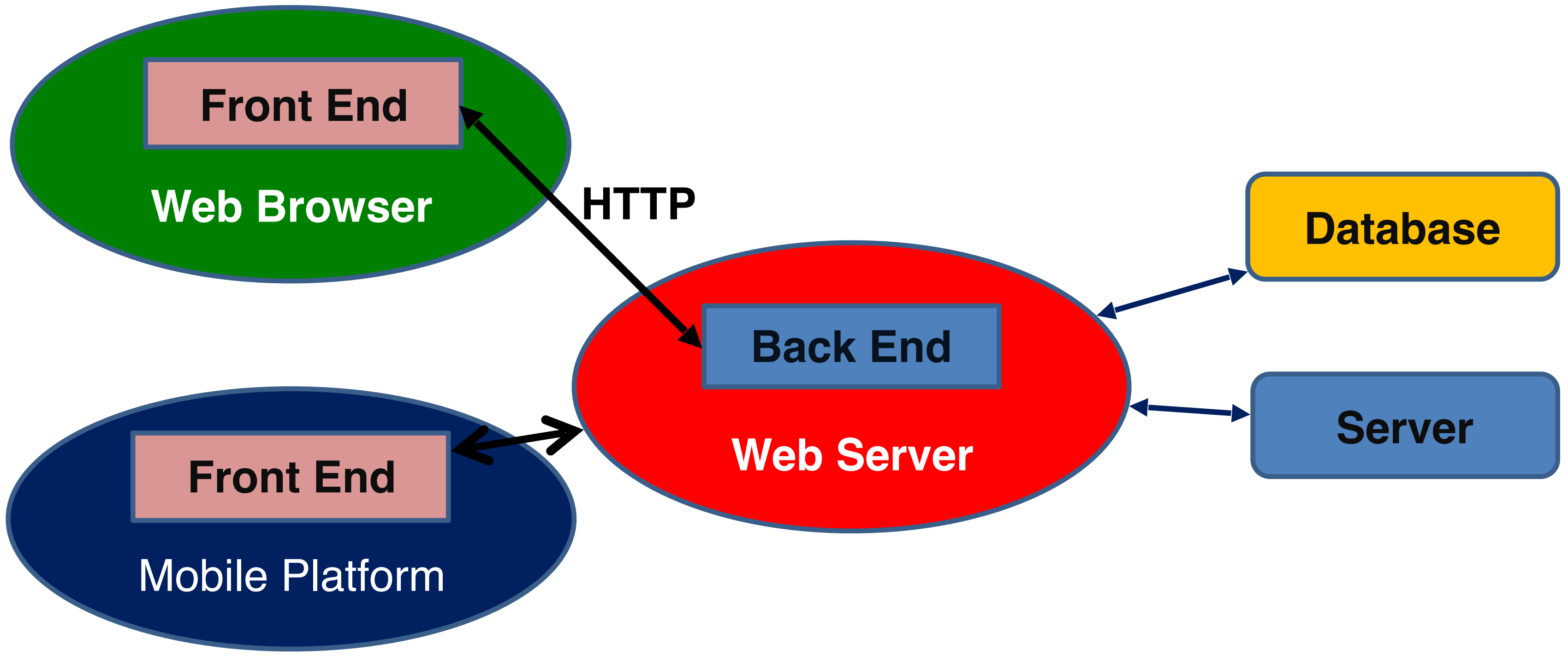
CS1320
Creating Modern Web and Mobile Applications

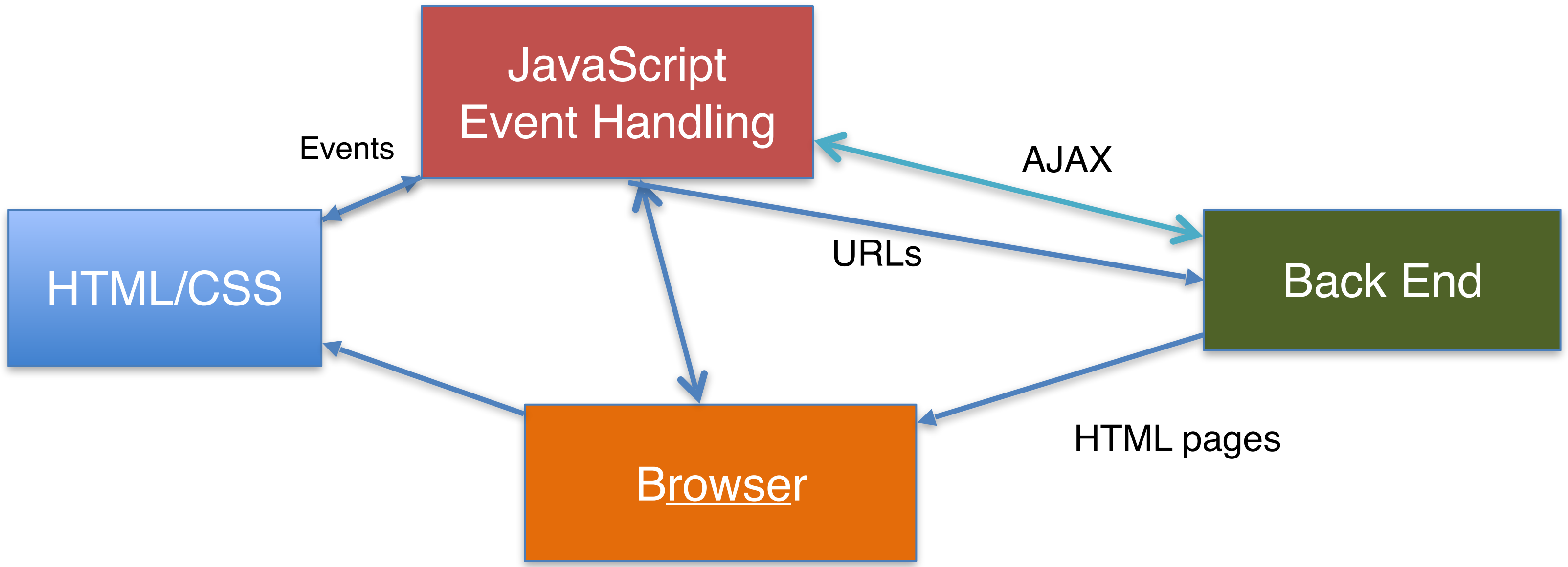
Lecture 21

Mobile Applications I

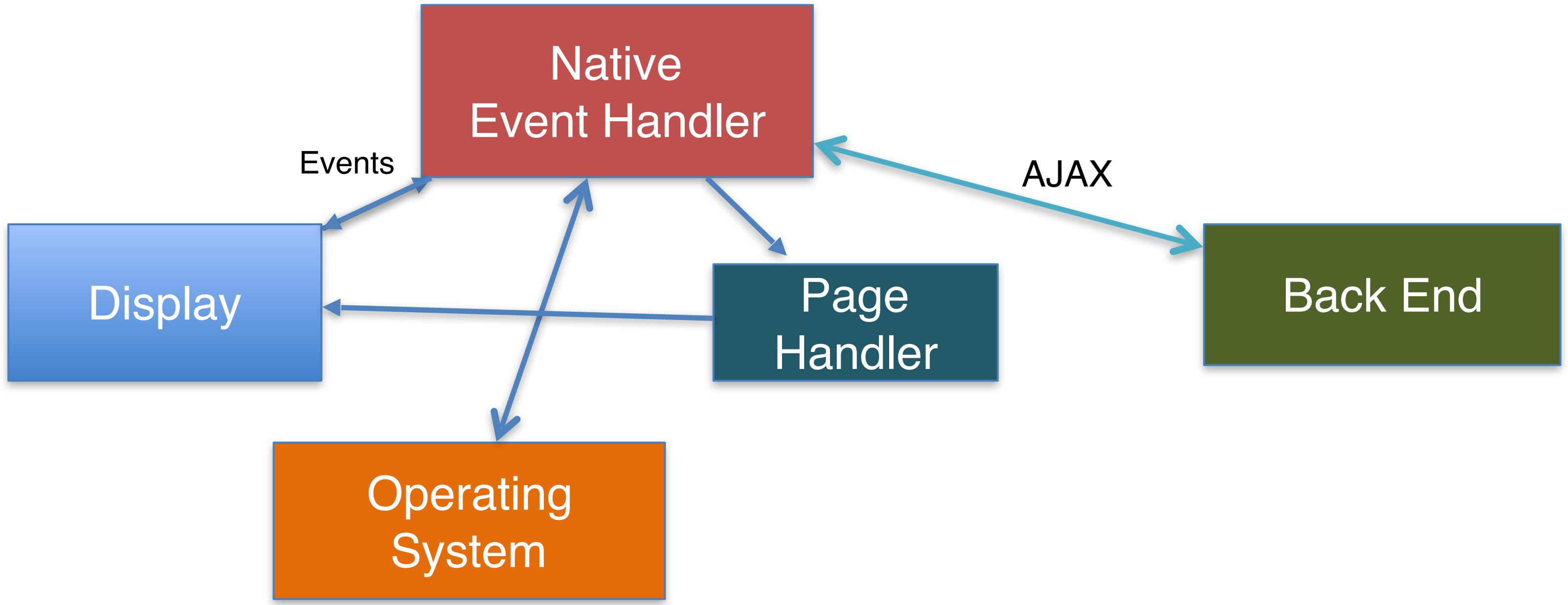
Web Application Architecture



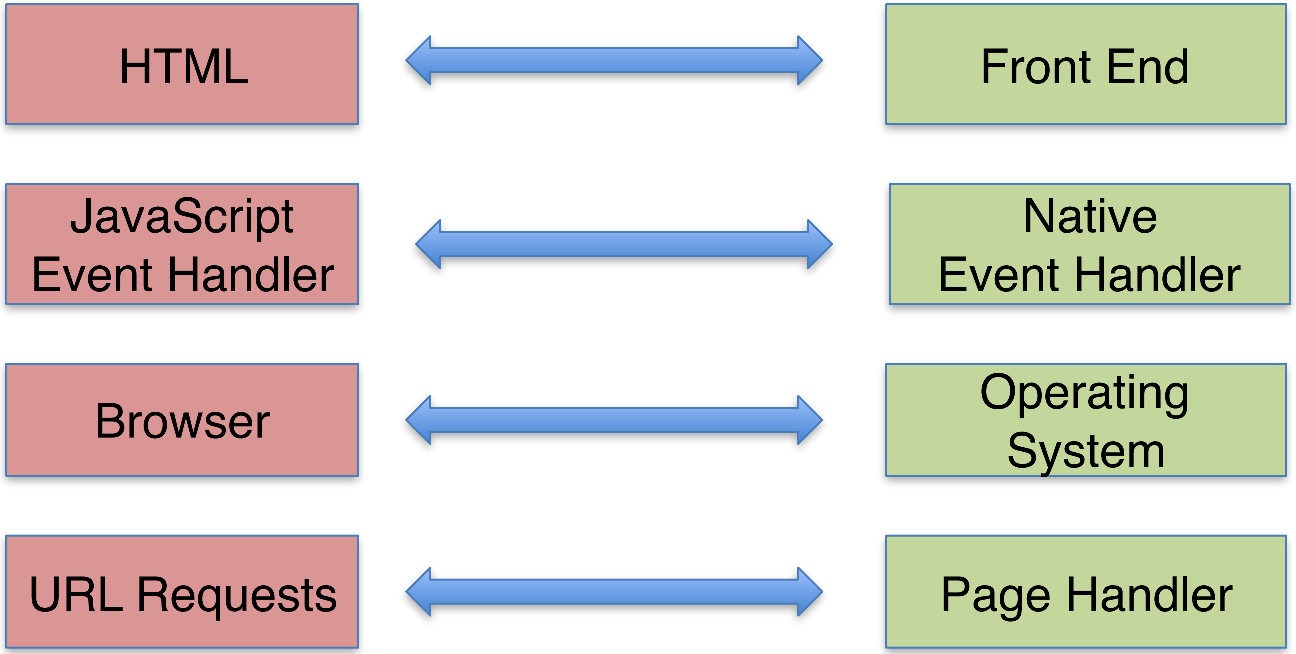
Structure of a Web Application



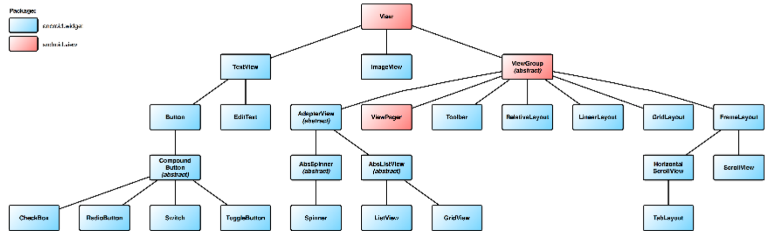
Structure of a Mobile Application



Web and Mobile Differences



Mobile Front Ends

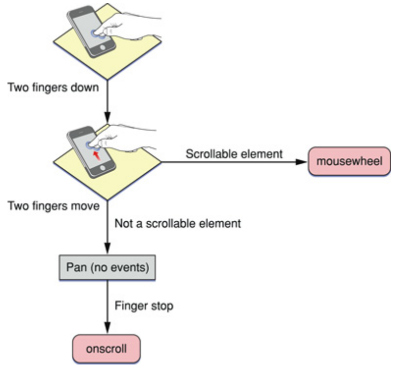


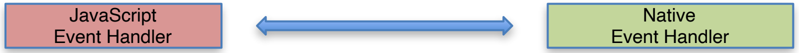
- Widget-Based
  - Hierarchy of widgets replaces HTML hierarchy
  - Text is in label widgets
  - Widgets exist for buttons, inputs, etc.
    - Corresponding to HTML form elements
  - Layout is done using layout widgets
    - These control how their contents are displayed
- Widget Properties control formatting and display
- Widgets can be created and nested directly
- There is a language for defining widget hierarchies
  - Generally XML-based static description
  - Differs on the different platforms

HTML
Front End

Mobile Event Handling

- The actual code is event-based
  - Wait for event
  - Act on the event by starting action that yields new events
- Events are similar to those of the browser
  - Based on user actions
  - Based on external events (timers, input ready, …)
  - But not quite the same (and they vary by platform)
- Event handling is written in the native language of the platform
  - This is what is actually executed





Browser versus Operating System

- For web applications, all interactions are with the browser
  - Mobile applications don't use the browser, they run directly
- The functionality of the browser is replaced the operating system
  - Along with a suite of system libraries that provide functionality
  - Different platforms = Different names but the same functionality
  - Can have more functionality than the browser
    - Especially for newer features of the phone
    - But the browser is generally catch

Browser
Operating System

URL Requests versus Page Management

- A mobile application doesn't go to the back end to get the next page
  - Instead it tells the operating system to switch to a different set of widgets
  - These are defined by pages (page == widget hierarchy for the page)
- Pages are akin to HTML pages
  - Can have separate code, events, etc.
  - Back on the phone goes to previous page, forward to next, …
- All pages defined as part of the mobile application

URL Requests
Page Handler

Other Mobile Differences

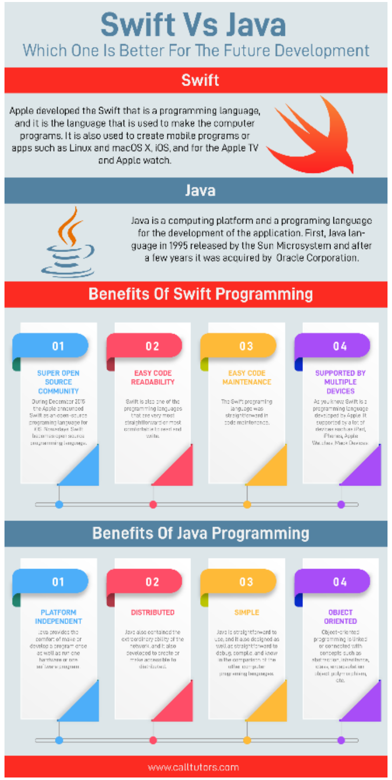
- Deployment
  - Need to go through an app store to make the app available
  - This can require certification and money
- Development
  - Development platforms for mobile apps are specific to the platf
    - Android – Android Development Environment (IntelliJ extension)
    - Apple – XCode
  - Differ from the platforms you used for mobile applications
    - No browser-based debugger for example

Native Mobile Applications

- Are platform specific
  - iOS for the iPhone using swift (used to be objective-c)
  - Android for android phones using Java
- Use platform specific widget sets (close, but not quite matching)
- Use platform specific library calls (close, but not quite matching)
- Use platform specific environments
- How to approach native development

What You Need to Know to Write a Mobile Application

- Native language: Swift or Java
- Event set
- Platform widget set and their properties
- OS and library calls
- Page model



Swift Vs Java

Which One Is Better For The Future Development

Swift

Apple developed the Swift that is a programming language, and it is the language that is used to make the computer programs. It is also used to create mobile programs or apps such as Linux and macOS X, iOS, and for the Apple TV and Apple watch.

Java

Java is a computing platform and a programing language for the development of the application. First, Java language in 1995 released by the Sun Microsystem and after a few years it was acquired by Oracle Corporation.

Benefits Of Swift Programming

01 SUPER OPEN SOURCE COMMUNITY

During December 2015 the Apple announced Swift as an open-source programing language for iOS. Nowadays Swift becomes open source programming language.

02 EASY CODE READABILITY

Swift is also one of the programming languages that are very most straightforward or most comfortable to read and write.

03 EASY CODE MAINTENANCE

The Swift programing language was straightforward in code maintenance.

04 SUPPORTED BY MULTIPLE DEVICES

As you know Swift is a programming language developed by Apple. It supported by a lot of devices such as iPad, iPhones, Apple Watches, Mac Devices.

Benefits Of Java Programming

01 PLATFORM INDEPENDENT

Java provides the comfort of make or develop a program once as well as run one hardware or one software program.

02 DISTRIBUTED

Java also contained the extraordinary ability of the network, and it also developed to create or make accessible to distributed.

03 SIMPLE

Java is straightforward to use, and it also designed as well as straightforward to debug, compile, and know in the comparison of the other computer programming languages.

04 OBJECT ORIENTED

Object-oriented programming is linked or connected with concepts such as abstraction, inheritance, class, encapsulation, object, polymorphism, etc.

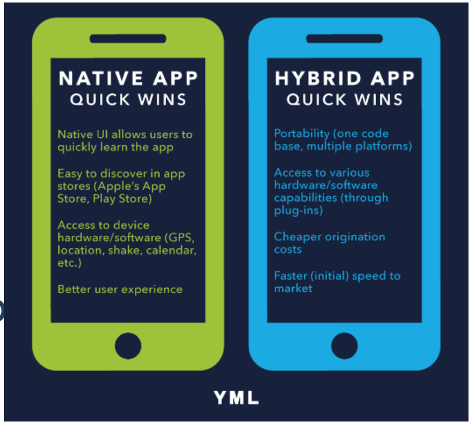
www.calltutors.com

Default Approach: Write Multiple Front I

- Pros
  - They will look like native applications (specific to the platform)
  - They can use different UI capabilities (interactions; specific to the platform)
  - They can use different phone capabilities (latest on the platform)
- Cons
  - Almost all of the functionality is the same
  - Almost all of the capabilities on one platform exist on others as well
  - Want your application to be about the same on all platforms
  - More difficult to maintain multiple versions

Alternative: Write a Web (Hybrid) Application

- Front end is HTML, CSS and JavaScript
  - Needs to be responsive to handle different sizes
  - Can provide different functionality based on platform
  - Most of the technologies are available through HTML
    - Geolocation, camera, sound, gestures, …
- A web application can be packaged to look like a native ap
  - Screen icon, with click to start
  - Packaging tools exist and are easy to use
- Disadvantages
  - Performance is not as good as a native app
  - Can't access latest OS features
  - Interface might not look or feel native
- Advantages
  - Easy to write; single platform; many apps can be done this way



NATIVE APP QUICK WINS

Native UI allows users to quickly learn the app

Easy to discover in app stores (Apple's App Store, Play Store)

Access to device hardware/software (GPS, location, shake, calendar, etc.)

Better user experience

HYBRID APP QUICK WINS

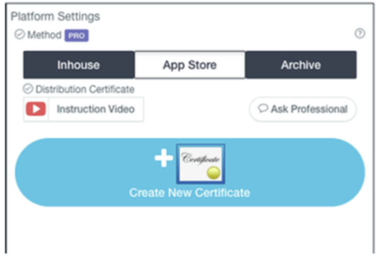
Portability (one code base, multiple platforms)

Access to various hardware/software capabilities (through plug-ins)

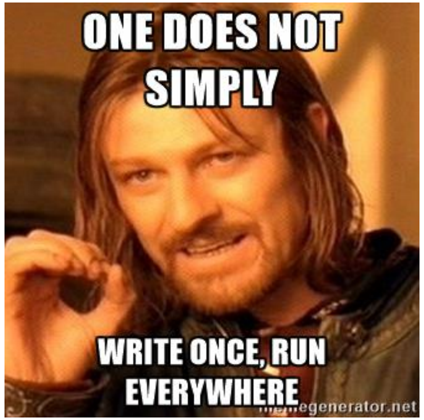
Cheaper origination costs

Faster (initial) speed to market

YML

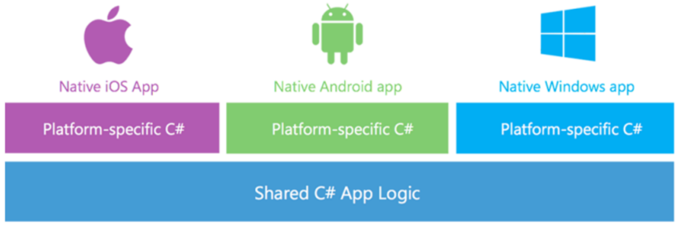


Alternative: Write Once



- The target platforms are quite similar
  - Languages, APIs, capabilities
  - People/companies have realized this and made use of it
- Write the front end in language X for some X
  - Using a fixed set of libraries
  - Compile X into Java or Swift (or Objective/C); or interpret X natively
    - Map library calls to library calls on native platform
    - Either directly or through an intermediate library
  - Generate multiple applications from a single source
  - Still need to determine how to specify UI
    - Take a common UI format and map to UI data for applications
    - Take a common set of widgets and map to native widgets

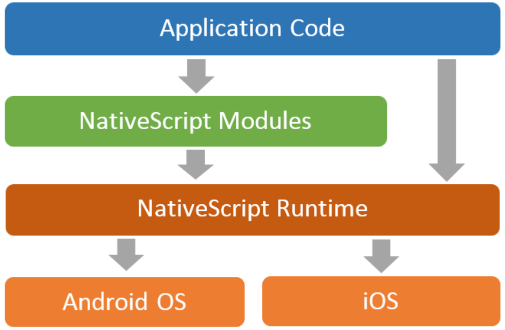
Xamarin: C#



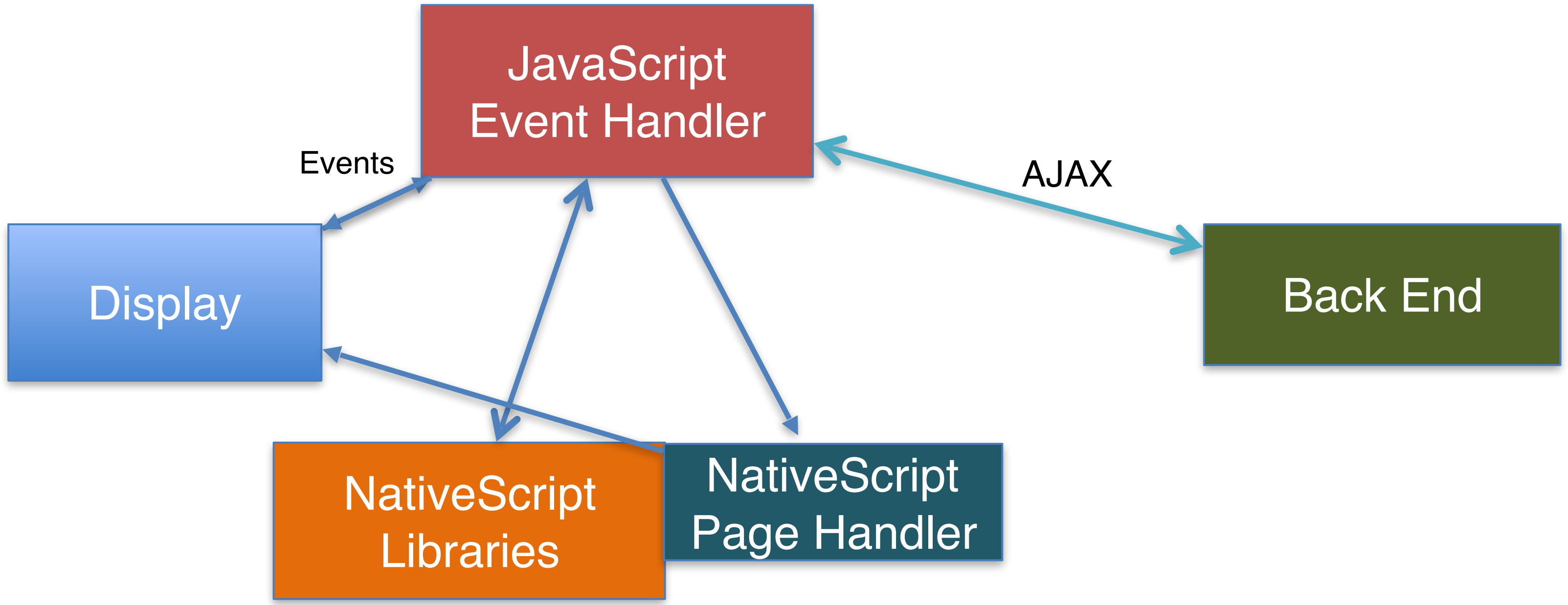
- Xamarin lets you write the app in C
  - Using Visual Studio if desired
- Using a standard UI library (and XAML)
  - XML-specified widgets
- Using common libraries to access native APIs
- Can develop on Windows and Mac
  - Community (free) edition or Enterprise (paid)

NativeScript ,React Native, Ionic: JavaScript

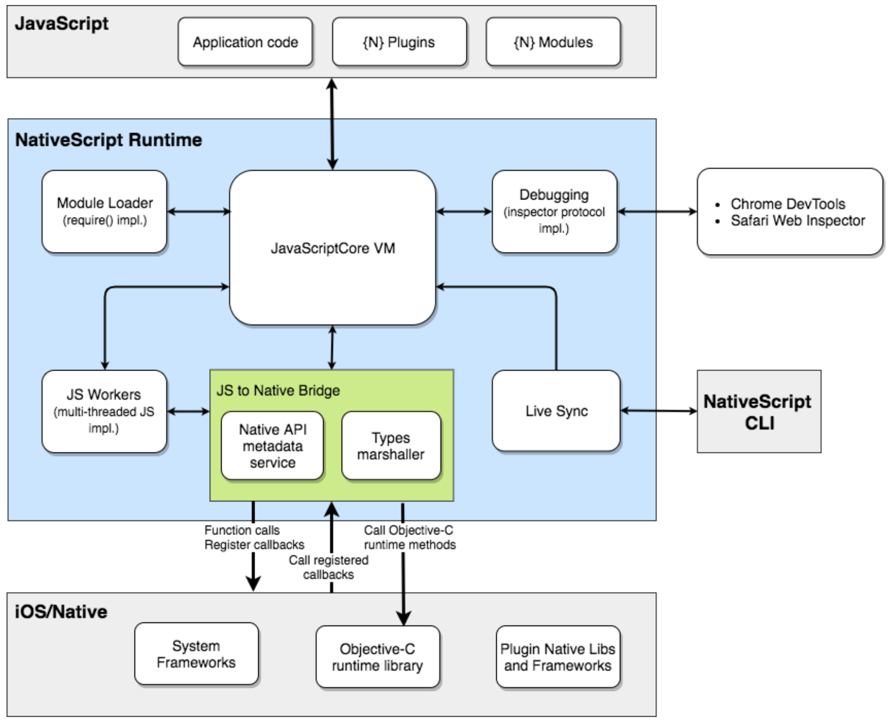
- Write the front end in JavaScript
  - With a static description of the user interface
- Have run as native code for
  - Apple, Google, …
- REACT Native uses React-like constructs
- Ionic uses html, css and interfaces with
- NativeScript uses Vue (Mustache)-like c

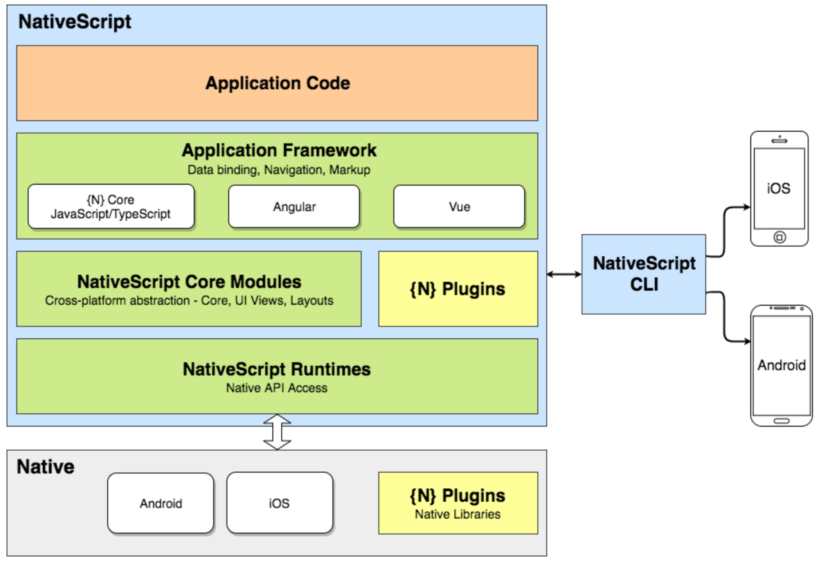


Structure of a NativeScript Application

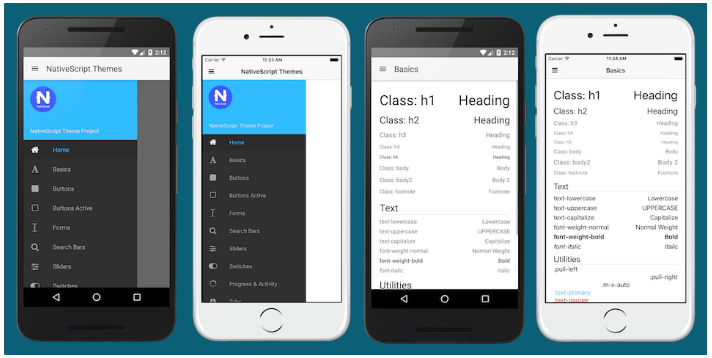


How NativeScript Works


JavaScript
Application code
{N} Plugins
{N} Modules
NativeScript Runtime
Module Loader (require() impl.)
JavaScriptCore VM
Debugging (inspector protocol impl.)
• Chrome DevTools
• Safari Web Inspector
JS Workers (multi-threaded JS impl.)
JS to Native Bridge
Native API metadata service
Types marshaller
Live Sync
NativeScript CLI
Function calls
Register callbacks
Call registered callbacks
Call Objective-C runtime methods
iOS/Native
System Frameworks
Objective-C runtime library
Plugin Native Libs and Frameworks



NativeScript Basic Widgets



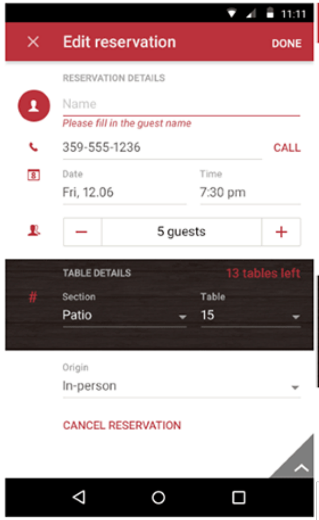
- Label – basic holder for strings
  - Also for fancy strings: FormattedString
  - HtmlView provides for HTML content
- Image
- Input widgets
  - Button, DatePicker, ListPicker, Slider, Switch, TextField, TimePicker
  - TextView, SearchBar, ListView, SegmentedBar
- Other
  - Placeholder, Progress, ActivityIndicator, WebView, ListView
- Examples: https://docs.nativescript.org/ui/overview

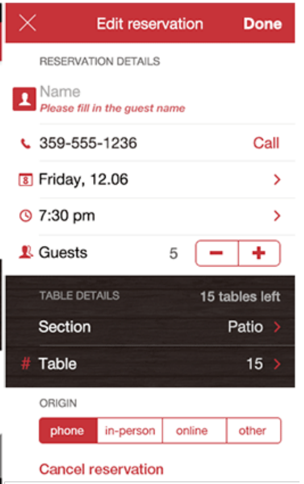
NativeScript Styling

- Widgets have properties that can be set explicitly or dynamically
  - But this isn't the default way of formatting
- CSS is used to apply to widgets
  - Selectors: single widget, all widgets of a class, …
  - Properties
    - All the common CSS properties are supported
- Also supports themes, less, …

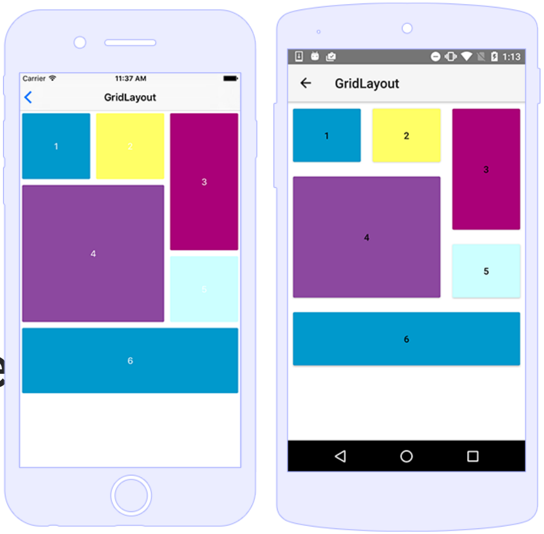
Extended NativeScript Widgets

- RadSideDrawer
- RadListView
- RadCalendar
- RadChart
- RadAutoCompleteTextView
- RadDataForm
- RadGauge
- https://docs.nativescript.org/ui/overview#components


11:11
Edit reservation DONE
RESERVATION DETAILS
Name
Please fill in the guest name
359-555-1236 CALL
Date Fri, 12.06
Time 7:30 pm
– 5 guests +
TABLE DETAILS 13 tables left
Section Patio
Table 15
Origin In-person
CANCEL RESERVATION

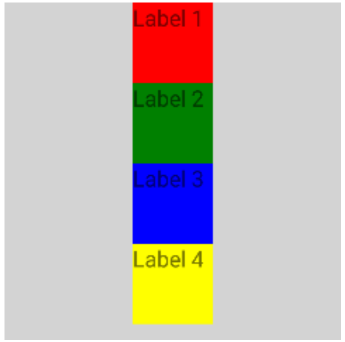

Edit reservation Done
RESERVATION DETAILS
Name
Please fill in the guest name
359-555-1236 Call
Friday, 12.06
7:30 pm
Guests 5 – +
TABLE DETAILS 15 tables left
Section Patio
Table 15
ORIGIN
phone | in-person | online | other
Cancel reservation

NativeScript Layout Widgets

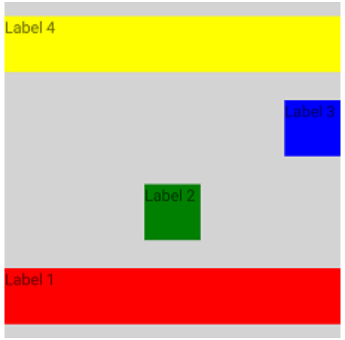


- StackLayout – vertical or horizontal rows
- FlexboxLayout – similar to CSS flex boxe
- GridLayout – similar to HTML tables
- DockLayout – around border and center
- WrapLayout – stack where things can wrap if needed (HTML standard)
- AbsoluteLayout – absolute positioning

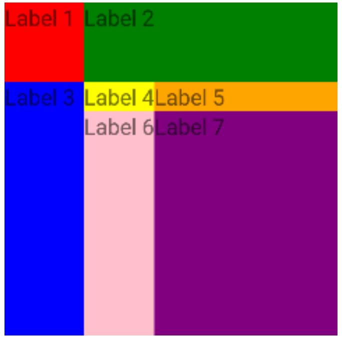
Layout Examples



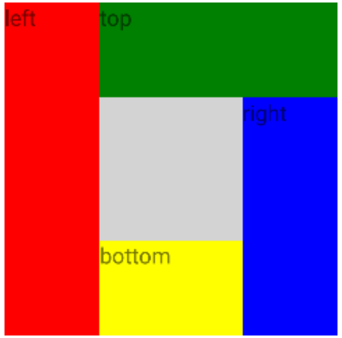
STACK



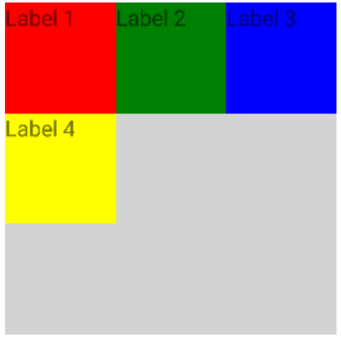
Flexbox



Grid



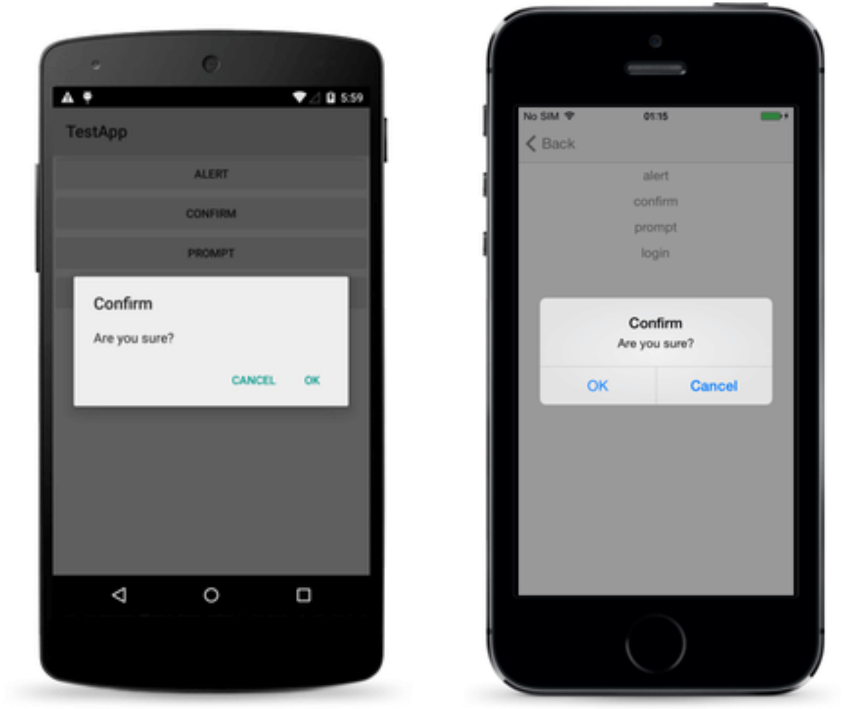
Dock



Wrap

NativeScript Organizational Widgets

- Page
  - Back, Forward, Load
- Dialog
  - AlertDialog, ActionDialog,
  - ConfirmDialog, LoginDialog,
  - PromptDialog

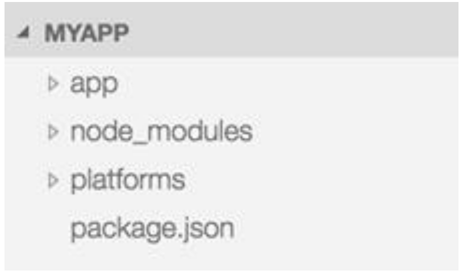


NativeScript Uses Familiar Concepts

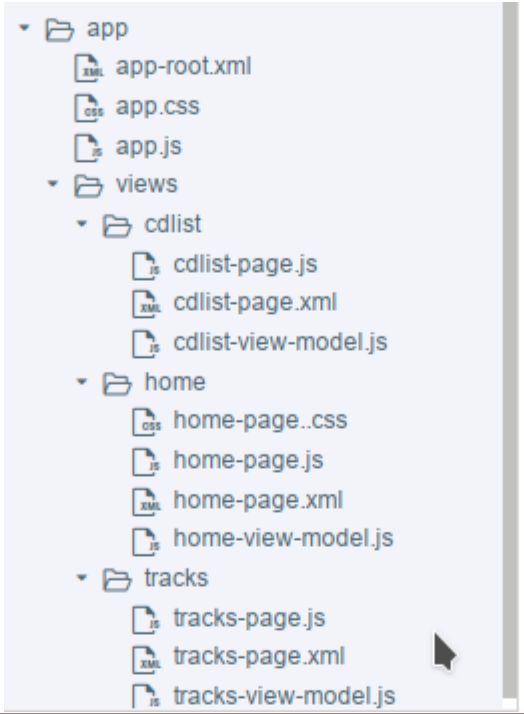
- Formatting
  - Done using CSS for the widgets, rather than widget properties
  - Much of CSS is directly usable
  - Layouts match the HTML/CSS frameworks you've used
- Page Contents
  - Done using templating ala VUE
  - Context provided as part of a page description
  - Can use VUE, REACT, Angular, … as well
- Common libraries
  - Fetch – basically the same as fetch in the browser

Application Organization

- File Structure
  - Global files
  - Per-Page files
    - XML: description of the page display
    - CSS: CSS to format the page
- Page Rendering
  - Vue-like templates
  - User provided context (goes with the page)



```
MYAPP
  app
  node_modules
  platforms
  package.json
```



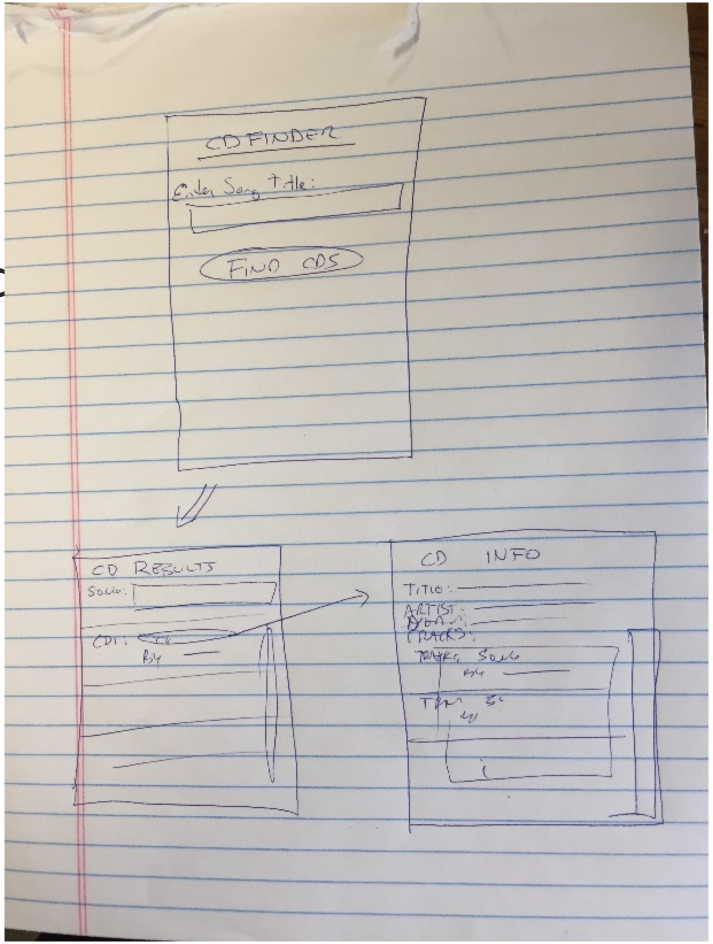
```
app
  app-root.xml
  app.css
  app.js
  views
    cdlist
      cdlist-page.js
      cdlist-page.xml
      cdlist-view-model.js
    home
      home-page..css
      home-page.js
      home-page.xml
      home-view-model.js
    tracks
      tracks-page.js
      tracks-page.xml
      tracks-view-model.js
```

Next Time

- Creating A Mobile Application Using NativeScript

Creating a Mobile Application

- Start by understanding the pages nee
- Sketch those pages
  - What they might look like
  - Interactions on the page
  - Interactions between pages
- Implement the pages one-by-one
  - Map diagram to layout widgets
  - Using sample data at first
  - Then using real data



CD FINDER

- Show using airmedia

File Organization for CD FINDER

- Pages
- Platforms

NativeScript Playground

CD FINDER Home Page

- XML
- CSS
- JavaScript

Playground And the Phone

- Connecting Playground with the phone
- Showing Changes

CD FINDER CD List Page

- XML
- CSS
- JavaScript
- Determining what the input looks like

CD FINDER Details Page

CD FINDER Back End

- RESTful interface using fetch
- Node.JS server to handle the request
- Using MONGODB from node

Next Time

- Poster Session