Hybrid apps and Cordova
Secure Mobile IoT Application Development

Dr. Hale
University of Nebraska at Omaha
Secure Mobile IoT Development– Lecture 2

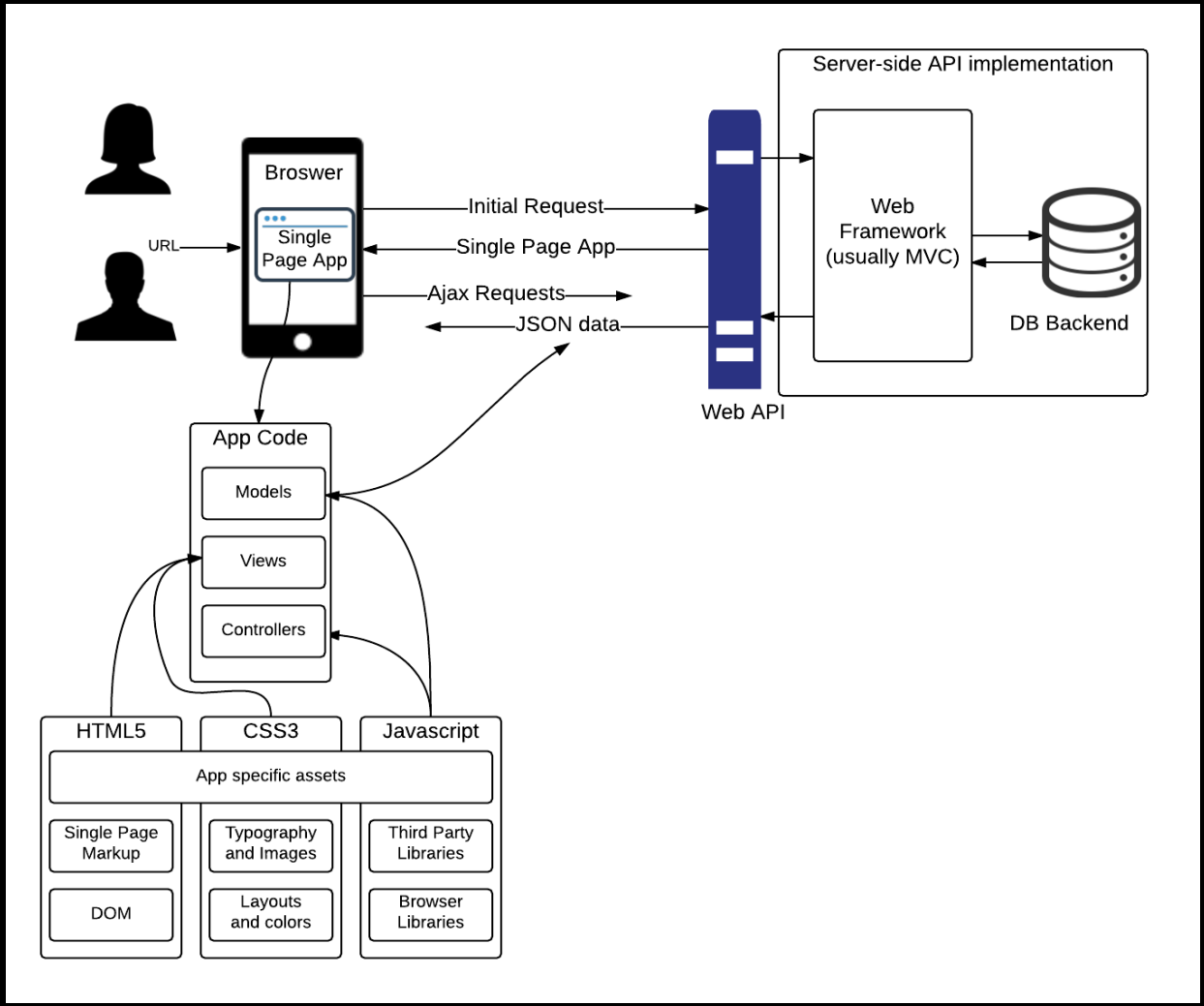
Today's Class: More on Hybrid apps and Lab time

Part 1: Designing a Cordova App

- Differences from a basic single page web app
- Application workflow
- UI changes to make it responsive on mobile
- More on Cordova
- More on Webviews
- Using native features with plugins in cordova

Part 2: Lab time

Singe Page Browser Web App Architecture



Data

- Client-side apps do not talk directly with a database, but instead pass requests through a RESTful Web API.
- Some caching may occur at the client using in-memory stores, WEBSQL, and/or key/value stores (local/session storage)
- Data at the client-side is passed through and handled by the M(model) layer of the MVC framework in-use to convert the WEB API supplied data from whatever format its in (usually JSON) to app-level objects

Backend

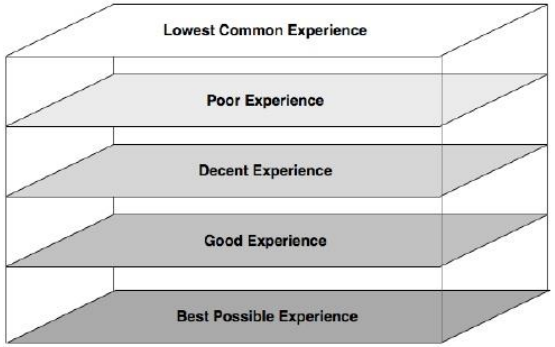
- The WEBAPI needs to know nothing about the client-side app
  - just provides the RESTful API for supplying data to the app
- It can be any server stack using any language and any database server
  - some popular web frameworks include Ruby, Django, PHP, Spring, and Express
  - some popular db options include MySQL, PostgreSQL, and MongoDB

Responsive, Adaptive UI

- All devices vary in screen size, resolution, and computation power
  - means its hard to make assumptions about your user interface
- To account for this, responsive apps detect device capabilities and adapt accordingly
- Usually this means using javascript and CSS media queiries
  - if(navigator.userAgent.match(/AppleWebKit/i) && navigator.userAgent.match(/Mobile/i){
    isiPhone = true;
    }
  - @media (max-width: 600px){ .sidebar {display: none;}}
  - can really bloat your code and make debugging tedious – since you have to inspect all the devices you support

Responsive, Adaptive UI

Progressive Enhancement



Key concept. Graceful degradation

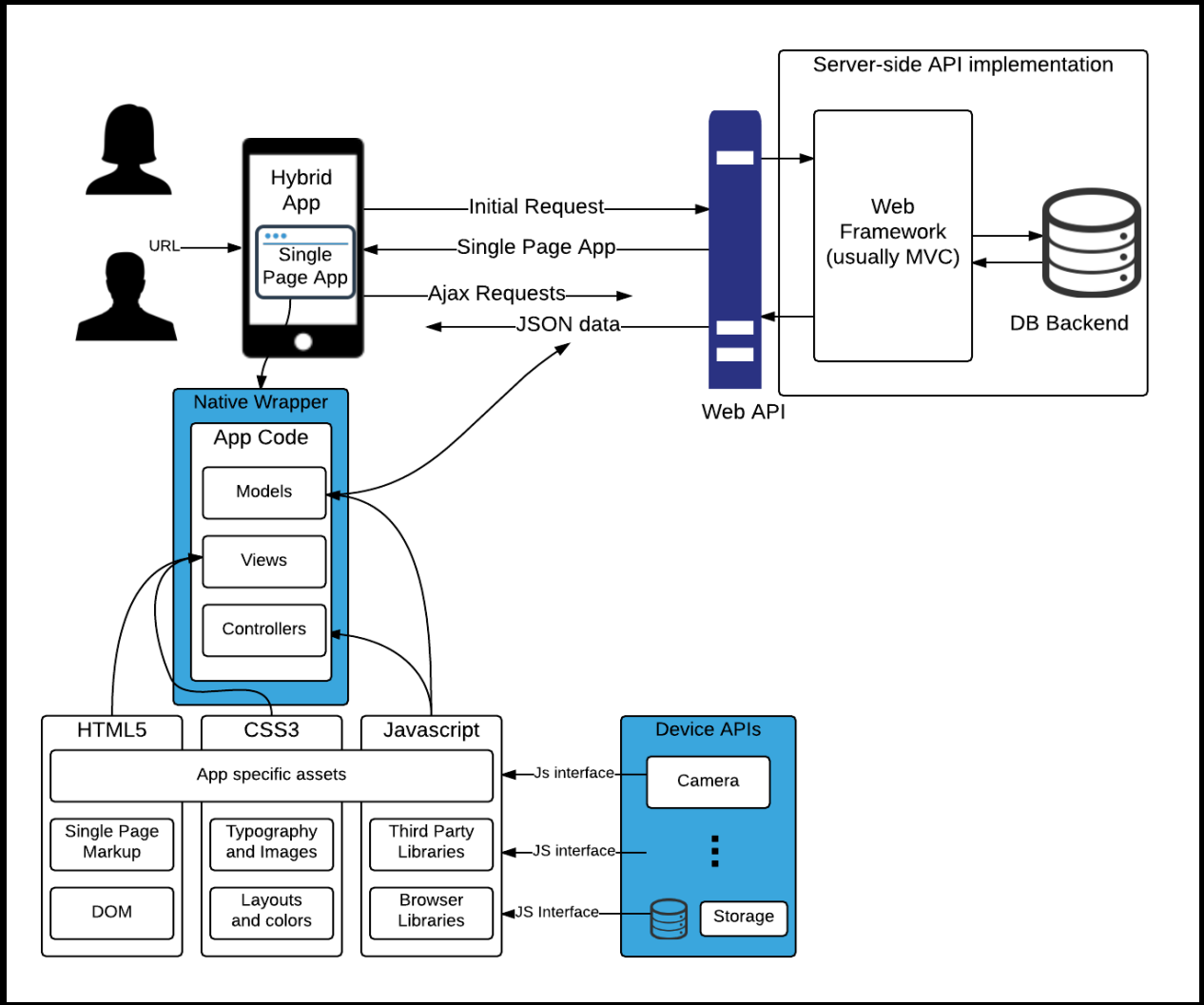
dipartimentoinformatica
Università degli Studi dell'Aquila

Responsive, Adaptive UI

Progressive Enhancement Techniques

- Always code your markup semantically
  - ensure that the page is always usable, even with no stylesheet
- Have a device plan
  - Know which device classes you intend to support before you start to code
- Have both your LCD and high-end device designs before you begin to code
  - Try to visualize a way to create both versions from one code base.
- Test on different mobile devices from the beginning
  - your incremental work must display correctly in the intended devices
- If you plan to add a desktop layer, always create the mobile version first

Hybrid Mobile App Architecture



Data differences

- Hybird apps add the ability to cache data locally using the device storage options (key/value, internal storage, external storage[sd card], SQLite)
- WebAPIs are still needed to get data and persist it outside of the local phone environment.
  - This is important for any app that requires online data or for apps with interacting users

Backend differences

- The WebAPI doesn't need to adapt to the device in anyway*
- *ideally

- In reality, it needs to be more resilient to dropped connections (mobile devices routinely lose connection)
- and it needs to minimize data usage where possible, since bandwidth is at a premium
- This is also true for the mobile web app

Native Features

- In the hybrid scenario, the client-side app code can access a variety of native-only APIs provided by the device
- In cordova, these are usually accessed via an injected interface into the navigator namespace

Native UI

- In the hybrid scenario, the client-side app code can also access a set of native UI features (like notifications)
- It also focuses the screen, preventing browser-like zooming to simulate a more native-like interface
- Developers need to adjust their client-side code to render appropriately (large enough to read, swipeable, decent visual layout, etc) for a mobile app
- While its possible to make any client-side app a hybrid app, its best practice to spend some time on making it mobile-friendly

Using Frameworks

- Frameworks like JQuery Mobile JQTouch and others are useful for matching standard mobile user interface expectations that users are accustomed to
  - there are both JS and CSS libraries that can be used
- Ember, backbone, angular, etc are all GREAT, but be very careful with how complex you let your apps get
  - The higher complexity (more libraries, more advanced calculations or computed properties, long for loops) the worse the app will perform on mobile devices.

See: http://demos.jquerymobile.com/1.4.5/intro/
And https://github.com/albertogonper/ember-jquery-mobile
And http://nativedroid.godesign.ch/material/

Cordova

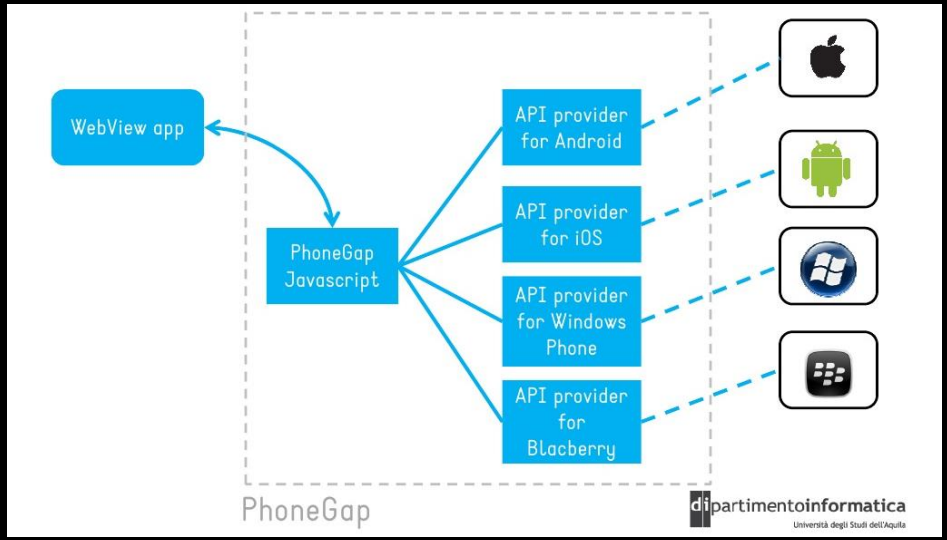
- The UI layer is a webview with 100% width and 100% height and no zoom
- Provides a headless web browser wrapped in a native app container
  - no chrome/safari
  - no url bar
  - no window decorations or back/stop buttons
  - no zooming or text selection
- Uses UIWebView API for iOS and android.webkit.WebView for android

Cordova: Application execution workflow

1. User clicks app icon
2. The app loads from the app manager on native device
3. Webview is initialized starting with cordova.js generating a device ready action and the index.html file in the app code initializing as it would in a browser
4. events/notification happen and execute inside webview as expected
5. events targeting native features execute in javascript as any third party script would
6. cordova.js intercepts device targeting event using plugin js interface
7. the plugin selected executes its native code according to parameters passed in at the js interface from cordova.js
8. native code can invoke any accessible device feature (e.g. write to a file)
9. once native code is done, return values are passed back to the interface, then to cordova.js, then to the webview, and finally to the client-side webapp
10. The webapp handles the event and updates the UI accordingly

Cordova

- You build your mobile app using JS/CSS/HTML or with your favorite JS framework (ember obviously)
- Follows the same architecture from before with the addition of all the native device API plugins you add



Cordova: Supported native features

	android	blackberry10	ios	Ubuntu	wp8 (Windows Phone 8)	windows (8.1, 10, Phone 8.1)	OS X
cordova CLI	✓ Mac, Windows, Linux	✓ Mac, Windows, Linux	✓ Mac	✓ Ubuntu	✓ Windows	✓	✓ Mac
Embedded WebView	✓ (see details)	✗	✓ (see details)	✓	✗	✗	✓
Plugin Interface	✓ (see details)	✓ (see details)	✓ (see details)	✓	✓ (see details)	✓	✓
	Core Plugin APIs						
Accelerometer	✓	✓	✓	✓	✓	✓	✗
BatteryStatus	✓	✓	✓	✓	✓	✓ * Windows Phone 8.1 only	✗
Camera	✓	✓	✓	✓	✓	✓	✗
Capture	✓	✓	✓	✓	✓	✓	✗
Compass	✓	✓	✓ (3GS+)	✓	✓	✓	✗
Connection	✓	✓	✓	✓	✓	✓	✗
Contacts	✓	✓	✓	desktop only	✓	partially	✗
Device	✓	✓	✓	✓	✓	✓	✓
Events	✓	✓	✓	✓	✓	✓	✗
File	✓	✓	✓	✓	✓	✓	✓
File Transfer	✓	✓ * Do not support onprogress nor abort	✓	✗	✓ * Do not support onprogress nor abort	✓ * Do not support onprogress nor abort	✗
Geolocation	✓	✓	✓	✓	✓	✓	✗
Globalization	✓	✓	✓	✓	✓	✓	✗
InAppBrowser	✓	✓	✓	✓	✓	uses iframe	✗
Media	✓	✓	✓	✓	✓	✓	
Notification	✓	✓	✓	✓	✓	✓	✗
Splashscreen	✓	✓	✓	✓	✓	✓	✗
Status Bar	✓	✗	✓	✗	✓	✓ Windows Phone 8.1 only	✗
Storage	✓	✓	✓	✓	✓ localStorage & indexedDB	✓ localStorage & indexedDB	✗
Vibration	✓	✓	✓	✓	✓	✓ * Windows Phone 8.1 only	✗

Cordova: development tips

- Place all css at the beginning of your app to allow progressive rendering
- Place all JS scripts at the end of your <body> tag or follow framework expectations (e.g. ember generates your scripts, css, and index.html files for you and follows these practices)
  - This will speed up initial page render time
- Avoid large images or large files in general
  - use CSS gradients and effects instead
- Avoid complex CSS or JS animations (e.g. text-shadow, box-shadow, opacity transitions)
- Use touch events instead of onClick for UI handling
  - onClick can take 500ms to execute

Cordova: development tips

- If you need an input field, scale the keyboard to open only the needed elements
  - e.g. <input type="text" pattern="[0-9]*" value="numeric"/>
  - this will open just the numeric keyboard – improving the user interface
- Disable user selection, if its not done automatically by a framework you are using
  - <style> * {-webkit-touch-callout: none; -webkit-user-select: none;} </style>
- MINIFY EVERYTHING – so much speed savings here
- don't use large complex libraries with bad benchmarks

Part 2: Lab time – continue with the hybrid lab

https://github.com/MLHale/CYBR8480/blob/master/modules/hybrid-app-tutorial-part2.md

ASK QUESTIONS!

Questions?

Matt Hale, PhD

University of Nebraska at Omaha

Interdisciplinary Informatics

mlhale@unomaha.edu

Twitter: @mlhale_