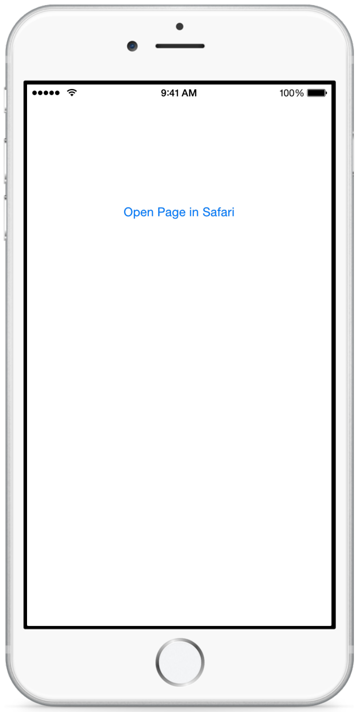
EasyBrowser
Lesson 1

Description

Add a button to the interface that causes Safari to enter the foreground.



Learning Outcomes

- Practice using Interface Builder to add user interface elements to a view.
- Practice establishing IBAction connections between a view and its controller.
- Define URL, and relate URLs to the `NSURL` class.
- Discover the `UIApplication` class, and describe how it represents the app.
- Discover how to use iOS URL schemes to bring other apps, such as Safari, to the foreground.

Vocabulary

`@IBAction`	URL	`NSURL`
`UIApplication`	optional binding	`if let`

Materials

- **EasyBrowser Lesson 1** Xcode project

Opening

Have you ever used an app that started a phone call, started an email, or opened a page in Safari?

Agenda

- Using Interface Builder and the Object Library (⌥⌘L), add a button and set its label to **Open in Safari**.
- Add constraints to center the button in the view.
- Using Interface Builder and the Assistant Editor (⌥⌘↩), create a connection from the button to a controller action called `openPageInSafari:`.

```
@IBAction func openPageInSafari(sender: UIButton) {
   // open http://developer.apple.com
}
```

- Implement the `openPageInSafari:` method.

```
@IBAction func openPageInSafari(sender: UIButton) {
   if let url = NSURL(string: "http://developer.apple.com") {
      UIApplication.sharedApplication().openURL(url)
   }
}
```

- Explain the creation of an `NSURL`, representing a URL or "the location of a resource": a web page that we wish to open.
- Discuss the optional binding used to unwrap the optional `NSURL?` returned by the `NSURL` initializer.
- Using the Xcode Documentation and API Reference (⇧⌘0), explore the `UIApplication` class reference and the `openURL:` method.
- Explain how the call to `UIApplication.sharedApplication` returns a reference to the app instance itself.
- Run the app (⌘R), tap the button, and observe how Safari enters the foreground.

Closing

What are the benefits and drawbacks of sending users to Safari from our app?

Modifications and Extensions

- Describe the relationship between every app's inherent `UIApplication` object and the App Delegate.
- Pass an invalid URL string literal to the `NSURL` initializer, and display an alert when the optional binding fails.

Resources

UIKit User Interface Catalog: Button https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/UIKitUICatalog/UIButton.html

Xcode Overview: Build a User Interface https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/edit_user_interface.html

Creating an Action Connection https://developer.apple.com/library/ios/recipes/xcode_help-IB_connections/chapters/CreatingAction.html

Apple URL Scheme Reference http://developer.apple.com/library/ios/featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html

NSURL Class Reference http://developer.apple.com/library/ios/documentation/Cocoa/Reference/Foundation/Classes/NSURL_Class/Reference/Reference.html

UIApplication Class Reference https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIApplication_Class/

The Swift Programming Language: Optional Binding https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/TheBasics.html#//apple_ref/doc/uid/TP40014097-CH5-ID333