Porting iOS apps to Android with GCC and GNUstep

Not Your Father’s Web View

Who’s that guy?

ADHD

Photo by William Sutherland, public domain

M. le Dr GILLES DE LA TOURETTE,
Médecin des Hôpitaux de Paris, directeur en chef
du service médical de l'Exposition de 1900.
Cliché E. Pirou.

Tourette Syndrome

Artist unknown, copyright likely expired

I don’t let that define me

- Proficient iOS developer, beginning with iOS 3.0β
- Conference attendee (CocoaConf x4, WWDC Scholarship x2, MadisonRuby x2, SnowMobile, UXMad)
- Conference presenter (CocoaConf Columbus 2014)
- Interested in people’s rights, both as content creators and individuals

Not Your Father’s Web View

UIWebView

- Introduced in original iPhone SDK
- Executes web content in application's process
- Does not use Nitro JavaScript engine

My cat and my CAT-6

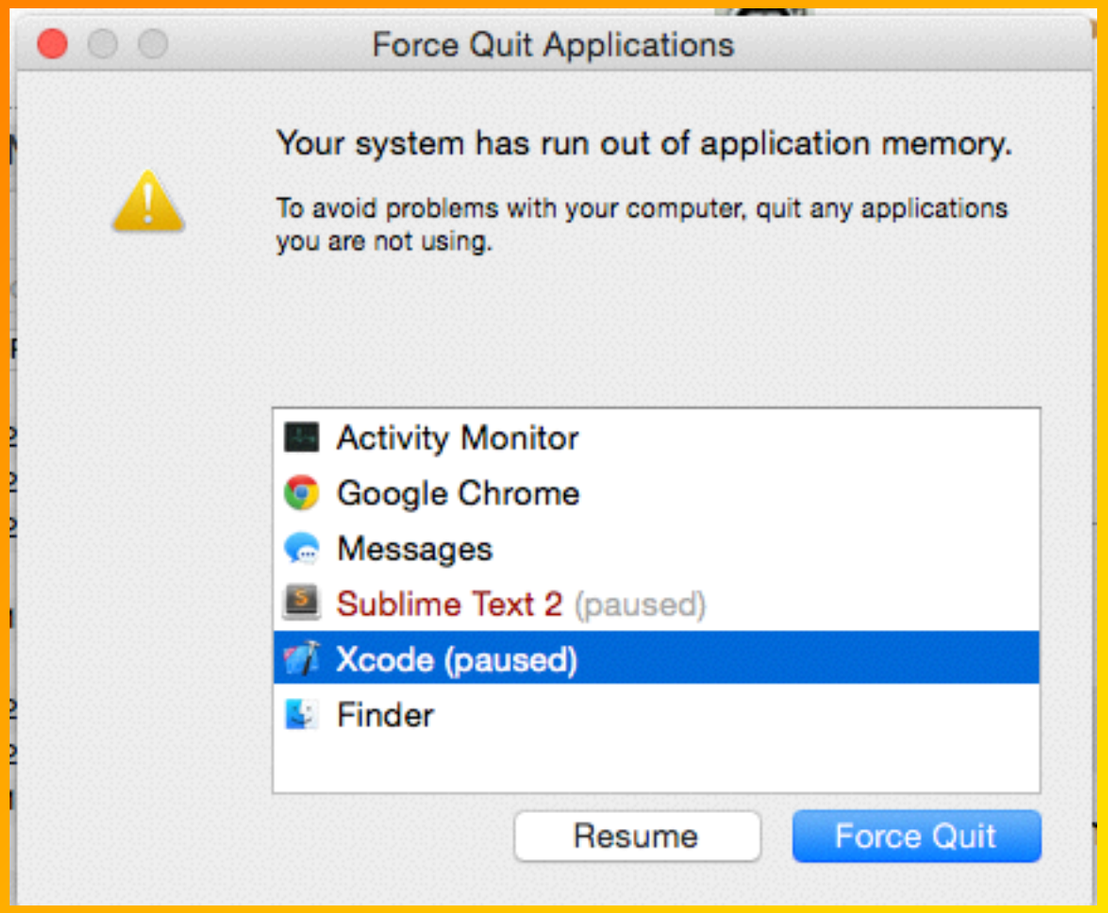
Photo by me, CC0

UIWebView’s replacement

Still in beta!

Process Name	Memory
com.apple.dt.Xcode.Playgr…	32.03 GB
SourceKitService	31.88 GB
Xcode (Not Responding)	522.3 MB

OVER 64 GIGABYTES!

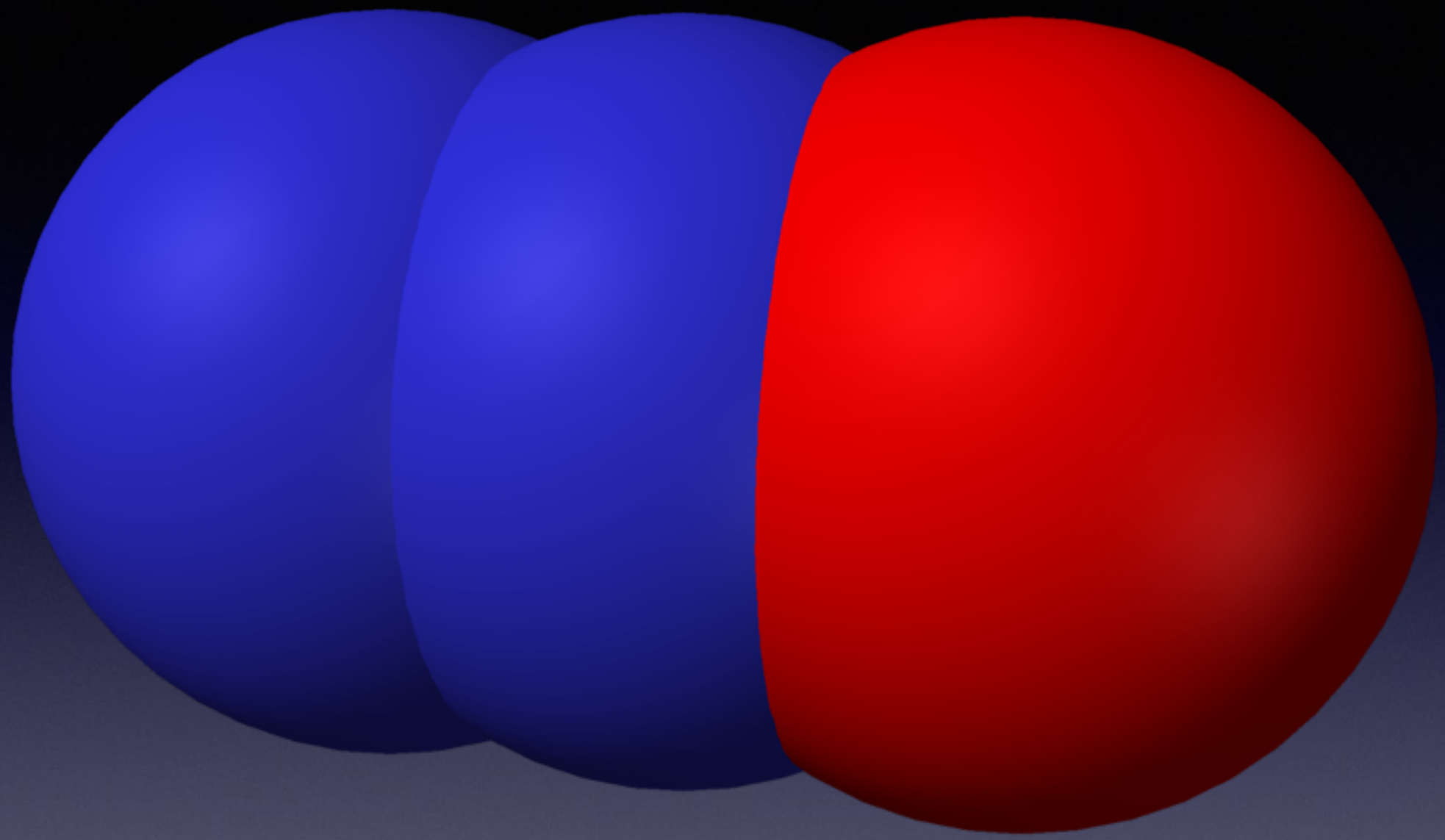


WebKit

- Open source browser engine
- Forked from KHTML and KJS
- Safari and MobileSafari
- Google Chrome
- Kindle web browser

WebKit™ - Apple Inc

Squirrelfish



Nitrous Oxide

Image by Ben Mills, public domain

JavaScriptCore

- WebKit's JavaScript engine
- Implementation of ECMA-262
- Also known as Squirrelfish or Nitro

WebKit.framework

- Introduced in iOS 8
- Replaces UIWebView (iOS) and WebView (Mac)

WKWebView

- Executes web content outside of application's process
- Execution can be paused when web views are not visible on screen - managed internal to the framework
- JavaScript execution is sandboxed and accelerated by Nitro engine

Why should I switch?

- By utilizing the latest JavaScriptCore advances, WebKit is significantly faster than UIWebView
- Sunspider executes in 4102ms in UIWebView, 980ms in WKWebView on iPhone 5 (beta 3)

Why should I upgrade?

- JavaScript memory leaks won’t crash your application
- Web content can be stopped to save CPU cycles
- You’ll take advance of future enhancements to WebKit

Making the switch

- Drop-in replacement
- Most APIs remain the same
- A few APIs moved to WKWebViewConfiguration or WKPreferences

New Features!

New WebKit Features

- Revised navigation API
- User scripts
- Script messaging

Revised navigation API

UIWebView Navigation

- `var canGoBack: Bool { get }`
- `var canGoForward: Bool { get }`
- `func goBack()`
- `func goForward()`

WKWebView Navigation

- `var canGoBack: Bool { get }`
- `var canGoForward: Bool { get }`
- `func goBack() -> WKNavigation!`
- `func goForward() -> WKNavigation!`

WKNavigation

- `var initialRequest: NSURLRequest { get }`
- `var request: NSURLRequest! { get }`
- `var response: NSURLResponse! { get }`
- `var error: NSError! { get }`

WKWebView Navigation

- `var allowsBackForwardNavigationGestures: Bool`
- `var backForwardList: WKBackForwardList! { get }`
- `func goToBackForwardListItem(`
  `_ item: WKBackForwardListItem!) -> WKNavigation`

Navigation Policy

- ```
  enum WKNavigationActionPolicy : Int {
      case Cancel
      case Allow
  }
  ```

- ```
  @optional func webView(_ webView: WKWebView!,
      decidePolicyForNavigationAction navigationAction: WKNavigationAction!,
      decisionHandler decisionHandler: ((WKNavigationActionPolicy) -> Void)!)
  ```

- ```
  @optional func webView(_ webView: WKWebView!,
      decidePolicyForNavigationResponse navigationResponse: WKNavigationResponse!,
      decisionHandler decisionHandler: ((WKNavigationResponsePolicy) -> Void)!)
  ```
  ```

User Scripts

User scripts

- Arbitrary JavaScript can be injected into web pages to modify their behavior
- Scripts can include functions or full frameworks, or be simple calls to existing functionality
- Replaces `stringByEvaluatingJavaScriptFromString:` for calling into JavaScript context from native code
- Scripts can't return objects directly anymore (web views are out-of-process). Use script messaging.

WKUserScript

```
class WKUserScript {
  init(source: String!,
    injectionTime: WKUserScriptInjectionTime,
    forMainFrameOnly: Bool)
}
```

Script execution timing

```
enum WKUserScriptInjectionTime {
  // Inject the script after the
  // document element has been created,
  // but before any other content has
  // been loaded.
  case AtDocumentStart

  // Inject the script after the
  // document has finished loading,
  // but before any subresources may
  // have finished loading.
  case AtDocumentEnd
}
```

Script Loading

```
let scriptPath = NSBundle.mainBundle().pathForResource(
    “myScript",
    ofType: “.js”)

let scriptSource = NSString.stringWithContentsOfFile(
    scriptPath,
    encoding: NSASCIIStringEncoding,
    error: nil)

let script = WKUserScript(
    source: scriptSource,
    injectionTime: .AtDocumentStart,
    forMainFrameOnly: true)
```

Script Injection

```
let webview: WKWebView = /* .. */;

let userscript: WKUserScript = /* .. */;

webview.configuration
  .userContentController
  .addUserScript(userscript)
```

Script Messaging

Script Messaging

- Native code implements `WKScriptMessageHandler` protocol
- JavaScript calls handler on `document.webkit` object.
- Combined with user scripts, script messaging provides two-way communication between JavaScript and native code

Script Message Handling

```
protocol WKScriptMessageHandler {

  func userContentController(
     userContentController: WKUserContentController!,
     didReceiveMessage message: WKScriptMessage!)

}
```

Script Message Registration

```
let webview: WKWebView = /* .. */;

webview.configuration.userContentController
  .addScriptMessageHandler(self, "handlerName")
```

Hijacking console.log

```
(function() {
  var defaultLog = console.log;
  console.log = function(message) {
    webkit.messageHandlers.consoleLog.postMessage(message);
    defaultLog(message);
  };
})();

console.log("testing console.log redirection”);
```

Hijacking console.log

```
class MyClass : MySuperclass, WKScriptMessageHandler {
  var webview: WKWebView?

  func hijackConsoleLog() {
    webview!.configuration.userContentController
      .addScriptMessageHandler(self, name: “consoleLog”);
  }

  func userContentController(
     userContentController: WKUserContentController!
     didReceiveScriptMessage message: WKScriptMessage!) {
    println("scriptMessage: \(message.body)”);
  }
}
```

Live demo!

QQ’s!

Further resources

- github.com/wjlafrance/cocoaconf
- webkit.org
- WKWebView Class Reference and headers
- WWDC 2014 Videos
  - 206 - Introducing the Modern WebKit API
  - 506 - Your App, Your Website, and Safari

Thanks for coming.

App.net: @wjl

Twitter: @wjlafrance

Email: wjlafrance@gmail.com

Blog: wjlafrance.net