Seamless Linking to Your App

Session 509

Conrad Shultz Safari and WebKit Software Engineer
Jonathan Grynspan Core Services Software Engineer
Your App, Your Website

WWDC 2014 Session Videos

A Look Inside Presentation Controllers
Frameworks: iOS

A Strategy for Great Work
Frameworks: iOS, OS X

Accessibility on iOS
Frameworks: iOS

Accessibility on OS X
Frameworks: OS X

Adapting Your App to the New UI of OS X Yosemite
Frameworks: OS X

Adapting Advanced Features of the New UI of OS X Yosemite
Frameworks: OS X

Adapting AirPrint
Frameworks: Core OS, iOS, OS X

Adapting Handoff on iOS and OS X
Frameworks: iOS, OS X

Advanced CloudKit
Frameworks: iOS, OS X

APP FRAMEWORKS

A Look Inside Presentation Controllers
2014 - Session 228

A Strategy for Great Work
2014 - Session 237

Accessibility on iOS
2014 - Session 210

Accessibility on OS X
2014 - Session 207

Adapting Your App to the New UI of OS X Yosemite
2014 - Session 209

Adapting Advanced Features of the New UI of OS X Yosemite
2014 - Session 210

Adapting Handoff on iOS and OS X
2014 - Session 219

Advanced Graphics and Animations for iOS
2014 - Session 419

Advanced iOS Application Architecture and Services
2014 - Session 229

Advanced Scrollviews and Touch Handling...
Demo

WWDC App

Conrad Shultz
Safari and WebKit Software Engineer
Building Seamless Integration

Have your app handle link to your website
Have your website link to your app
Help your users log into your app
Building Seamless Integration

Have your app handle link to your website
Have your website link to your app
Help your users log into your app
Building Seamless Integration

Have your app handle link to your website
Have your website link to your app
Help your users log into your app
Building Seamless Integration

Have your app handle link to your website
Have your website link to your app
Help your users log into your app
Linking to Your App

Jonathan Grynspan
Core Services Software Engineer
Handling URLs in Your App
Handling URLs in Your App

Custom URL schemes
Handling URLs in Your App

Custom URL schemes

Let apps communicate with each other
Handling URLs in Your App

Custom URL schemes

Let apps communicate with each other
Don’t always map to your app
Handling URLs in Your App

Custom URL schemes

Let apps communicate with each other
Don’t always map to your app
Don’t work without your app installed
Handling URLs in Your App

Custom URL schemes

Let apps communicate with each other
Don’t always map to your app
Don’t work without your app installed
Don’t protect users’ privacy
Handling URLs in Your App
A better way?

Let apps communicate with each other
Don’t always map to your app
Don’t work without your app installed
Don’t protect users’ privacy
Handling URLs in Your App

A better way?

Let apps communicate with each other just as easily
Don’t always map to your app
Don’t work without your app installed
Don’t protect users’ privacy
Handling URLs in Your App
A better way?

Let apps communicate with each other just as easily
Don’t always map to your app
Don’t work without your app installed
Don’t protect users’ privacy
Handling URLs in Your App

A better way?

Let apps communicate with each other just as easily

Securely map to apps you choose

Don’t work without your app installed

Don’t protect users’ privacy
Handling URLs in Your App

A better way?

Let apps communicate with each other just as easily

Securely map to apps you choose

Don’t work without your app installed

Don’t protect users’ privacy
Handling URLs in Your App

A better way?

Let apps communicate with each other just as easily
Securely map to apps you choose
Work universally, fall back to Safari
Don’t protect users’ privacy
Handling URLs in Your App

A better way?

Let apps communicate with each other just as easily
Securely map to apps you choose
Work universally, fall back to Safari
Don’t protect users’ privacy
Handling URLs in Your App
A better way

Let apps communicate with each other just as easily
Securely map to apps you choose
Work universally, fall back to Safari
Protect users’ privacy
Breakdown of a Universal Link
Breakdown of a Universal Link

Breakdown of a Universal Link


Scheme ("https" or "http")
Breakdown of a Universal Link


Domain
Breakdown of a Universal Link


Path or path prefix
Breakdown of a Universal Link

Getting Your Server Ready
Getting Your Server Ready

Create your “apple-app-site-association” file
Generate an SSL certificate
Sign your file
Upload to your server
Getting Your Server Ready

"apple-app-site-association" file

{
    "applinks": {
        "apps": [],
        "details": {
            "9JA89QQLNQ.com.apple.wwdc": {
                "paths": ["*"
            }
        }
    }
}
Getting Your Server Ready

“apple-app-site-association” file

```json
{
    "applinks": {
        "apps": [],
        "details": {
            "9JA89QQLNQ.com.apple.wwdc": {
                "paths": [ "*" ]
            }
        }
    }
}
```
Getting Your Server Ready

“apple-app-site-association” file

{
  "applinks": {
    "apps": [],
    "details": {
      "9JA89QQLNQ.com.apple.wwdc": {
        "paths": [ "*" ]
      }
    }
  }
}
Getting Your Server Ready
“apple-app-site-association” file

{
    "applinks": {
        "apps": [],
        "details": {
            "9JA89QQLNQ.com.apple.wwdc": {
                "paths": [ "*" ]
            }
        }
    }
}
Getting Your Server Ready

“apple-app-site-association” file

```json
{
   "applinks": {
      "apps": [],
      "details": {
         "9JA89QQLNQ.com.apple.wwdc": {
            "paths": [
               "/wwdc/news/",
               "/videos/wwdc/2015/*"
            ]
         }
      }
   }
}
```
Getting Your Server Ready

Create your “apple-app-site-association” file

Generate an SSL certificate

Sign your file

Upload to your server
Getting Your Server Ready

Create your “apple-app-site-association” file
Generate an SSL certificate

Sign your file
Upload to your server
Getting Your Server Ready

Signing your JSON file

openssl smime \
  -sign \ 
  -nodetach \ 
  -in "unsigned.json" \ 
  -out "apple-app-site-association" \ 
  -outform DER \ 
  -inkey "private-key.pem" \ 
  -signer "certificate.pem"
openssl smime \
  -sign \
  -nodetach \
  -in "unsigned.json" \
  -out "apple-app-site-association" \
  -outform DER \
  -inkey "private-key.pem" \
  -signer "certificate.pem"
Getting Your Server Ready

Signing your JSON file

openssl smime \
  -sign \
  -nodetach \
  -in "unsigned.json" \
  -out "apple-app-site-association" \
  -outform DER \
  -inkey "private-key.pem" \
  -signer "certificate.pem"
Getting Your Server Ready

Signing your JSON file

openssl smime \
  -sign \
  -nodetach \
  -in "unsigned.json" \
  -out "apple-app-site-association" \
  -outform DER \
  -inkey "private-key.pem" \
  -signer "certificate.pem" \
  -certfile "intermediate-certificate.pem"
Getting Your Server Ready

Create your “apple-app-site-association” file
Generate an SSL certificate
Sign your file

Upload to your server
Getting Your Server Ready

Create your “apple-app-site-association” file
Generate an SSL certificate
Sign your file
Upload to your server

https://www.example.com/apple-app-site-association
Getting Your Server Ready

Create your “apple-app-site-association” file

Generate an SSL certificate

Sign your file

Upload to your server
Getting Your Server Ready

Create your “apple-app-site-association” file

Upload to your server
Getting Your App Ready
Getting Your App Ready

UIApplicationDelegate support
Getting Your App Ready

UIApplicationDelegate support

```swift
func application(application: UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: ([AnyObject]!) -> Void) -> Bool
```
Getting Your App Ready

UIApplicationDelegate support

```swift
func application(application: UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: ([AnyObject]!) -> Void) -> Bool

var webpageURL : NSURL?
```
Getting Your App Ready

UIApplicationDelegate support

```swift
func application(application: UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: ([AnyObject]!)->Void) -> Bool

var webpageURL : NSURL?

var activityType : String
```
func application(application: UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: ([AnyObject]!) -> Void) -> Bool

var webpageURL : NSURL?

var activityType : String

let NSUserActivityTypeBrowsingWeb : String
Getting Your App Ready

UIApplicationDelegate support

```swift
func application(application: UIApplication, continueUserActivity userActivity: NSUserActivity, restorationHandler: ([AnyObject]!) -> Void) -> Bool

var webpageURL : NSURL?

var activityType : String

let NSUserActivityTypeBrowsingWeb : String

class NSURLComponents : NSObject
```
Getting Your App Ready
Configuring your associated domains

Associated Domains

Domains:
aplinks:example.com
applinks:www.example.com

Steps:
✓ Add the "Associated Domains" entitlement to your entitlements file
✓ Add the "Associated Domains" entitlement to your App ID
Getting Your App Ready

Best practices
Getting Your App Ready

Best practices

Validate input
Getting Your App Ready

Best practices

Validate input

Fail gracefully
Getting Your App Ready

Best practices

Validate input
Fail gracefully
Send data over HTTPS
Getting Your App Ready

Best practices

- Validate input
- Fail gracefully
- Send data over HTTPS
Getting Your App Ready

Best practices

Validate input
Fail gracefully
Send data over HTTPS

Networking with NSURLSession
Demo

Xcode and iOS 9 SDK

Jonathan Grynspan
Core Services Software Engineer
Linking to Your App

Conrad Shultz
Safari and WebKit Software Engineer
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Linking to Your App

Some advantages of universal links

- No flashes or bouncing through Safari
- No code required on your website
- Graceful fallback to Safari
- Platform-neutral
- No extra server round-trips
- Full user privacy protection
Linking to Your App

Some advantages of universal links

No flashes or bouncing through Safari
No code required on your website
Graceful fallback to Safari
Platform-neutral
No extra server round-trips
Full user privacy protection
Smart App Banners
Promote your app from your website
Smart App Banners
What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners
What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners

What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners

What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners

What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners

What are smart app banners?

Presented by Safari on iOS
Know whether an app is installed
Easy to add to your website
Used to index your app
More effective than custom URL schemes
Smart App Banners
Deploying smart app banners

Add a `meta` tag to your website’s `head`:

```html
<head>
  <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```
Add a `meta` tag to your website’s `<head>`:

```html
<head>
    <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```
Smart App Banners
Deploying smart app banners

Add a `meta` tag to your website’s `head`:

```html
<head>
  <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```
Smart App Banners
Deploying smart app banners

Add a **meta** tag to your website’s **head**:

```html
<head>
    <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```

Use [https://linkmaker.itunes.apple.com](https://linkmaker.itunes.apple.com) to find your App Store ID
Smart App Banners
Deploying smart app banners

Add a `meta` tag to your website’s `head`:

```html
<head>
  
  <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData"/>

  
</head>
```
Smart App Banners
Deploying smart app banners

Add a `meta` tag to your website’s `head`:

```html
<head>
  <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```

`app-argument` is passed to `application(_:openURL:sourceApplication:annotation:)`, and is also used to index your app
Add a `meta` tag to your website’s `head`:

```html
<head>
    <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">
</head>
```

`app-argument` is passed to `application(_:openURL:sourceApplication:annotation:)`, and is also used to index your app.

---

**Introducing Search APIs**

**Mission**
**Wednesday 11:00AM**
Smart App Banners
Deploying smart app banners

Add a `meta` tag to your website's `head`:

```html
<head>

  <meta name="apple-itunes-app" content="app-id=640199958, app-argument=https://developer.apple.com/wwdc/schedule, affiliate-data=optionalAffiliateData">

</head>
```

Smart App Banners will not appear in the iOS Simulator
Shiny

Username

Password

Log In

Forgot Password?
Select a Safari Saved Password to Use With “Shiny”

You can view and remove passwords in the Passwords & Autofill section of Safari settings.

conrad@example.com

Not Now OK
Shared Web Credentials

Help your users log into your app
Saved Passwords and Safari
How Safari helps users—and you

- Securely save credentials for users
- Sync to all devices with iCloud Keychain
- Suggest secure passwords for users
- AutoFill saved credentials
Saved Passwords and Safari
How Safari helps users—and you

Securely save credentials for users
Sync to all devices with iCloud Keychain
Suggest secure passwords for users
AutoFill saved credentials
Saved Passwords and Safari
How Safari helps users—and you

- Securely save credentials for users
- Sync to all devices with iCloud Keychain
- Suggest secure passwords for users
- AutoFill saved credentials
Saved Passwords and Safari
How Safari helps users—and you

- Securely save credentials for users
- Sync to all devices with iCloud Keychain
- Suggest secure passwords for users
- AutoFill saved credentials
Saved Passwords and Safari
How Safari helps users—and you

Securely save credentials for users
Sync to all devices with iCloud Keychain
Suggest secure passwords for users
AutoFill saved credentials
Shared Web Credentials

Adopting shared web credentials

Associate your app and your website
Teach your app to ask for Shared Web Credentials
Shared Web Credentials

Adopting shared web credentials

Associate your app and your website

Teach your app to ask for Shared Web Credentials
Shared Web Credentials

Adopting shared web credentials

Associate your app and your website

Teach your app to ask for Shared Web Credentials
App-Website Association
Server-side
Add a new top-level dictionary to your `apple-app-site-association` JSON file:

```
{
  "applinks": {
    "apps": [],
    "details": {
      "9JA89QQLNQ.com.apple.wwdc": {
        "paths": [
          "/wwdc/news/",
          "/videos/wwdc/2015/*"
        ]
      }
    }
  }
}
```
Add a new top-level dictionary to your `apple-app-site-association` JSON file:

```json
{
    "applinks": {
        ...
    },
    "webcredentials": {
        "apps": [ "A1B2C3D4E5.com.example.AppName" ]
    }
}
```
Add a new top-level dictionary to your `apple-app-site-association` JSON file:

```
{
    "applinks": {
        ...
    }
}

"webcredentials": {
    "apps": [ "A1B2C3D4E5.com.example.AppName" ]
}
```
Add a new top-level dictionary to your apple-app-site-association JSON file:

```json
{
    "applinks": {
        ...
    }
}
"webcredentials": {
    "apps": [ "A1B2C3D4E5.com.example.AppName" ]
}
}
App-Website Association

App-side
App-Website Association

App-side

Add a new entry to your app’s `com.apple.developer.associated-domains` entitlement:

```webcredentials:example.com```
App-Website Association

App-side

Add a new entry to your app’s `com.apple.developer.associated-domains` entitlement:

`webcredentials:example.com`

![Associated Domains](image.png)

**Steps:**
- Add the “Associated Domains” entitlement to your entitlements file
- Add the “Associated Domains” entitlement to your App ID
Shared Web Credentials

The app workflow

1. Check for Saved Credentials
2. Check Safari for Credentials
3. Prompt User
4. Save Credentials
5. Log In
Shared Web Credentials

The app workflow

- Log In
  - Check Safari for Credentials
  - Prompt User
  - Save Credentials
  - Check for Saved Credentials
Shared Web Credentials

The app workflow

1. Check for Saved Credentials
2. Check Safari for Credentials
3. Prompt User
4. Save Credentials
5. Log In

Shiny

Username
Password
Log In

Forgot Password?
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
  if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
    let credentials = cfCredentials as [AnyObject]
    let credential = credentials[0]
    let userName = credential[kSecAttrAccount as String]
    let password = credential[kSecSharedPassword as String]
    dispatch_async(dispatch_get_main_queue()) {
      // Log in with the user name and password.
    }
  } else {
    dispatch_async(dispatch_get_main_queue()) {
      // Show in-app login UI.
    }
  }
}
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
Retrieving Safari’s Saved Credentials

```swift
// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
```
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    } else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
Retrieving Safari’s Saved Credentials

// Find all matching user names from domains specified in the entitlement.
SecRequestSharedWebCredential(nil, nil) { cfCredentials, error in
    if let cfCredentials = cfCredentials where CFArrayGetCount(cfCredentials) != 0 {
        let credentials = cfCredentials as [AnyObject]
        let credential = credentials[0]
        let userName = credential[kSecAttrAccount as String]
        let password = credential[kSecSharedPassword as String]
        dispatch_async(dispatch_get_main_queue()) {
            // Log in with the user name and password.
        }
    }
    else {
        dispatch_async(dispatch_get_main_queue()) {
            // Show in-app login UI.
        }
    }
}
}
Shared Web Credentials

Updating Safari’s saved credentials

SecAddSharedWebCredential("example.com",
    userName,
    password) { error in }
let password: String = SecCreateSharedWebCredentialPassword().takeRetainedValue() as String
print("The generated password was: \(password)")

// E.g.: The generated password was: uFA-T9C-aRD-cDP
let password: String = SecCreateSharedWebCredentialPassword().takeRetainedValue() as String
print("The generated password was: \(password)")
// E.g.: The generated password was: uFA–T9C–aRD–cDP
Summary

Give your users the best experience by having your app handle link to your website
Deploy Smart App Banners to help users of your website discover your app
Help your users log into your app by adopting Shared Web Credentials
Summary

Give your users the best experience by having your app handle link to your website

Deploy Smart App Banners to help users of your website discover your app

Help your users log into your app by adopting Shared Web Credentials
Summary

Give your users the best experience by having your app handle link to your website
Deploy Smart App Banners to help users of your website discover your app
Help your users log into your app by adopting Shared Web Credentials
Summary

Give your users the best experience by having your app handle links to your website
Deploy Smart App Banners to help users of your website discover your app
Help your users log into your app by adopting Shared Web Credentials
More Information

Technical Support
Apple Developer Forums
http://developer.apple.com/forums

Developer Technical Support
http://developer.apple.com/contact

Shared Web Credentials Reference
More Information

General Inquiries
Jon Davis Web Technologies Evangelist
web-evangelist@apple.com

Jake Behrens App Frameworks Evangelist
behrens@apple.com
<table>
<thead>
<tr>
<th>Session</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing Safari View Controller</td>
<td>Nob Hill</td>
<td>Tuesday 1:30PM</td>
</tr>
<tr>
<td>Privacy and Your App</td>
<td>Pacific Heights</td>
<td>Tuesday 2:30PM</td>
</tr>
<tr>
<td>Introducing Search APIs</td>
<td>Mission</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Networking with NSURLSession</td>
<td>Pacific Heights</td>
<td>Thursday 9:00AM</td>
</tr>
<tr>
<td>Your App, Your Website, and Safari</td>
<td>WWDC14</td>
<td></td>
</tr>
<tr>
<td>Adopting Handoff on iOS and OS X</td>
<td>WWDC14</td>
<td></td>
</tr>
<tr>
<td>Labs</td>
<td>Frameworks Lab E</td>
<td>Friday 1:30PM</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>App Search and Spotlight</td>
<td>Frameworks Lab E</td>
<td>Friday 1:30PM</td>
</tr>
<tr>
<td>Safari and WebKit</td>
<td>Graphics, Games, and Media Lab A</td>
<td>Friday 12:00PM</td>
</tr>
</tbody>
</table>