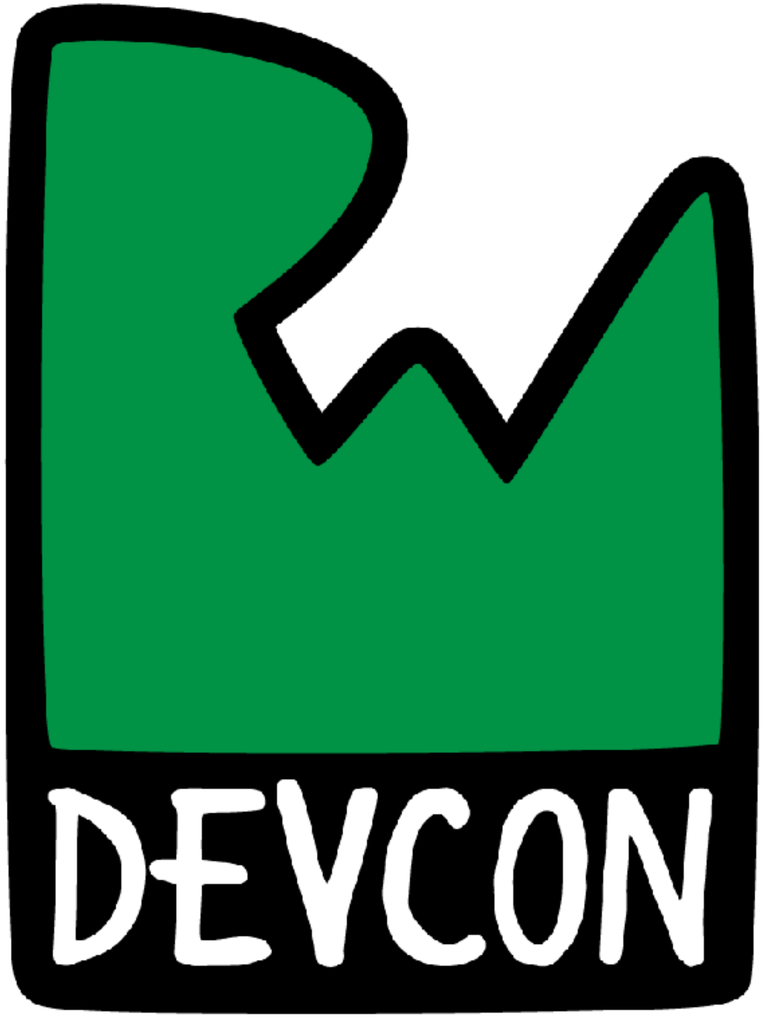
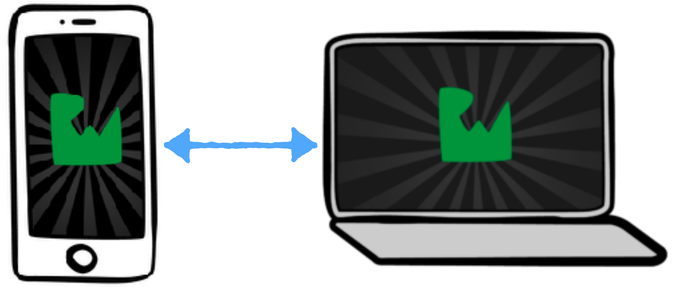
Session 11: Advanced WKWebView



WKWebView 101: Why & How

Demo Mechanics

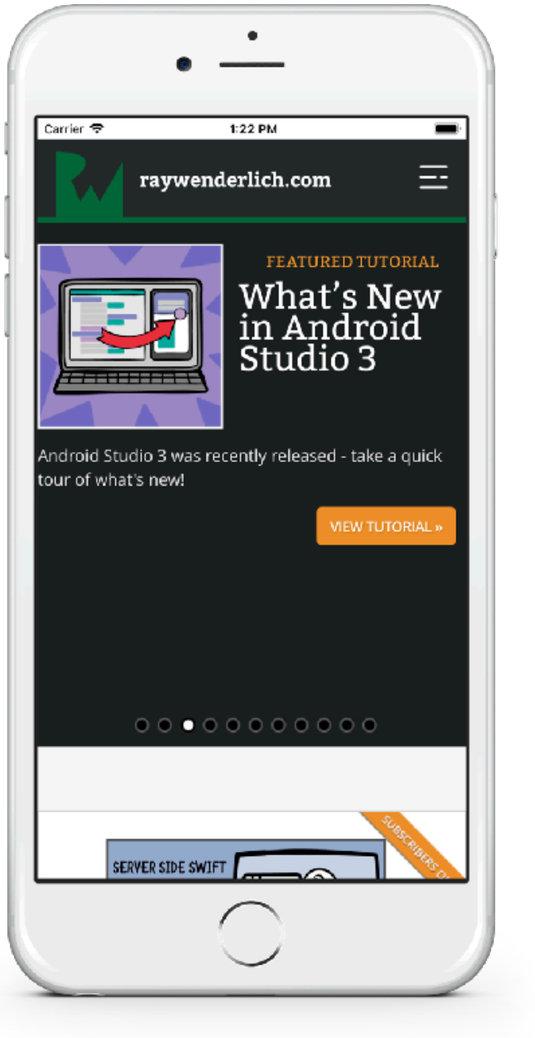
- Deskbook: An Employee Directory Server
  - NodeJS Server: json / html / css
  - App: Connects to the Deskbook API
  - We'll use Simulator today



- Depenency: NodeJS
  - node-v.8.9.0.pkg included in the files for this tutorial
  - Install this to follow along the Demo
  - Nothing else needed and no internet is required

WebKit Web View 101

- WKWebView (aka WebKit Web View) replaces UIWebView
- Renders HTML/CSS/JS
  - Very efficient, fast and secure
- WKWebView:
  - Not a browser
  - Not SFSafariViewController



WebKit Web View 101

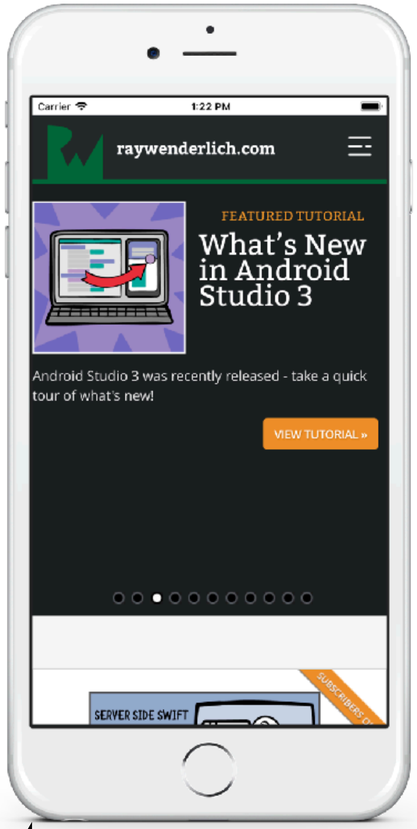
- WebKit finally 1st class citizen (XCode 9)
  - Previously was code only
  - iOS 10: Still code-only or your app will crash!
- High Performance & Secure
  - Uses the Nitro JIT JavaScript engine
  - Process Isolation
- Debug with Safari Developer Tools

Important

Starting in iOS 8.0 and OS X 10.10, use WKWebView to add web content to your app. Do not use UIWebView or WebView.

WebKit Web View 101

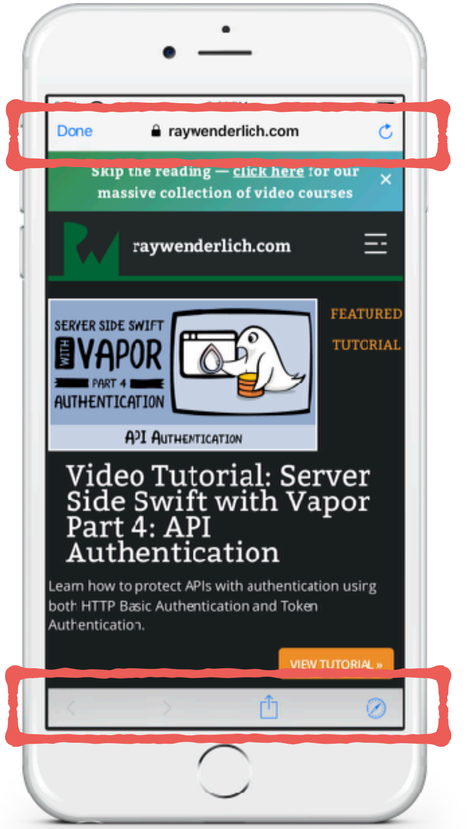
WebKit



- No Chrome
  No Controls
- Full control of the content (all phases)
- Can take any portion of the screen (standard view)
- Could be seamless with other content

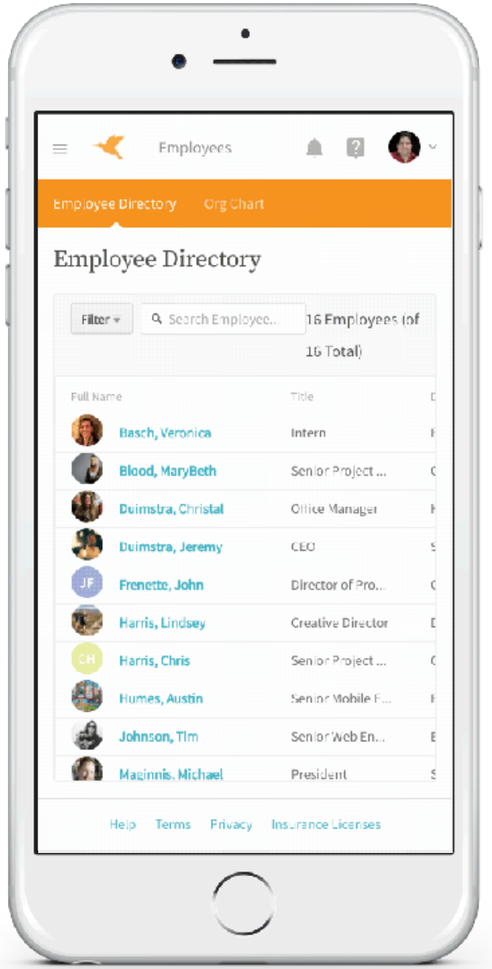
Safari View Controller

- Address Bar
  Controls
- No control of the content (after load)
- Full Screen Modal (not a standard view)
- Very obvious and different



Why WebKit Web Views?

- Shorten development
  - Reuse layout and content
  - No work when layout **might change often**
- Benefit:
  - Potentially avoid an AppStore review!
- Examples:
  - Terms & Conditions, Product Information, Profile Pages, Galleries



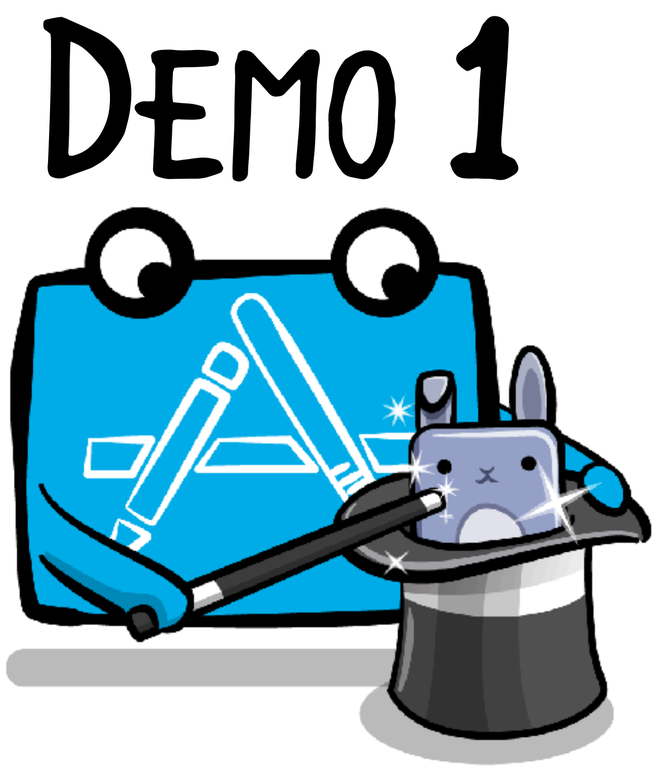
What this WebKit talk is not!

- No 100% Web Views Apps
  - PhoneGap / Cordova / Others
- Instead, we use Web Views to supplement Swift
  - Save time
  - Re-use content
  - Cross-platform layouts!

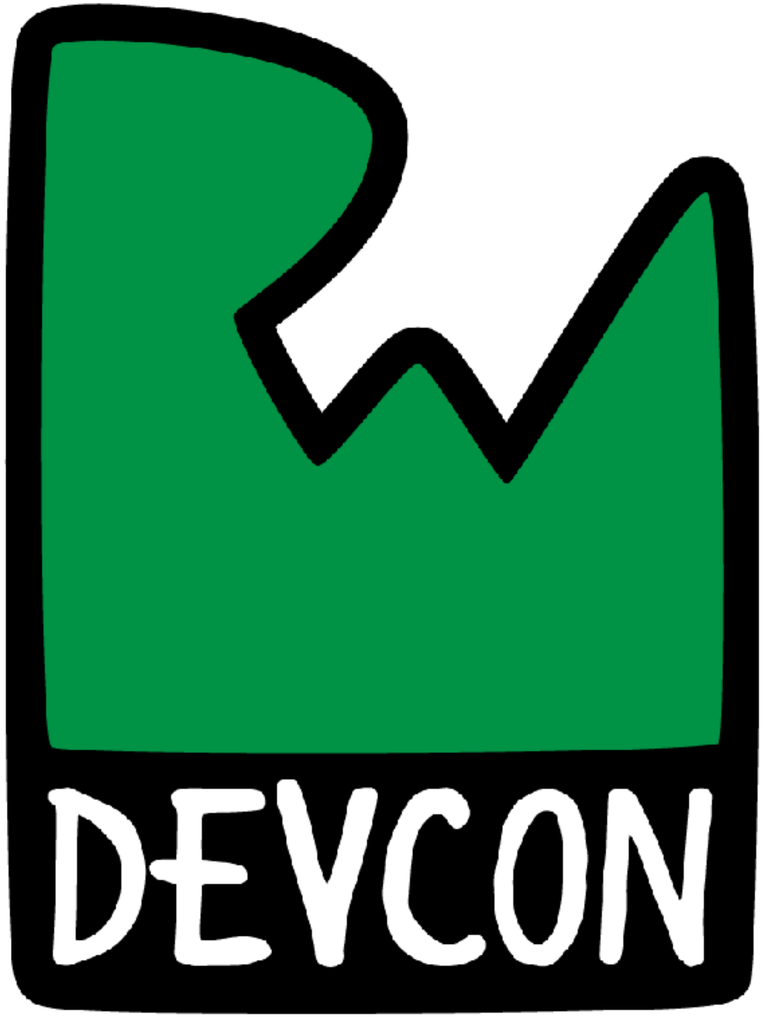
Demo Mechanics

- Deskbook: An Employee Directory
  - We'll discuss how to start it during Demo1
- Follow with Simulator (easiest)
- No Xcode Beta Please!

DEMO 1



Session 11: Advanced WKWebView



HTML & Links

HTML on Mobile (Refresher)

- Normal HTML:

```
<head>
  <title>Awesome Page</title>
  <meta ... />
  <script>...</script>
  <style>...</style>
</head>
<body>
  HTML here…
</body>
```

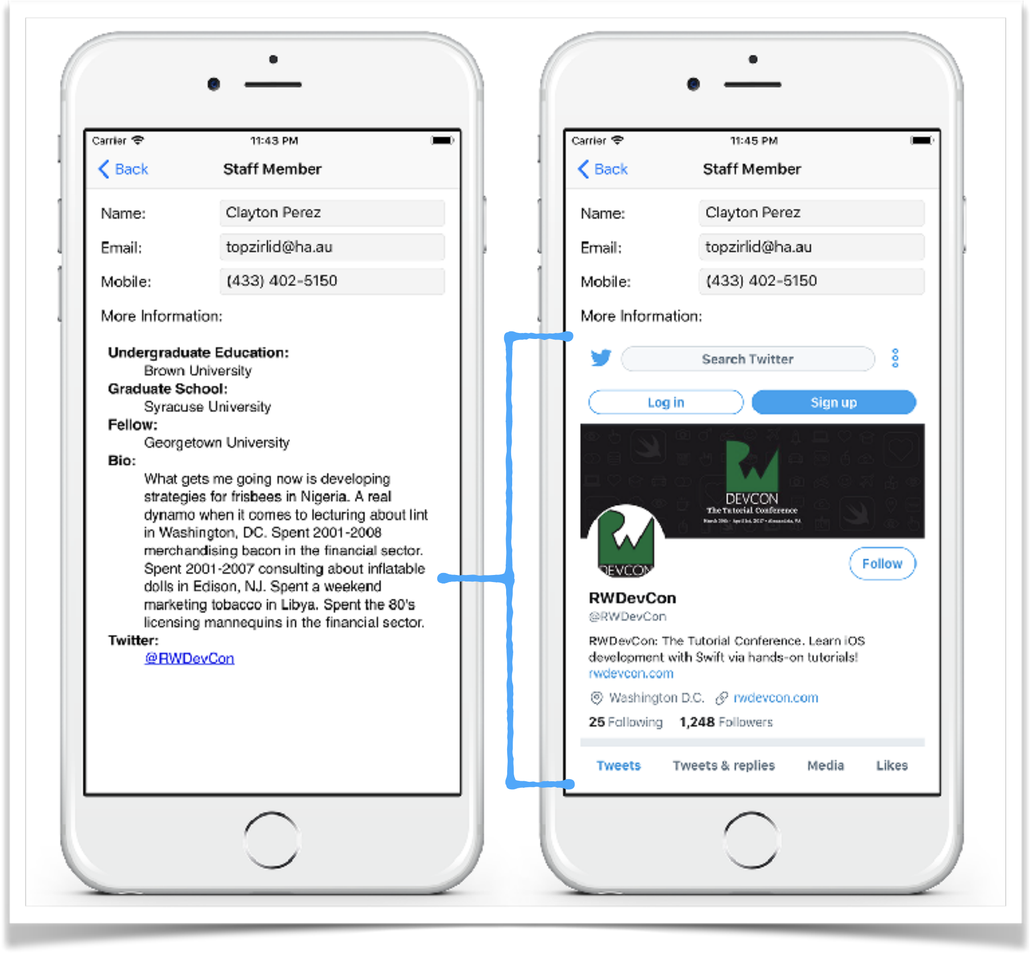
HTML on Mobile (Refresher)

- META VIEWPORT:
  - `<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">`
- Learn More:
  - Apple: https://developer.apple.com/library/content/documentation/AppleApplications/Reference/SafariWebContent/UsingtheViewport/UsingtheViewport.html
  - Generic: https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

Wrangling HTML

- Displaying HTML is useful
  - WKWebView acts like mini browser (albeit captive)
- As you saw, links work!
  - ✅ If you have full Control, might not be so bad.
  - ❌ If you have no control, you have a problem!

Bad UX Alert


Carrier
11:43 PM
Back
Staff Member
Name:
Clayton Perez
Email:
topzirlid@ha.au
Mobile:
(433) 402-5150
More Information:
Undergraduate Education:
Brown University
Graduate School:
Syracuse University
Fellow:
Georgetown University
Bio:
What gets me going now is developing strategies for frisbees in Nigeria. A real dynamo when it comes to lecturing about lint in Washington, DC. Spent 2001-2008 merchandising bacon in the financial sector. Spent 2001-2007 consulting about inflatable dolls in Edison, NJ. Spent a weekend marketing tobacco in Libya. Spent the 80's licensing mannequins in the financial sector.
Twitter:
@RWDevCon
Carrier
11:45 PM
Back
Staff Member
Name:
Clayton Perez
Email:
topzirlid@ha.au
Mobile:
(433) 402-5150
More Information:
Search Twitter
Log in
Sign up
DEVCON
Follow
RWDevCon
@RWDevCon
RWDevCon: The Tutorial Conference. Learn iOS development with Swift via hands-on tutorials!
rwdevcon.com
Washington D.C.
rwdevcon.com
25 Following
1,248 Followers
Tweets
Tweets & replies
Media
Likes

What if we could control links?

- We can!
- How: Implement a protocol using a delegate
- Navigation Delegate gets to allow/disallow all links:
  - ✅ Based on URL (we'll implement this one)
  - ❌ Based on Response

Navigation is more than links!

- Actually, the navigation delegate gets to allow/disallow all *navigation*:
  - WKWebView first loads with `about:blank`, which we'll explicitly allow
  - If we don't, the WKWebView won't load
  - IMPORTANT:
    - Allow/Disallow or Crash!
    - Allow/Disallow ONCE or Crash!

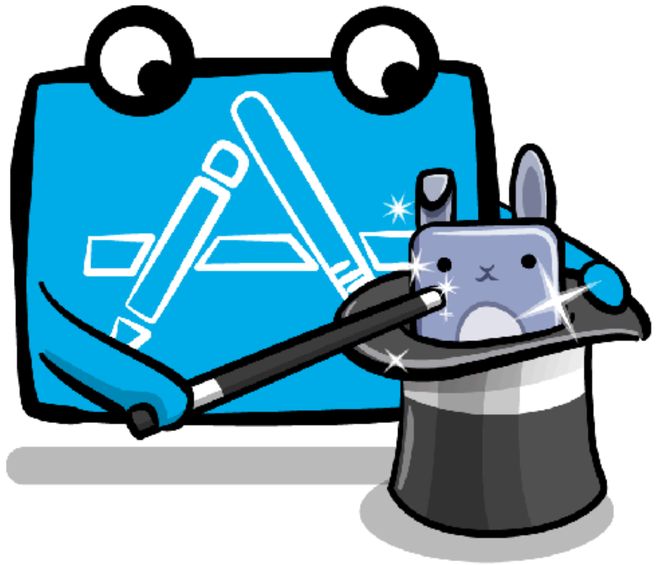
What if we could control links?

- More than just allow/disallow!
- In Deskbook:
  - Analyze the link target URL
  - Twitter? Launch a full browser SFSafariViewController
  - Conveniently, SFSafariViewController shows a full view plus other convenient UX elements

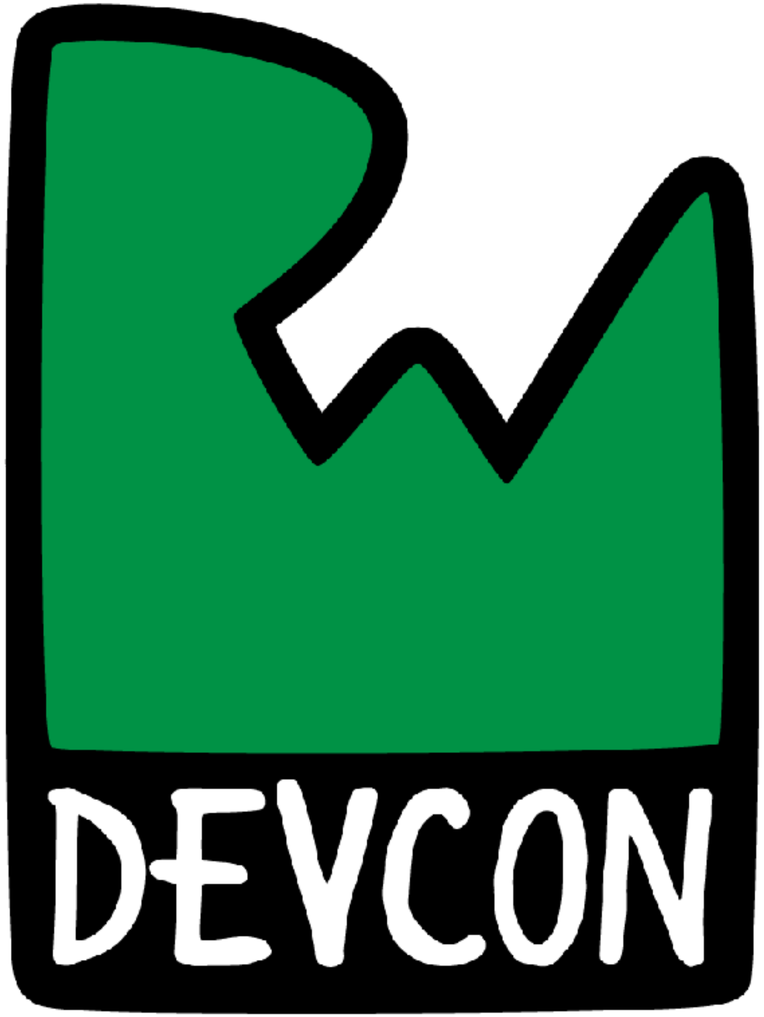
Demo Mechanics

- Deskbook: An Employee Directory
  - Run just like you did before. Likely still running.
- Follow with Simulator (easiest)
- No XCode Beta Please!

DEMO 2



Session 11: Advanced WKWebView



Using files in the Bundle

Power in the Bundle!

- Can include html, css, images in the bundle!
- Typical use cases:
  - styles unique to the mobile app
  - custom fonts
  - images
  - javascript

Previously

```
let wrappedHtml = """
  <head>
    <title>Deskbook WebView</title>
    <meta name="viewport" content="width=device-width,
      initial-scale=1, shrink-to-fit=no">
    \(getCss())
  </head>
  <body>
    <div class="\(wrapClass)">\(html)</div>
  </body>
"""
```

Bundle Access: Better Way

```
self.bioWebView.loadHTMLString("HTML-HERE", baseURL:
  Bundle.main.bundleURL)
```

- Example:
  - ‘https://somesite.com/style.css
    - does not require basepath
  - ‘style.css’
    - Does

Bundle Access: Better Way

```
let wrappedHtml = """
  <head>
    <title>Deskbook WebView</title>
    <meta name="viewport" content="width=device-width,
      initial-scale=1, shrink-to-fit=no">
    <link href="style.css" rel="stylesheet" type="text/css">
  </head>
  <body>
    <div class="\(wrapClass)">\(html)</div>
  </body>
"""
```

Custom Fonts, iOS

- iOS Supports Custom Fonts
- Support:
  - True Type Fonts (.ttf), Open Type Fonts (.otf)
  - Files included in the bundle
  - Declared in **info.plist**
- More information
  - https://developer.apple.com/documentation/uikit/text_display_and_fonts/adding_a_custom_font_to_your_app

Custom Fonts, WKWebView HTML

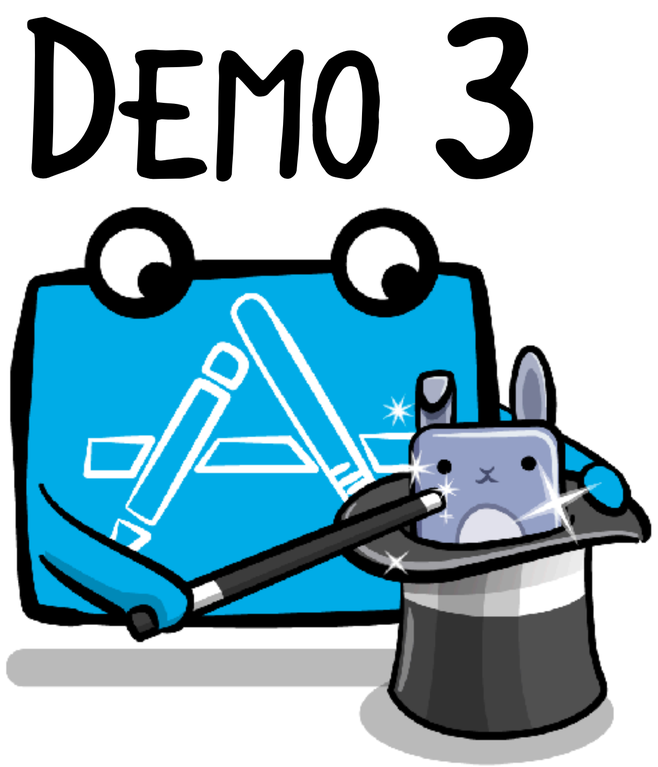
- Supported in CSS with @font-face
  - Will look at this during demo
- PostScript Name is css “font-family:”
  - Font Book shows this
- style.css

```
@font-face {
  font-family: Neuropol;
  src: url('NEUROPOL.ttf');
}
```

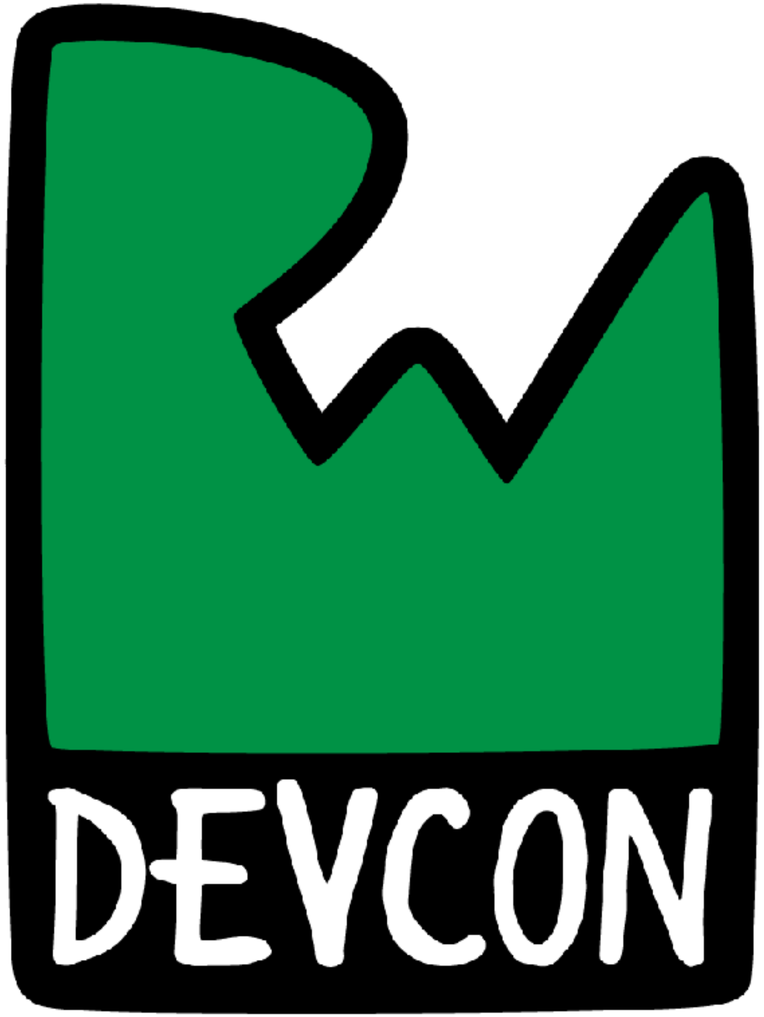
Demo Mechanics

- Deskbook: An Employee Directory
  - Run just like you did before. Likely still running.
- Follow with Simulator (easiest)
- No XCode Beta Please!

DEMO 3



Session 11: Advanced WKWebView



Conclusion

What You Learned

- **Demo 1**: WKWebView 101: Why & How
  - Html Structure & Viewport
- **Demo 2**: Intercepting Links & Navigation
  - Control the experience & everything that loads
- **Demo 3**: Using files in the bundle
  - Styles, fonts, images (even javascript)

Where To Go From Here?

- Apple's WKWebView Documentation
- WWDC 2017 "Customized Loading in WKWebView"
  - Cookie Management & Observation,
  - Content Rules & More
- Learn about JavaScript in WKWebView
  - Apple's Doc: evaluateJavaScript
- Twitter: @ericwastaken