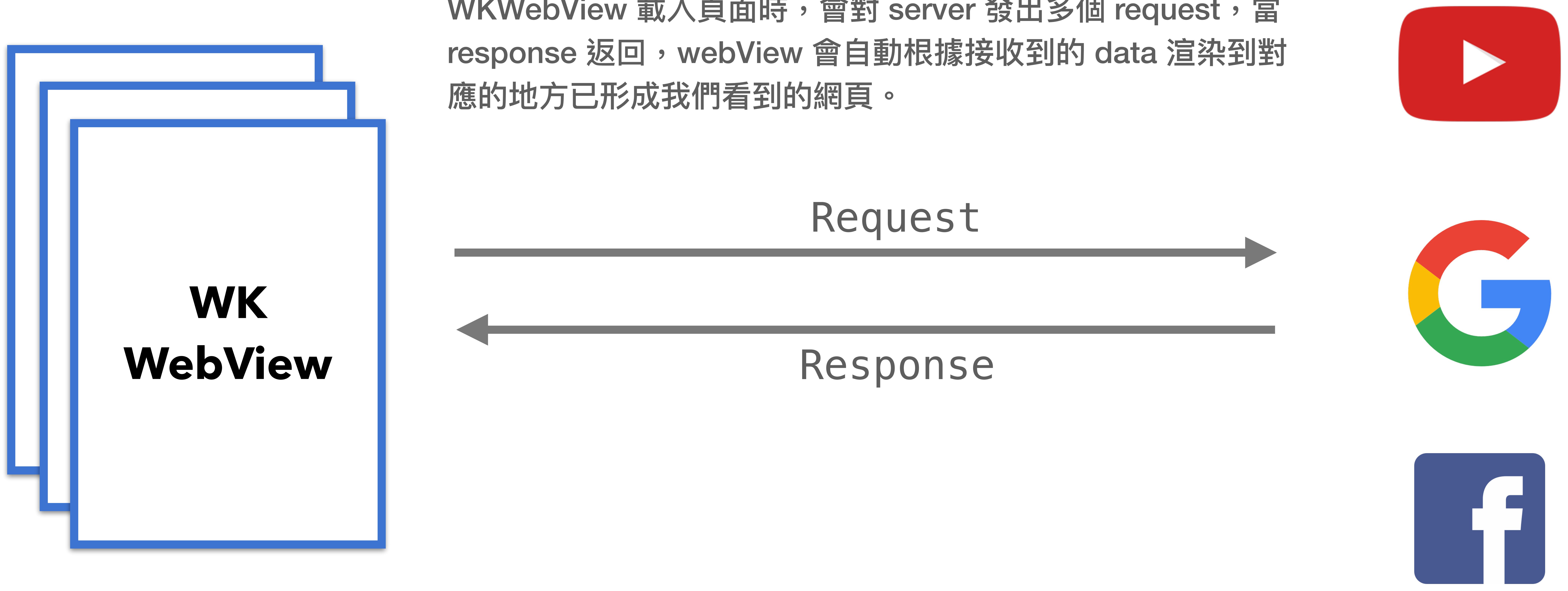
Browser
全網下載 ＆ Proxy

WKWebView Request 攔截方案

RickH. 20200914

How it works?

WKWebView 載入頁面時，會對 server 發出多個 request，當 response 返回，webView 會自動根據接收到的 data 渲染到對應的地方已形成我們看到的網頁。



Goal

- Set Proxy on WKWebView
- Hook Request & Response inout WKWebView

Solutions

Proxy

- UIWebView ❌ Deprecated since iOS 10
- VPN - iOS Network Extension ⚠️ Proxy the traffic from whole iPhone.
  ⚠️ Needed to control VPN’s lifecycle.
  ⚠️ Request User permissions.
- WKURLSchemeHandler + URLSession with Proxy ⚠️ Swizzled Private API ✅

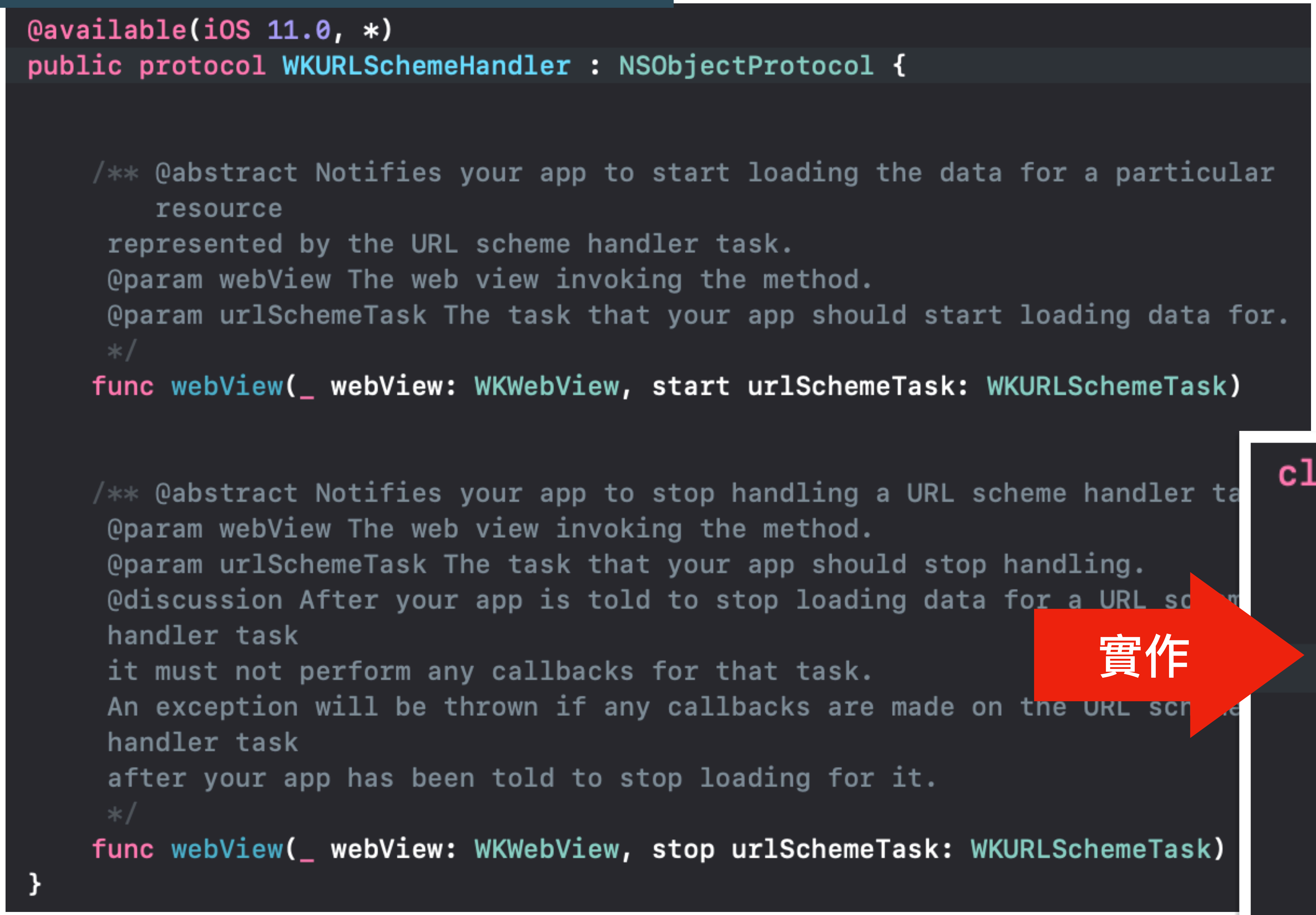
Solutions

Hook Request

- AJaxHook ❌ Doesn’t work on non-XMLHttpRequest
- URLProtocol ❌ Lost request body
  ⚠️ Swizzled Private API
- WKURLSchemeHandler ⚠️ Swizzled Private API. ✅

實作 WKURLSchemeHandler

WKURLSchemeHandler 官方文件



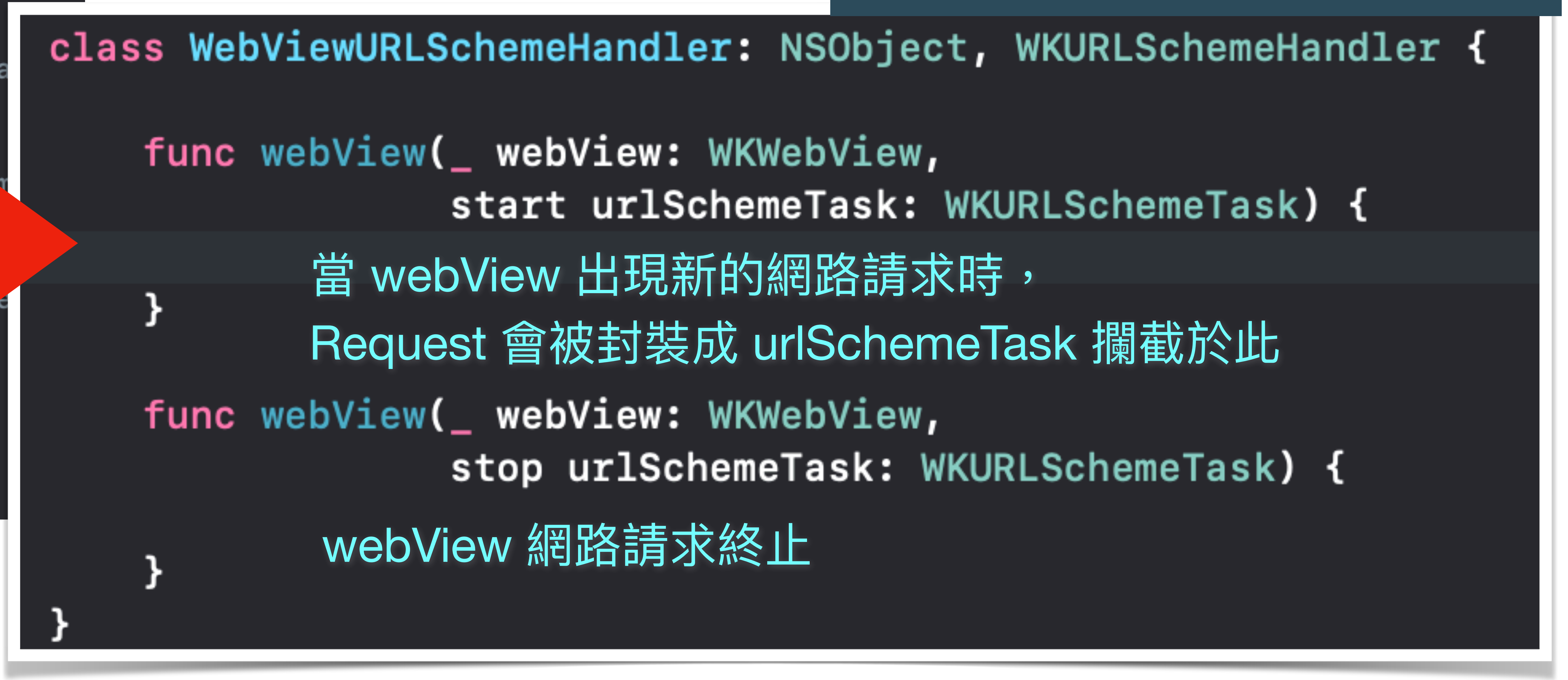
```
@available(iOS 11.0, *)
public protocol WKURLSchemeHandler : NSObjectProtocol {


    /** @abstract Notifies your app to start loading the data for a particular
        resource
     represented by the URL scheme handler task.
     @param webView The web view invoking the method.
     @param urlSchemeTask The task that your app should start loading data for.
     */
    func webView(_ webView: WKWebView, start urlSchemeTask: WKURLSchemeTask)


    /** @abstract Notifies your app to stop handling a URL scheme handler ta
     @param webView The web view invoking the method.
     @param urlSchemeTask The task that your app should stop handling.
     @discussion After your app is told to stop loading data for a URL sc
     handler task
     it must not perform any callbacks for that task.
     An exception will be thrown if any callbacks are made on the URL sch
     handler task
     after your app has been told to stop loading for it.
     */
    func webView(_ webView: WKWebView, stop urlSchemeTask: WKURLSchemeTask)
}
```

實作

@WebViewURLSchemeHandler



```
class WebViewURLSchemeHandler: NSObject, WKURLSchemeHandler {

    func webView(_ webView: WKWebView,
                 start urlSchemeTask: WKURLSchemeTask) {

    }

    func webView(_ webView: WKWebView,
                 stop urlSchemeTask: WKURLSchemeTask) {

    }
}
```

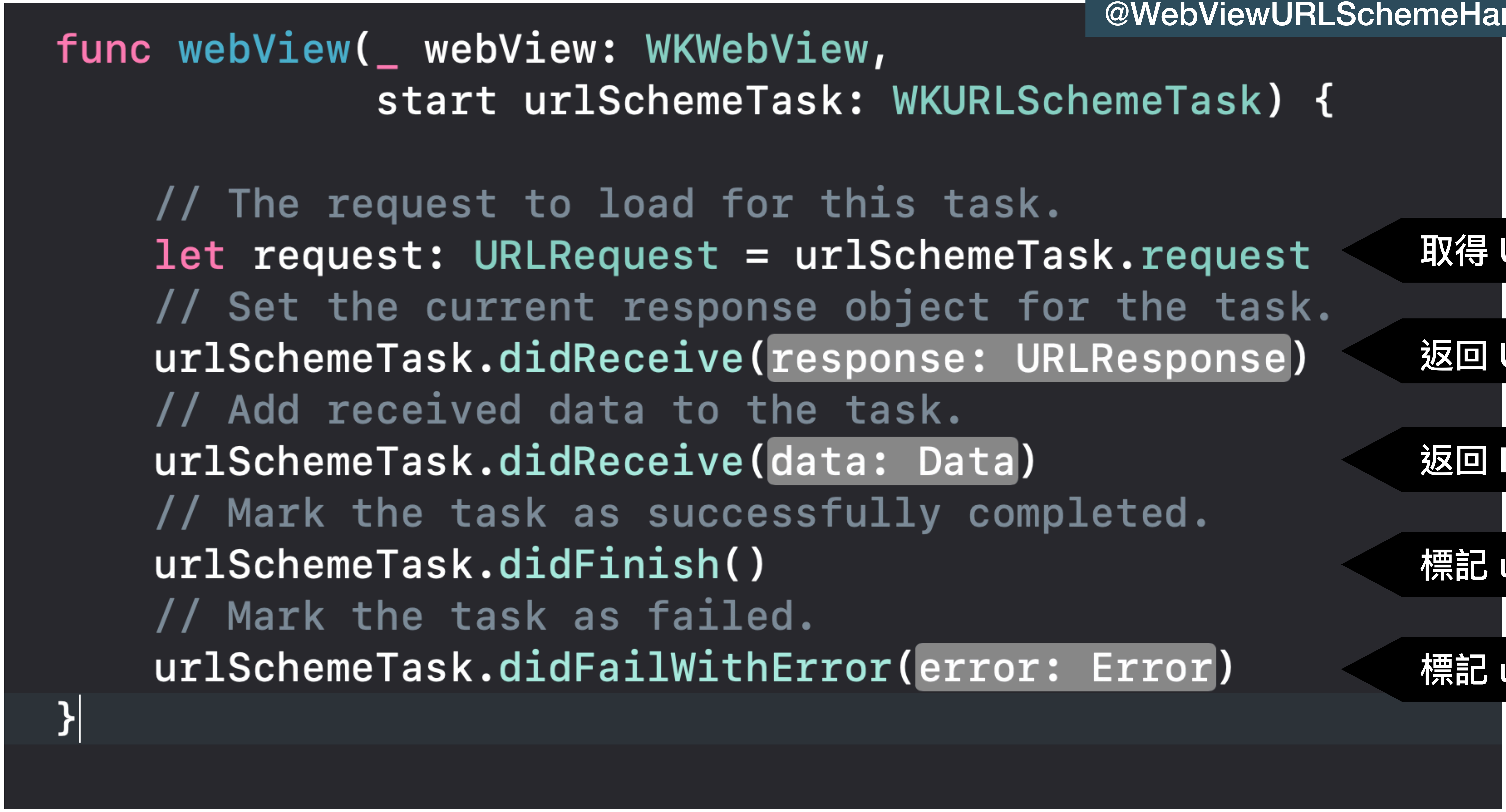
當 webView 出現新的網路請求時，
Request 會被封裝成 urlSchemeTask 攔截於此

webView 網路請求終止

UrlSchemeTask API

urlSchemeTask.request 可取得 webView 被攔截之 request
接著可以透過 urlSchemeTask 所提供的 API 將網路回應回填給 webView 做渲染。

@WebViewURLSchemeHandler



```swift
func webView(_ webView: WKWebView,
             start urlSchemeTask: WKURLSchemeTask) {

    // The request to load for this task.
    let request: URLRequest = urlSchemeTask.request
    // Set the current response object for the task.
    urlSchemeTask.didReceive(response: URLResponse)
    // Add received data to the task.
    urlSchemeTask.didReceive(data: Data)
    // Mark the task as successfully completed.
    urlSchemeTask.didFinish()
    // Mark the task as failed.
    urlSchemeTask.didFailWithError(error: Error)
}
```

取得 URLRequest

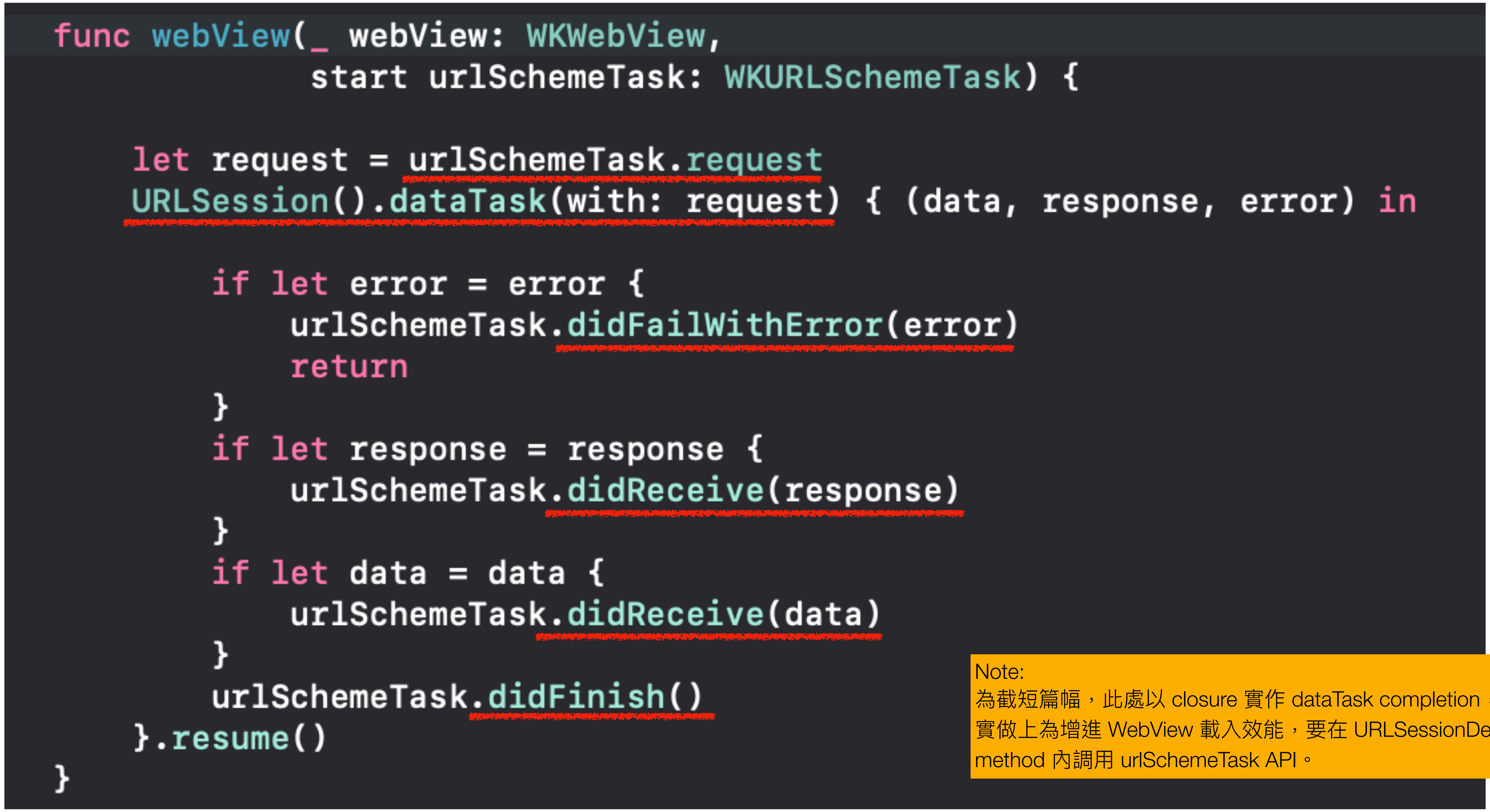
返回 URLResponse

返回 Data

標記 urlSchemeTask 完成

標記 urlSchemeTask 失敗

基礎架構



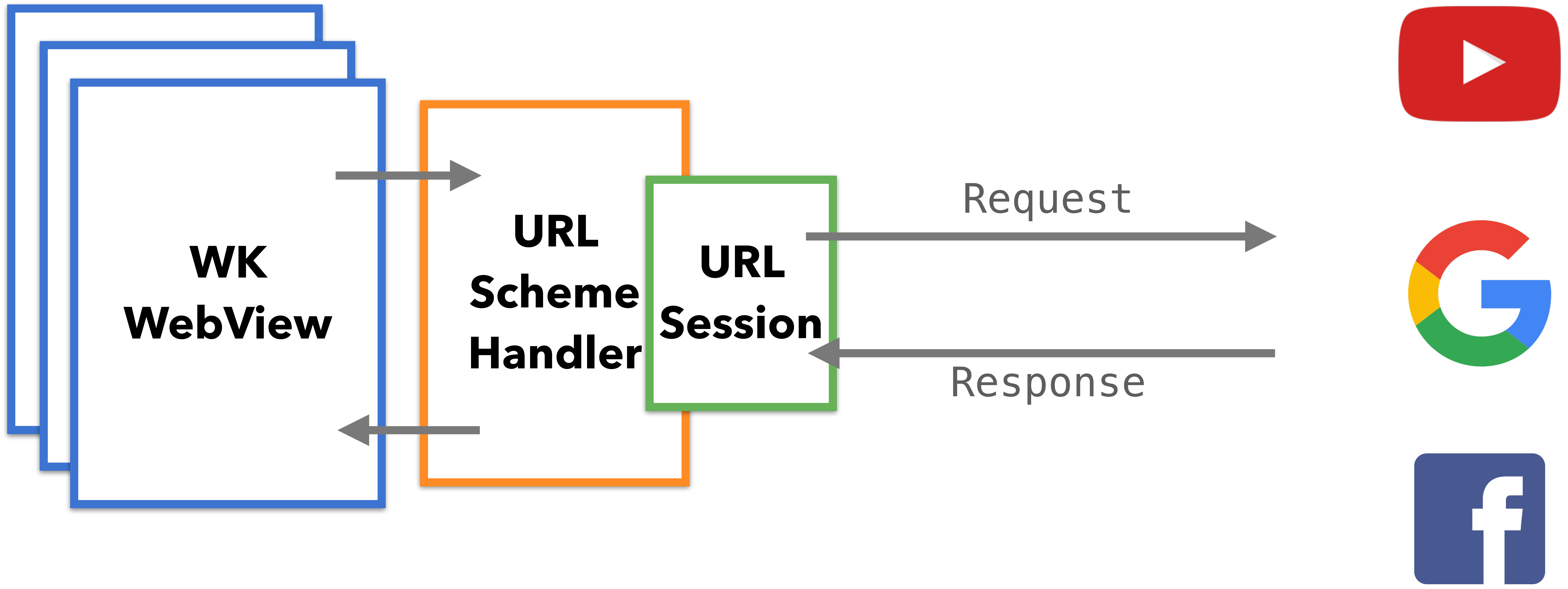
```
func webView(_ webView: WKWebView,
             start urlSchemeTask: WKURLSchemeTask) {

    let request = urlSchemeTask.request
    URLSession().dataTask(with: request) { (data, response, error) in

        if let error = error {
            urlSchemeTask.didFailWithError(error)
            return
        }
        if let response = response {
            urlSchemeTask.didReceive(response)
        }
        if let data = data {
            urlSchemeTask.didReceive(data)
        }
        urlSchemeTask.didFinish()
    }.resume()
}
```

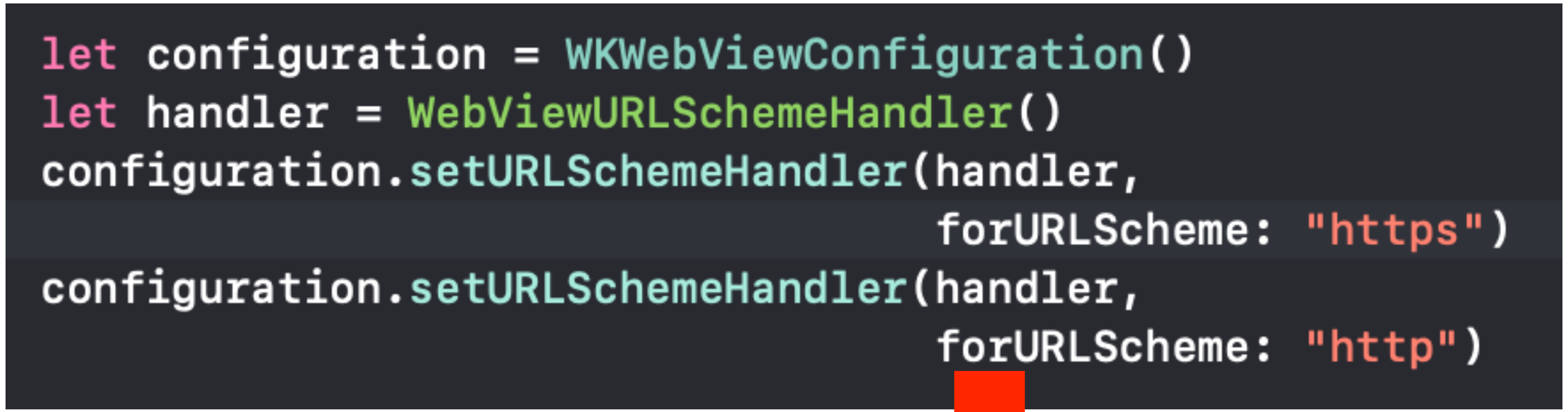
Note:
為截短篇幅，此處以 closure 實作 dataTask completion，
實做上為增進 WebView 載入效能，要在 URLSessionDelegate
method 內調用 urlSchemeTask API。

基礎架構

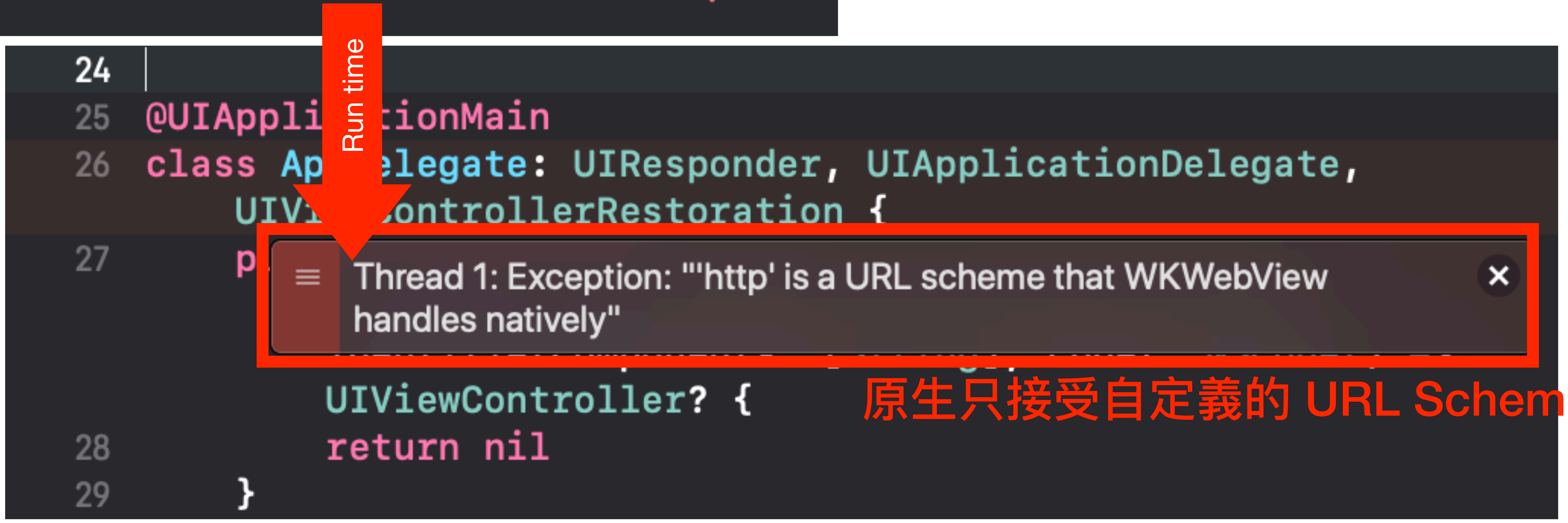

WK
WebView
URL
Scheme
Handler
URL
Session
Request
Response

Add WKURLSchemeHandler to WKWebViewConfiguration

a. WKWebConfiguration set WKURKSchemeHandler



```
let configuration = WKWebViewConfiguration()
let handler = WebViewURLSchemeHandler()
configuration.setURLSchemeHandler(handler,
                                  forURLScheme: "https")
configuration.setURLSchemeHandler(handler,
                                  forURLScheme: "http")
```



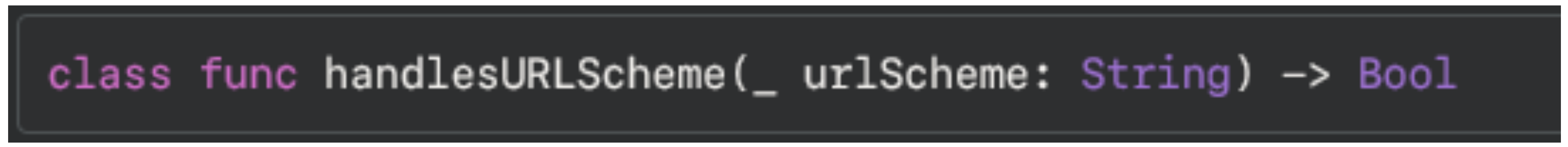
```
@UIAppli    tionMain
class Ap    elegate: UIResponder, UIApplicationDelegate,
    UIVi    ontrollerRestoration {
    p
            UIViewController? {
        return nil
    }
```

Thread 1: Exception: "'http' is a URL scheme that WKWebView handles natively"

原生只接受自定義的 URL Scheme

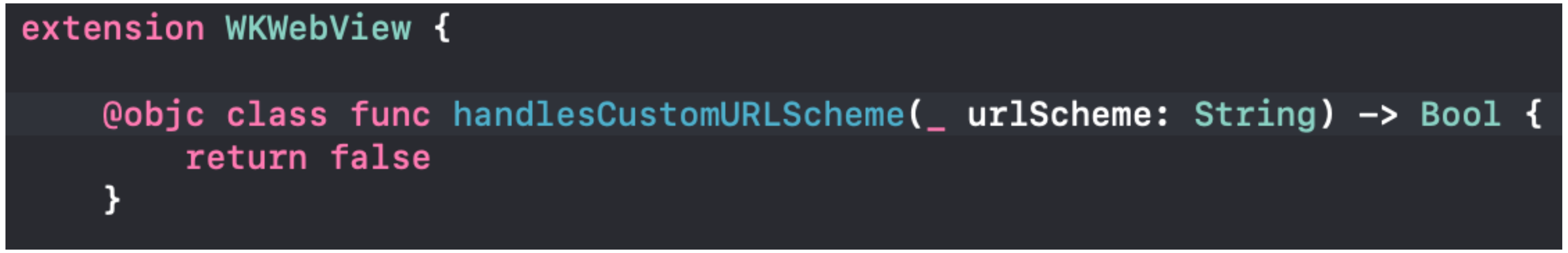
How to handle Http/Https on WKURLSchemeHandler

b.再加上... Swizzle method handlesURLScheme



```
class func handlesURLScheme(_ urlScheme: String) -> Bool
```

According to Google...
得知 WebKit 是藉由 handlesURLScheme
來判斷 scheme 是由 Native 或 WKURLSchemeHandler 處理

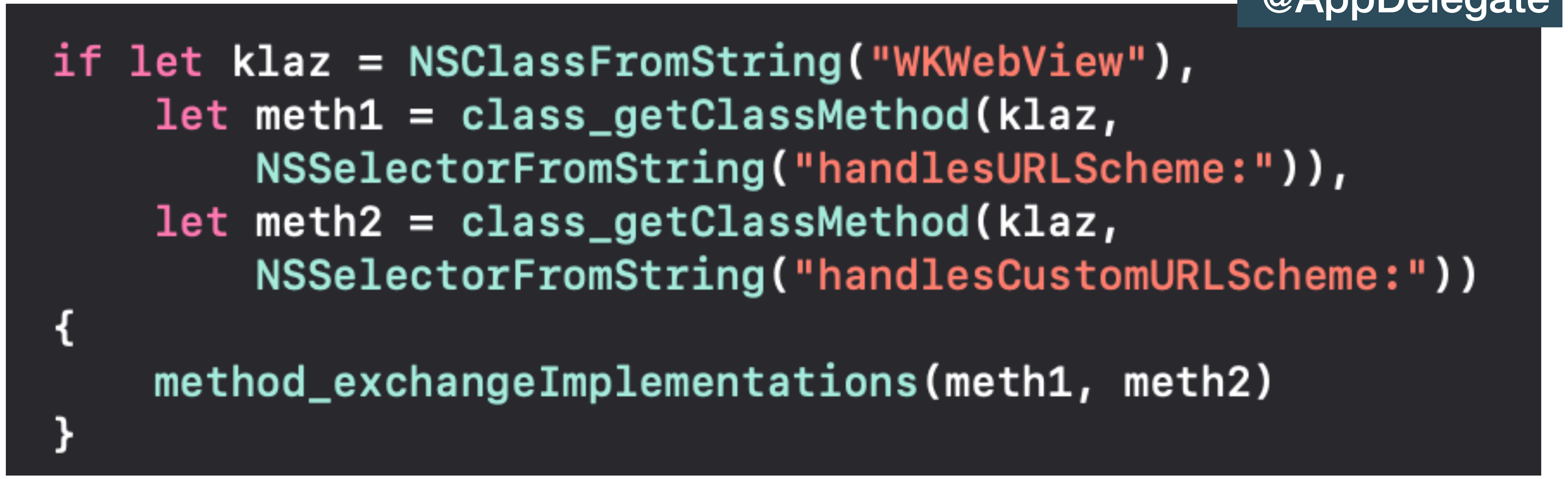


```
extension WKWebView {

    @objc class func handlesCustomURLScheme(_ urlScheme: String) -> Bool {
        return false
    }
```

仿照 handlesURLScheme 的 interface 寫一方法
並 return false 代表註冊的所有 urlScheme
都透過 WKURLSchemeHandler 處理

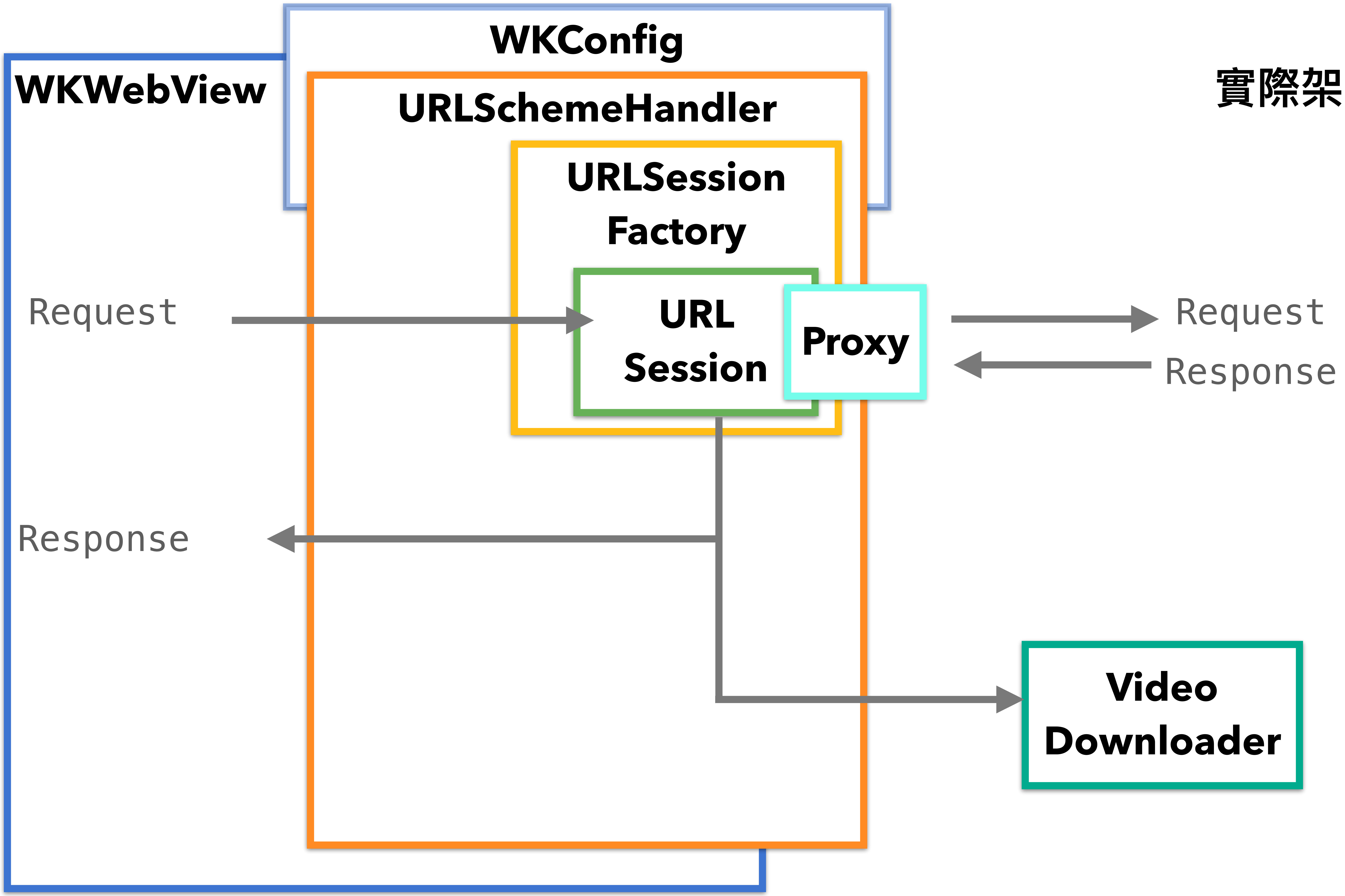
@AppDelegate



```
 if let klaz = NSClassFromString("WKWebView"),
     let meth1 = class_getClassMethod(klaz,
         NSSelectorFromString("handlesURLScheme:")),
     let meth2 = class_getClassMethod(klaz,
         NSSelectorFromString("handlesCustomURLScheme:"))
 {
     method_exchangeImplementations(meth1, meth2)
 }
```

利用 Method Swizzling 將兩方法替換
當 Webkit 要調用 handlesURLScheme 時
實際上會調用 handlesCustomURLScheme

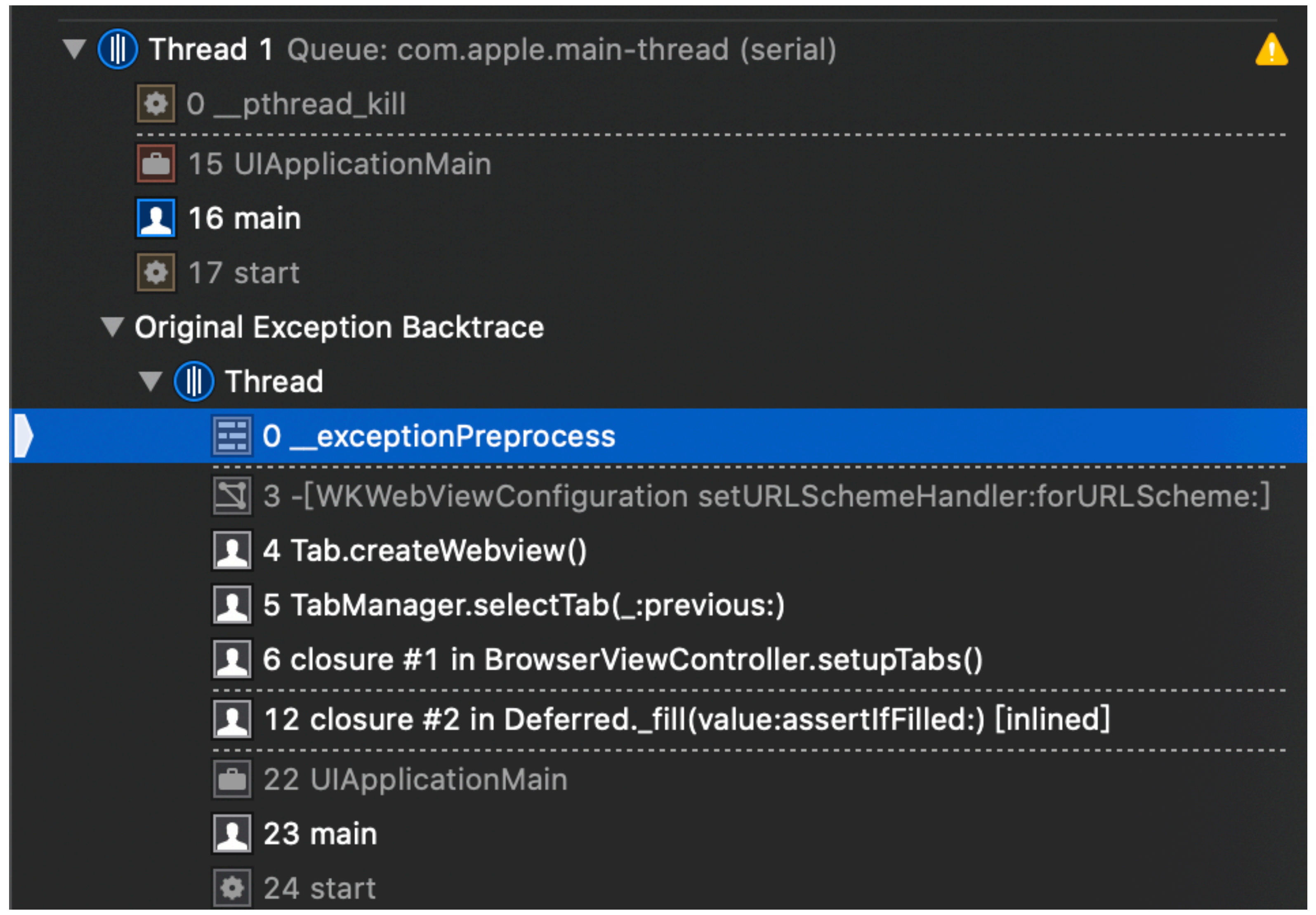
實際架構


WKConfig
WKWebView
URLSchemeHandler
URLSession Factory
URL Session
Proxy
Request
Response
Request
Response
Video Downloader

聖光阿！你有看到滿滿的坑嗎

- Duplicate setURLSchemeHandler exception.
- UrlSchemeTask API throws objc exception.
- WKWebView 第一次載入網域無回應: UrlSchemeTask API no response on iOS 12.
- Server 身分認證: URLAuthenticationChallenge delegate.
- 轉址後 WKWebView.url 無對應變動: willPerformHTTPRedirection delegate.
- Crash on posting large body.

Duplicate setURLSchemeHandler exception



```
▼ Thread 1 Queue: com.apple.main-thread (serial)
    0 __pthread_kill
    15 UIApplicationMain
    16 main
    17 start
  ▼ Original Exception Backtrace
    ▼ Thread
        0 __exceptionPreprocess
        3 -[WKWebViewConfiguration setURLSchemeHandler:forURLScheme:]
        4 Tab.createWebview()
        5 TabManager.selectTab(_:previous:)
        6 closure #1 in BrowserViewController.setupTabs()
        12 closure #2 in Deferred._fill(value:assertIfFilled:) [inlined]
        22 UIApplicationMain
        23 main
        24 start
```

What: 若已對 WKWebViewConfiguration setURLSchemeHandler “HTTPS”，
再次對同一 Config 實體 setURLSchemeHandler “HTTPS” 則會引發閃退

How: 故調用 setURLSchemeHandler 前需先檢查 urlSchemeHandler 是否為空值

@Tab. createWebView( )

```swift
if configuration!.urlSchemeHandler(forURLScheme: "https") == nil {
    let handler = WebViewURLSchemeHandler(delegate: self)
    configuration!.setURLSchemeHandler(handler, forURLScheme: "https")
    configuration!.setURLSchemeHandler(handler, forURLScheme: "http")
}
```

UrlSchemeTask API throws objc exception

What: 調用 urlSchemeTask API 時
若 Webkit 已認為該 Task 已過期
會直接丟出 Obj-C exception;

How: Obj-C 實作 try-catch 內執行 closure，
在 closure 內調用 urlSchemeTask API

@ExceptionCatcher.h

```
#ifndef ExceptionCatcher_h
#define ExceptionCatcher_h


#endif /* ExceptionCatcher_h */

#import <Foundation/Foundation.h>

NS_INLINE NSException * _Nullable tryBlock(void(^_Nonnull
    tryBlock)(void)) {
    @try {
        tryBlock();
    }
    @catch (NSException *exception) {
        return exception;
    }
    return nil;
}
```

URLSchemeHandler

```
let objcException = tryBlock {
    urlSchemeTask.didReceive(response)
}
```

```
let objcException = tryBlock {
    urlSchemeTask.didReceive(data)
}
```

```
let objcException = tryBlock {
    if let error = error,
        error._code != NSURLErrorCancelled {
        urlSchemeTask.didFailWithError(error)
    } else {
        urlSchemeTask.didFinish()
    }
}
```

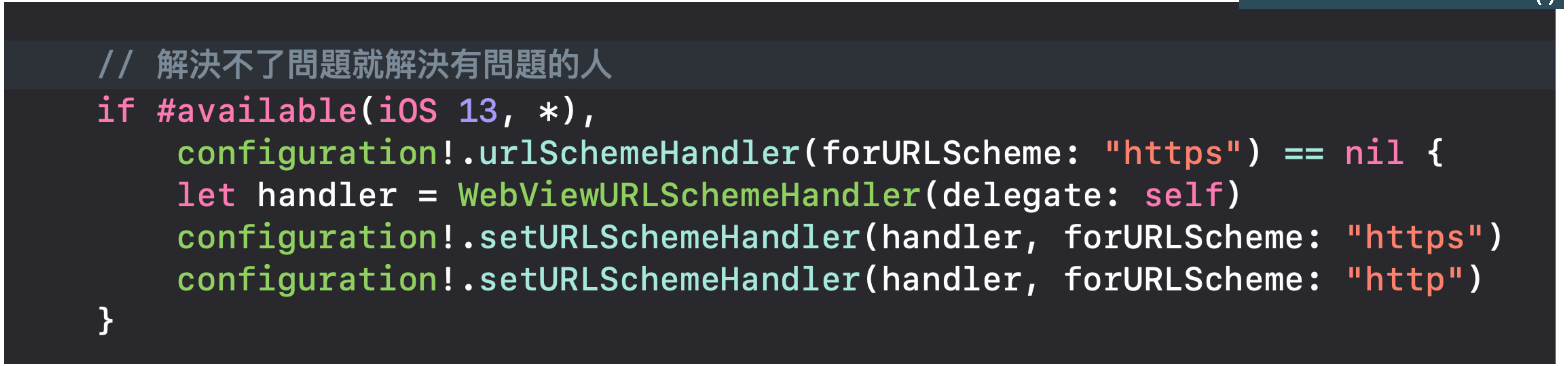
WKWebView 第一次載入網域無回應

UrlSchemeTask API no response on iOS 12.

What: WebView 進入新網域時的第一個請求沒有反應

How: 該問題只發生於 iOS 12，判斷可能為 Webkit 的 bug，
故只針對 iOS 13 以上使用 URLSchemeHandler 功能

@Tab. createWebView( )



```swift
// 解決不了問題就解決有問題的人
if #available(iOS 13, *),
    configuration!.urlSchemeHandler(forURLScheme: "https") == nil {
    let handler = WebViewURLSchemeHandler(delegate: self)
    configuration!.setURLSchemeHandler(handler, forURLScheme: "https")
    configuration!.setURLSchemeHandler(handler, forURLScheme: "http")
}
```

Server 身分認證
(Challenge–response authentication)

URLAuthenticationChallenge delegate.

What: 某些 Server 會對 client 發出身分驗證;
client 需實作 Authentication Challenge
回應否則會被 Server 拒絕訪問。

How: 在 url session delegate 內呼叫
WK navigation delegate method,
將 completion handler 交由後者處理

```swift
extension WebViewURLSchemeHandler: URLSessionTaskDelegate {

    func urlSession(_ session: URLSession, task: URLSessionTask, didReceive challenge:
        URLAuthenticationChallenge, completionHandler: @escaping
        (URLSession.AuthChallengeDisposition, URLCredential?) -> Void){

    }
```

在 url session delegate 內呼叫 WK navigation delegate method,
將 completion handler 交由後者處理

```swift
extension BrowserViewController: WKNavigationDelegate {

    func webView(_ webView: WKWebView, didReceive challenge:
        URLAuthenticationChallenge, completionHandler: @escaping
        (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {
    }
```

Ref: https://developer.apple.com/documentation/foundation/url_loading_system/handling_an_authentication_challenge

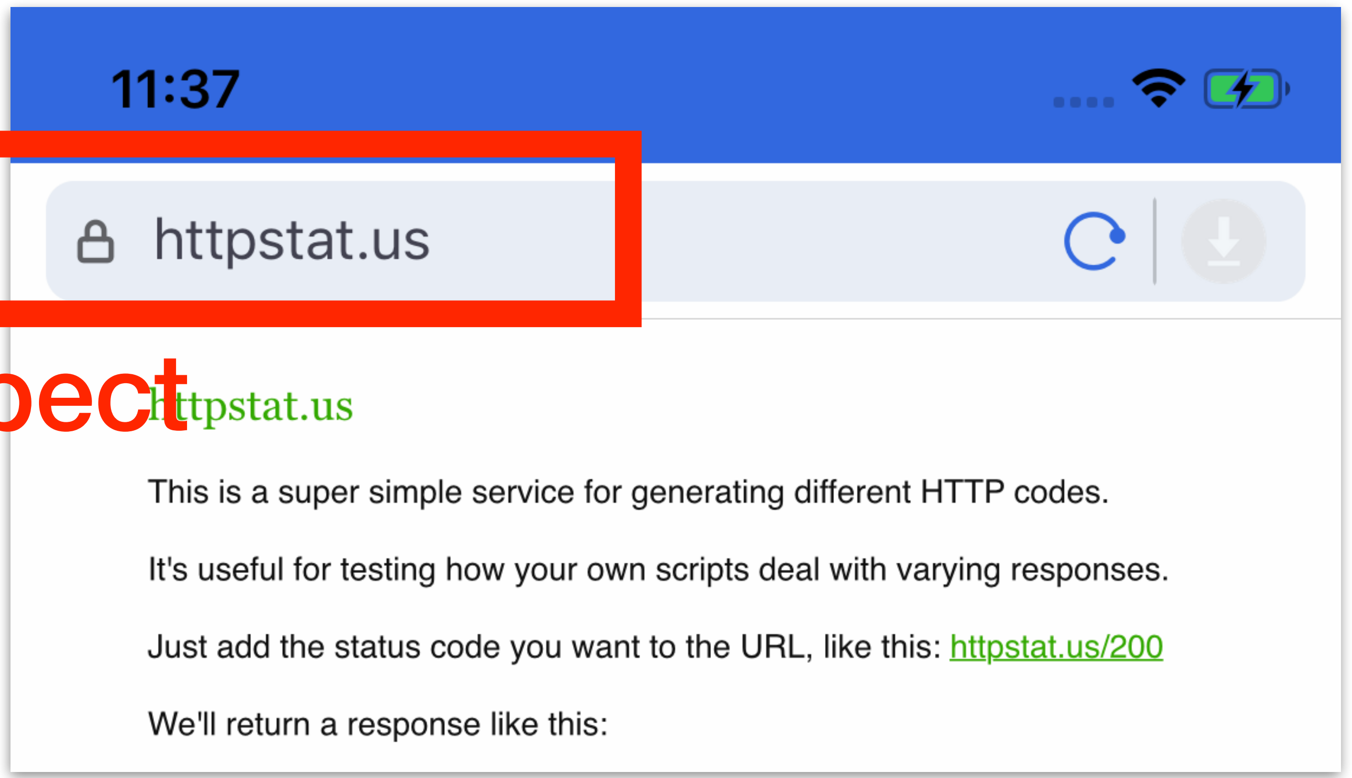
轉址後 WKWebView.url 無對應變動

willPerformHTTPRedirection delegate.

What: 當發生 request 轉址時，webView 預期應將 url 替換為轉址後的 url，但透過 URLSchemeHandler 送回給 webView 的轉址回應會發生 webView url 沒有對應改變成轉址後 url 問題。

How: 在 urlSession willPerformHTTPRedirection delegate 可以攔截到發生轉址的 response 及轉址 request，藉由判斷 response url 是否為主頁面，若否，將讓 urlSession 幫我們拿回轉址後的 response 後回填 urlSchemeTask；若是，讓 webView 直接載入轉址 request 已達到 webView url 對應轉址改變的效果。

Expect


11:37

httpstat.us

httpstat.us

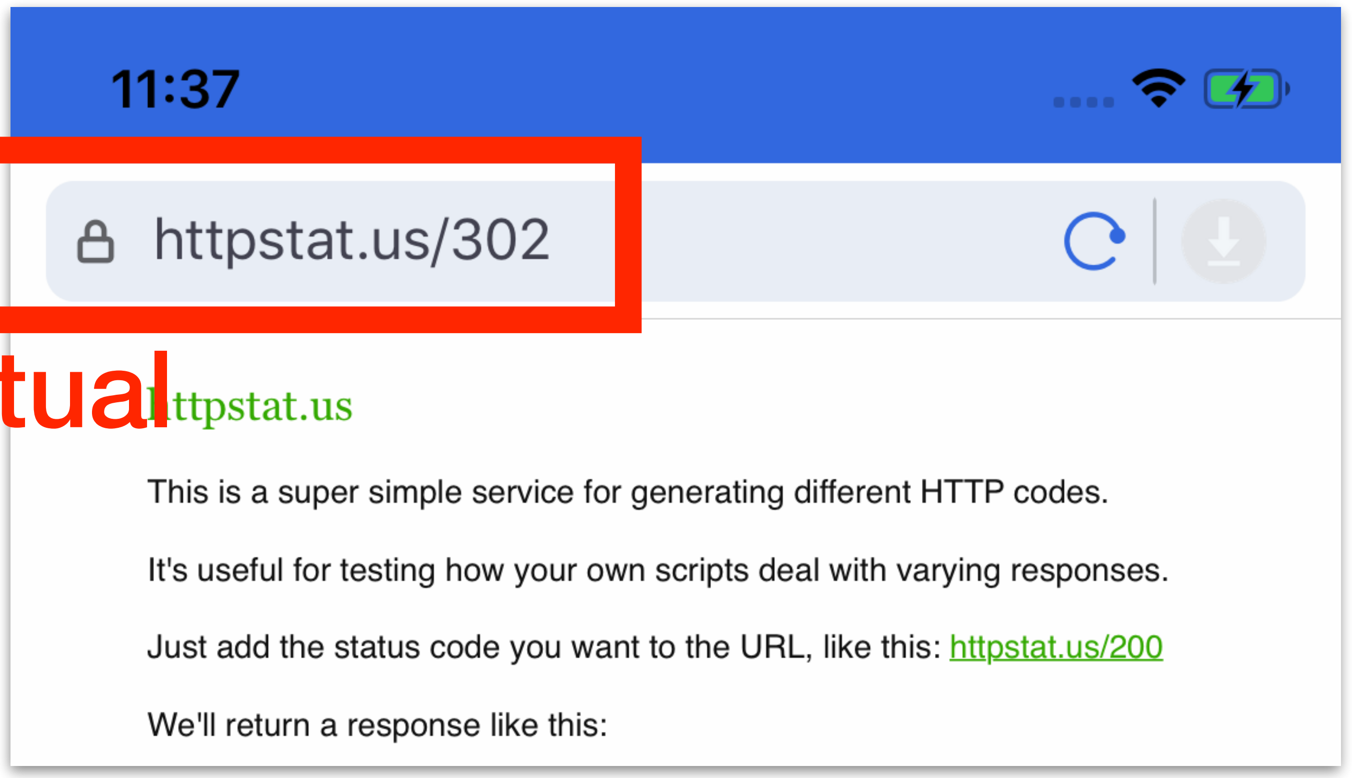
This is a super simple service for generating different HTTP codes.

It's useful for testing how your own scripts deal with varying responses.

Just add the status code you want to the URL, like this: httpstat.us/200

We'll return a response like this:

Actual


11:37

httpstat.us/302

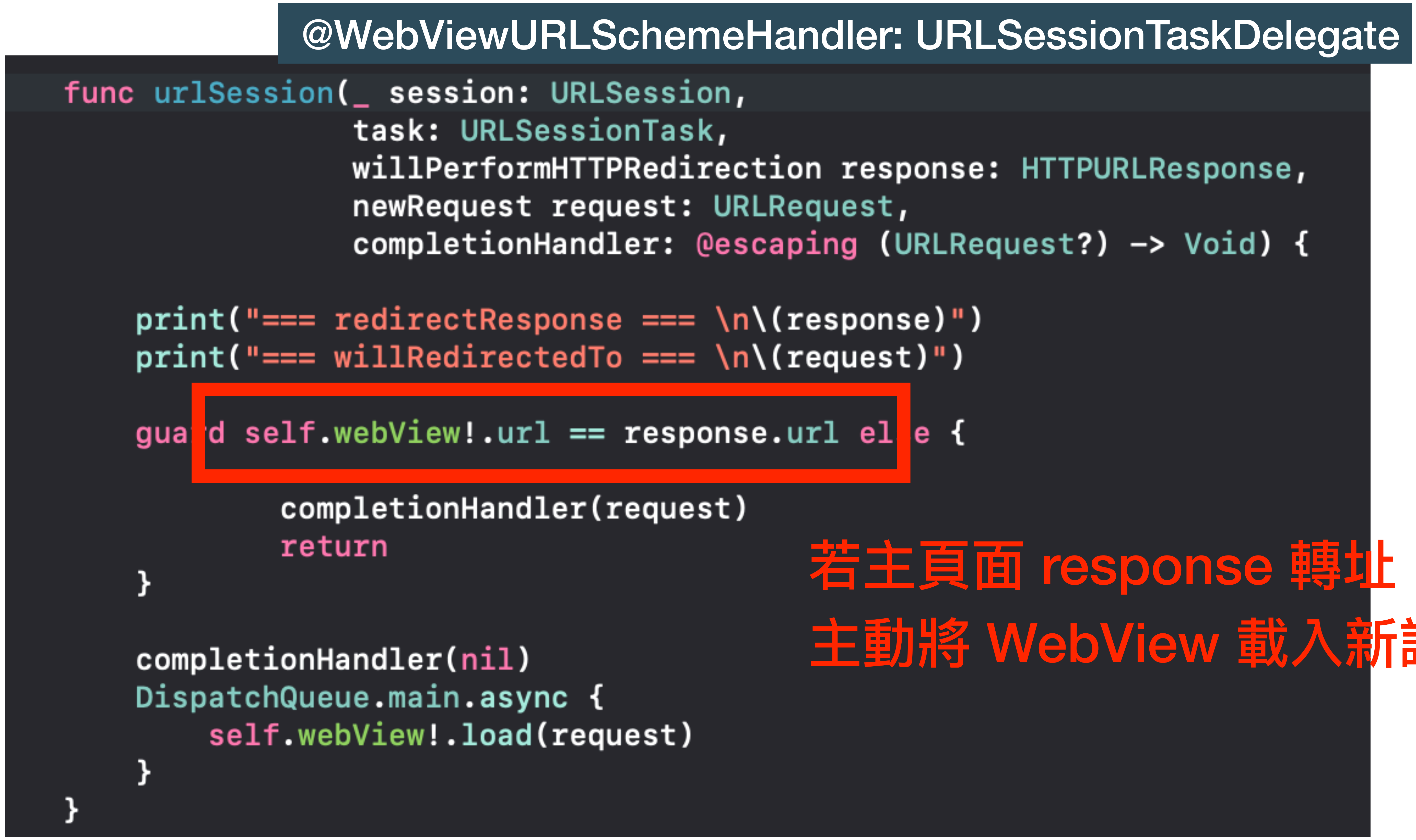
httpstat.us

This is a super simple service for generating different HTTP codes.

It's useful for testing how your own scripts deal with varying responses.

Just add the status code you want to the URL, like this: httpstat.us/200

We'll return a response like this:


@WebViewURLSchemeHandler: URLSessionTaskDelegate

```
func urlSession(_ session: URLSession,
                task: URLSessionTask,
                willPerformHTTPRedirection response: HTTPURLResponse,
                newRequest request: URLRequest,
                completionHandler: @escaping (URLRequest?) -> Void) {

    print("=== redirectResponse === \n\(response)")
    print("=== willRedirectedTo === \n\(request)")

    guard self.webView!.url == response.url else {

            completionHandler(request)
            return
    }

    completionHandler(nil)
    DispatchQueue.main.async {
        self.webView!.load(request)
    }
}
```

若主頁面 response 轉址，
主動將 WebView 載入新請求

Crash on posting large HTTP body

What:某些狀況下在 POST request 時 WebKit 內部會產生錯誤導致 crash， 推測為當 post body 過大（例圖片或大量文字等等）， WebKit 導流至 URLSchemeHandler 的 bug。

How: 根據 stack-overflow 搜尋相關文獻，調用 Private API: WebView.setLoadResourcesSerially 設值為 false 即可解決 crash 現象。

@WebViewURLSchemeHandler. init( )

```
let selector = sel_registerName("_setLoadResourcesSerially:")
if let wv = NSClassFromString("WebView") as? NSObject.Type,
    wv.responds(to: selector) {
    wv.perform(selector, with: false as Any)
}
```

Ref: https://stackoverflow.com/questions/60198551/ios-wkwebview-wkurlschemehandler-crash-on-posting-body-exc-bad-access/63368878#63368878