SCIENCES SORBONNE UNIVERSITÉ

WKWebView

Fabrice.Kordon@lip6.fr

LIP6

As an introduction…

A bit of history

- To display a web page… UIWebView
- Deprecated (since iOS8)
  - **Still available (Still in iOS12)**
  - **But warning telling it will not be supported in the future**
- You must forget it!!!

UIWebView is dead, long live to WKWebView

- Framework located in WebKit
  - **You have to import it**
- Embeds dedicated mechanisms
  - **zoom/scroll, web pages stack, etc.**
- Associated to other protocols
  - **WKNavigationDelegate, WKUIDelegate, etc.**

As an introduction…

A bit of history

By the way…

Important: You should not embed `WKWebView` or `UITableView` objects in `UIScrollView` objects. If you do so, unexpected behavior can result because touch events for the two objects can be mixed up and wrongly handled.

UIWebView is dead, long live to WKWebView

- Framework located in WebKit
  - **You have to import it**
- Embeds dedicated mechanisms
  - **zoom/scroll, web pages stack, etc.**
- Associated to other protocols
  - **WKNavigationDelegate, WKUIDelegate, etc.**

How it works (1/2)

- **Init**
  - Either «à la swift» or «à la Objective-C»
- **Load content**
  - Create an URL
    - **URL/NSURL**
  - Create a request
    - **URLRequest/NSURLRequest**
  - For secure interactions : URLSession/NSURLSession
    - **Supports APS (App Transport Security, iOS9) — RFC 2818**
    - **Creation of a session**
    - **Configuration of a session**
    - **Creation of tasks — use of delegation for their completion**
    - **Not more here**

How it works (2/2)

Navigate in a stack of views

- Each one is a web page
  - **goBack(), goForward()**
- Useful readonly attributes
  - **canGoBack, canForward**

WKNavigationDelegate protocol

- All methods are optional
  - **webView(_:didStartProvisionalNavigation:) / webView:didStartProvisionalNavigation:**
  - **webView(_:didFailProvisionalNavigation:withError:) / webView:didFailProvisionalNavigation:withError:**
  - **webView(_:didFinish:) / webView:didFinishNavigation:**
  - **webView(_:didReceiveServerRedirectForProvisionalNavigation:) / webView:didReceiveServerRedirectForProvisionalNavigation:**

How it works (2/2)

Navigate in a stack of views

- Each one is a web page
  - **goBack(), goForward()**
- Useful readonly attributes
  - **canGoBack, canForward**

WKNavigationDelegate protocol

- All methods are optional
  - **webView(_:didStartProvisionalNavigation:) / webView:didStartProvisionalNavig**
  - **webView(_:didFailProvisionalNavig webView:didFailProvisionalN tion:wit**
  - **webView(_:didFinish:) / w inis Navigation:**
  - **webView(_:didReceiveS ovisionalNavigation:) / webView:didReceiveSe visionalNavigation:**

(NS)URL & (NS)URLRequest

URL - elaboration of an URL

- init(string:) / initWithString:
- init(fileURLWithPath:) / initFileURLWithPath:

URLRequest, query based on an URL

- Built based on a URL

Some properties

- Configuration of the connection
  - **Cache policy**
  - **Timeout interval**
- Network type
  - **allowsCellularAccess**
- Manipulation
  - **absoluteString, host, baseURL, etc.**

(NS)URL & (NS)URLRequest

URL - elaboration of an URL

- init(string:) / initWithString:
- init(fileURLWithPath:) / initFileURLWithPath:

URLRequest, query based on an URL

- Built based on a URL

Some properties

- Configuration of the connectio
  - **Cache policy**
  - **Timeout interval**
- Network type
  - **allowsCellularAccess**
- Manipulation
  - **absoluteString, host, baseURL, etc.**

Handling errors

Thank to WKNavigationDelegate

- Methods to catch failures

Error / NSError

- localizedDescription
  - **readonly attribute**
- More for NSError
  - **code, localizedFailureReason, etc.**

Code injection in WKWebView

The WKUserScript class

- Object to be injected in a web page

The WKUserContentController class

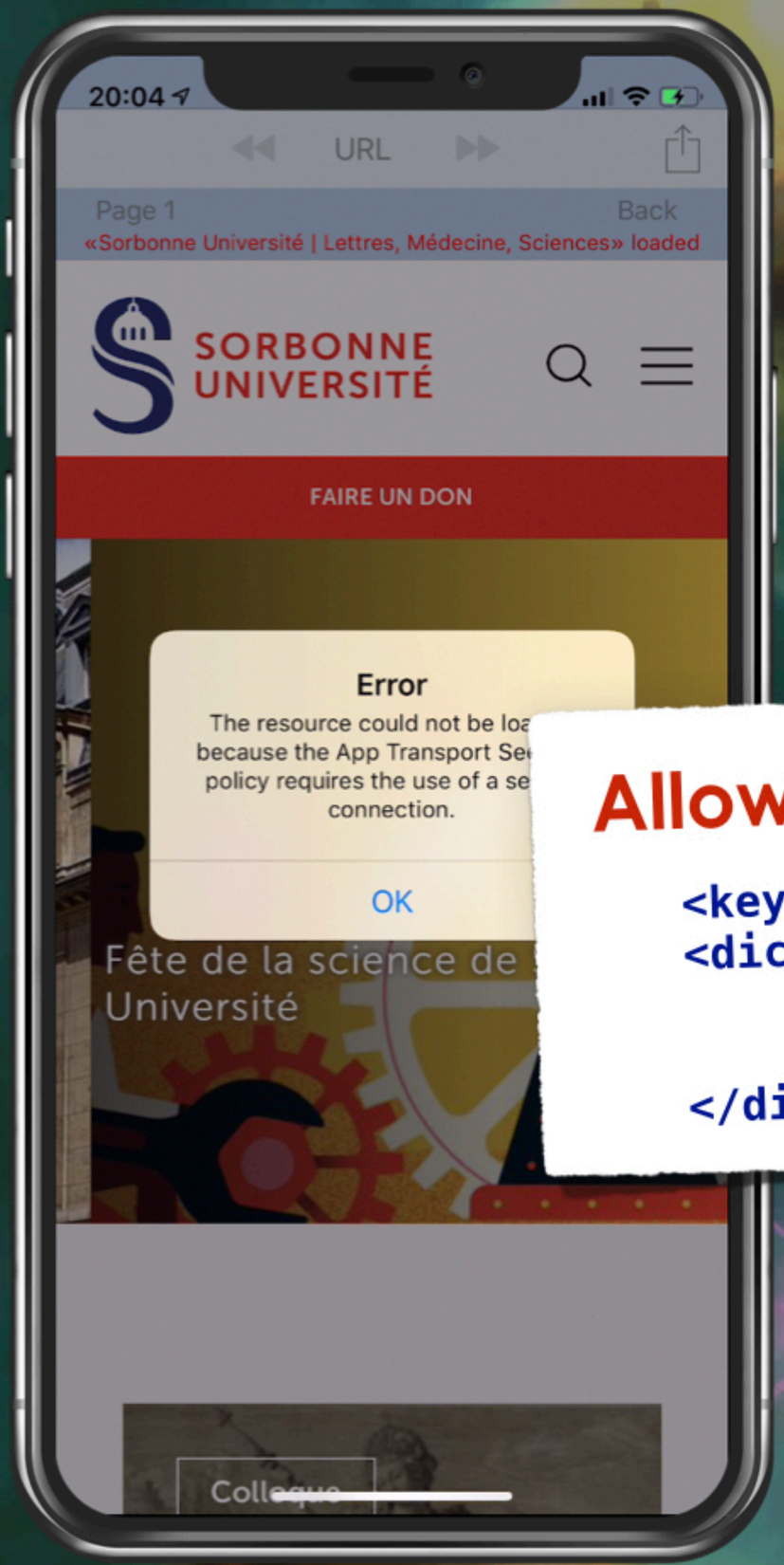
- To inject code (e.g. JavaScript)
- Inside a WKWebViewConfiguration
  - **Then passed to he WKWebView**
- A small example (from R. Kim)

```
let contentController = WKUserContentController()
let scriptSource = "document.body.style.backgroundColor = `red`;"
let script = WKUserScript(source: scriptSource,
                          injectionTime: .atDocumentEnd,
                          forMainFrameOnly: true)
contentController.addUserScript(script)

let config = WKWebViewConfiguration()
config.userContentController = contentController

let webView = WKWebView(frame: .zero, configuration: config)
```

Security concerns



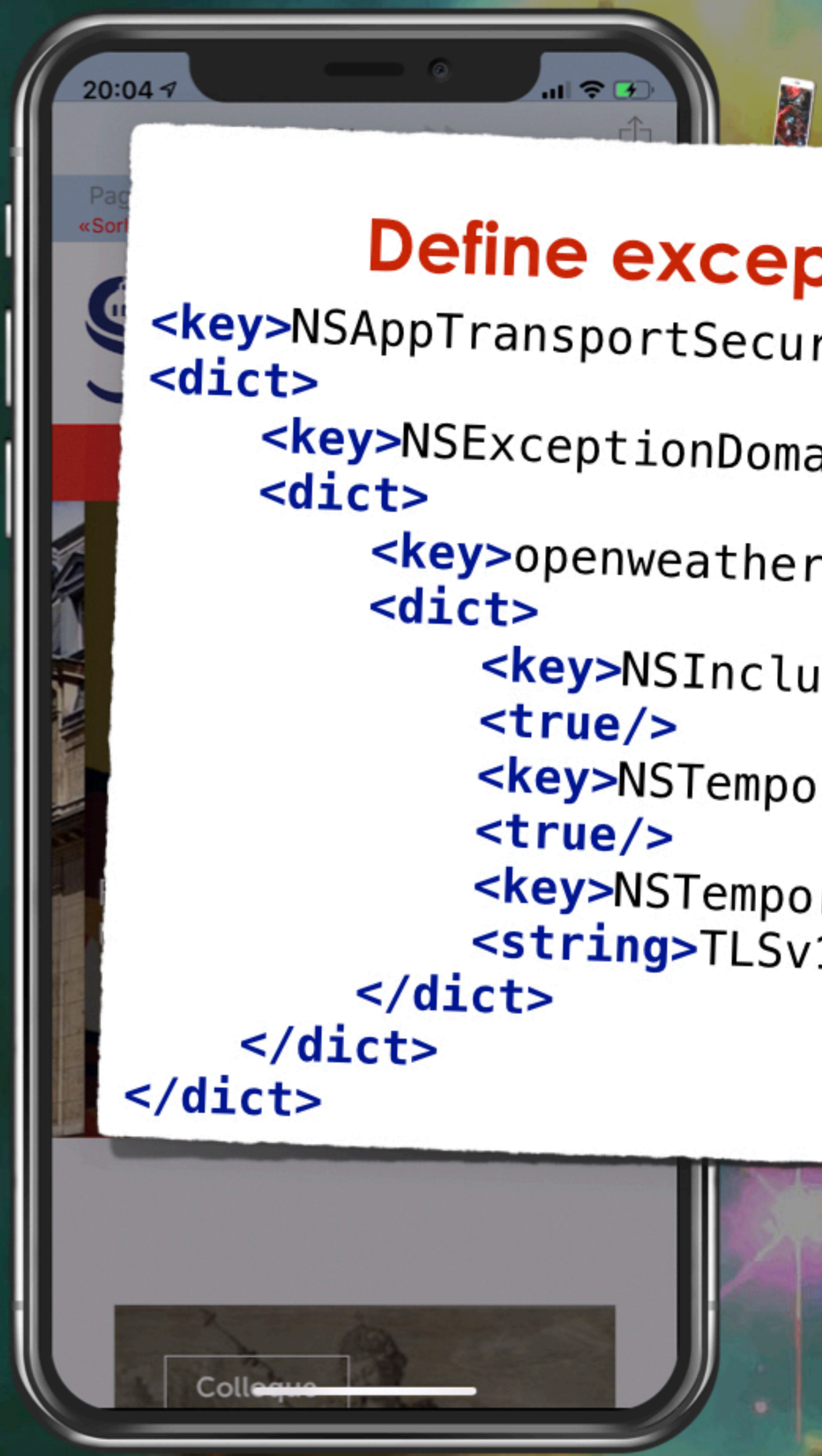
About Insecure connections

- Not supported «by default»
  - **https: instead of http:**
- Need to change configuration
  - **NSAppTransportSecurity**
  - **In he info.plist file**

Allow all insecure connections

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

Security concerns



About Insecure connections

Define exception for insecure connections

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
        <key>openweathermap.org</key>
        <dict>
            <key>NSIncludesSubdomains</key>
            <true/>
            <key>NSTemporaryExceptionAllowsInsecureHTTPLoads</key>
            <true/>
            <key>NSTemporaryExceptionMinimumTLSVersion</key>
            <string>TLSv1.1</string>
        </dict>
    </dict>
</dict>
```

Security concerns

About Insecure connections

Define exception for insecure connections

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NS tionDomains</key>
    <dict>
        <
        <
                                                    TPLoads</key>
            <key>NSTemporaryExceptionMinimumTLSVersion</key>
            <string>TLSv1.1</string>
        </dict>
    </dict>
</dict>
```

Impact on network queries too

You may process similarly

WKWebViews, JavaScript & popups

Be careful with JavaScript features

- Display windows not allowed by default
- You may also control some aspects of Javascript

WKUIDelegate protocol

- To present «native user interface elements» on behalf of a web page
  - **webView(_:runJavaScriptAlertPanelWithMessage: initiatedByFrame:completionHandler:)**
  - **etc.**

WKWebViews, JavaScript & popups

Be careful with JavaScript features

- Display windows not allowed by default
- You may also control some aspects of Javascript

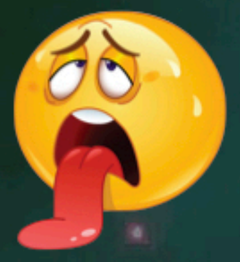
WKUIDelegate protocol

- To present «native user interface elements» on behalf of a web page
  - webView(_:runJavaScriptAlertPanelWithMessage: initiatedByFrame:completionHandl
  - etc.

As a conclusion…

Access to online content is easy

- With possible protections
  - **Painful from the developer's point of view**



Embed structured text in you Apps

- No excuse to avoid documentation
  - **In-App**
- No excuse to reach updated documentation
  - **Though an external link**

All is included

- Zooming, scrolling, handling font size / etc.