Ionic Framework Training

Ken Sodemann

- Senior Systems Architect @ Ionic
- @kensodemann - Twitter
- ken@ionic.io

About This Course

- Some Lecture
- Mostly Hands On
- Covers Full Lifecycle at High Level
  - Ionic CLI - Scaffold the App
  - Ionic Framework - Build the App
  - Ionic Pro - Deploy the App
  - Git - Manage the Source

Skills Covered

- Creating a new Ionic application
- Integrating the application with Ionic Pro, including Deploy and Monitor
- Adding and managing third party libraries
- Layout out pages
- Adding new services
- Working with asynchronous code
- Accessing native functionality via Cordova plugins

Day by Day Breakdown

- Day 1
  - Introduction
  - Ionic Pro
  - Begin Weather app
- Day 2
  - Continue Weather App
- Day 3
  - Ionic v4 Conversion Lab
  - Identity Vault and Studio
  - AMA - Ask Me Anything

Before We Begin

- Node.js and NPM
- Git
- https://github.com/ionic-team/framework-training-deck
- Ionic and Cordova
- Xcode and/or Android Studio
- Ionic Pro Account

Ionic CLI Basics

- ionic start - create a new application
- ionic serve - start up a development server for the application
- ionic generate - create a page, service, etc.
- ionic cordova - perform a Cordova related task

Git Overview

- Local Repository
- Remote Repositories
- Branches
  - Master - do not code here
  - Feature branches - do code here

Banch and Merge Workflow

- Create Branch - git checkout -b feature/doSomething
- Add changes - git add / git stage
- Commit - git commit
- Squash and Rebase - git rebase -i origin/master
- Pull Request - push, create pull request
- Merge into master - git checkout master; git merge feature/doSomething

Lab: Generate an Application

Ionic Pro Training

What Is Ionic?

- Components
- Tooling
  - CLI
  - Ionic Pro
- Integrations
  - Ionic Native Wrappers
  - Other Libraries (IV, Storage, Ionicons, etc)

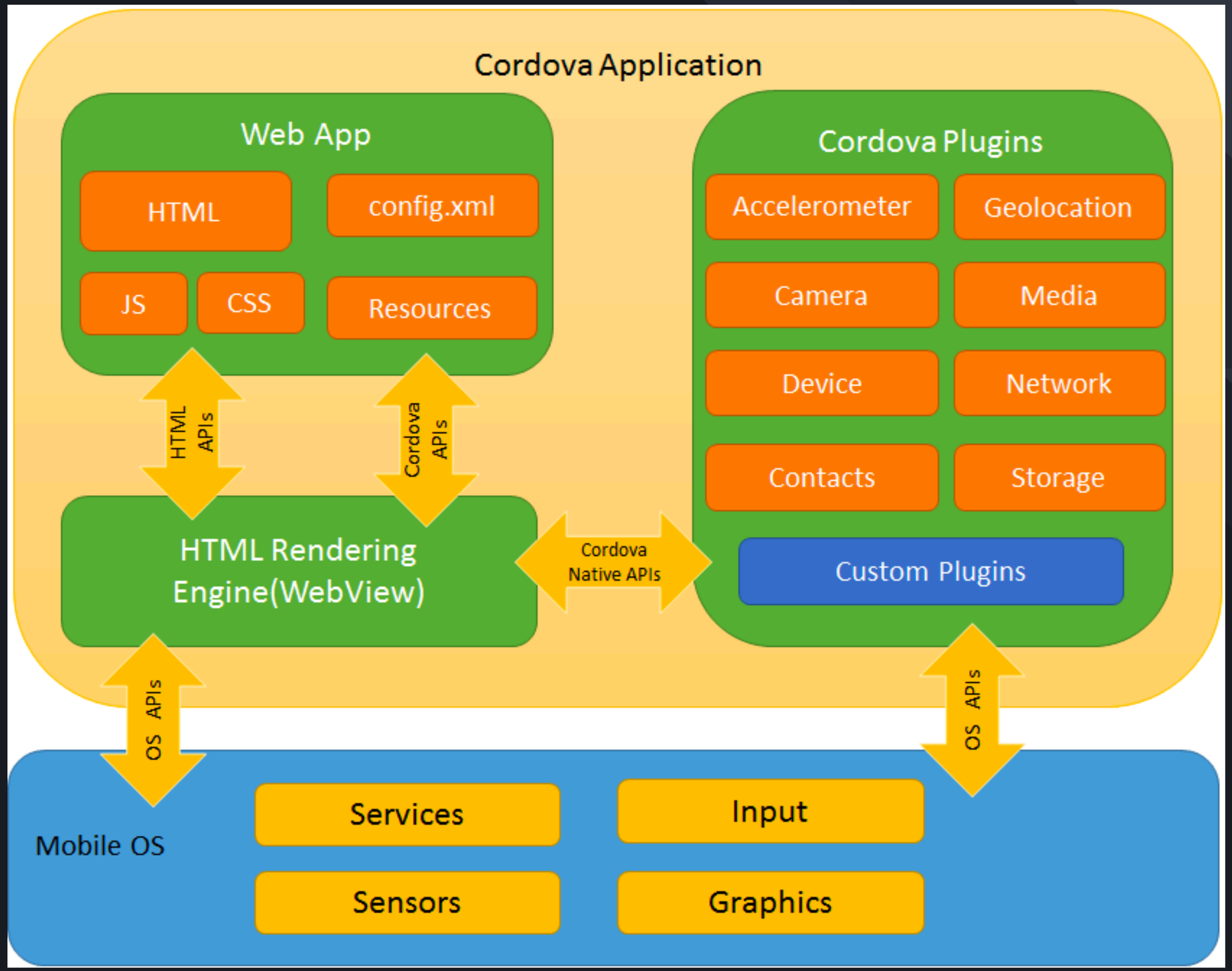
Ionic Uses Web Technologies

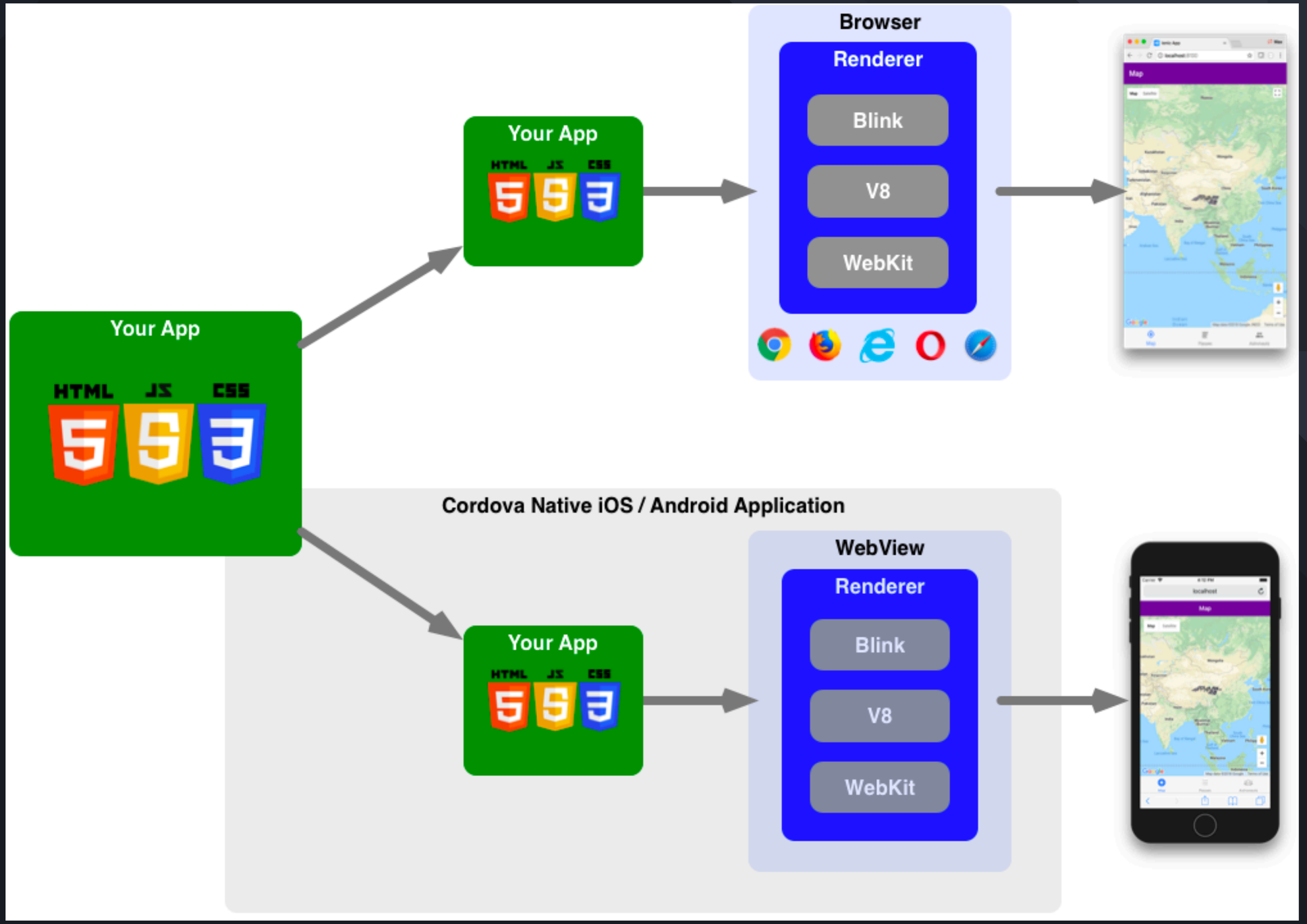
Ionic Applications are written using HTML, CSS, and TypeScript. They are web applications that can run in any “web” context. Thus you can target:

- Hybrid Mobile
- Electron
- PWAs
- Web

Hybrid Mobile Architecture

- Web App
  - HTML
  - CSS
  - JavaScript
  - Assets - images, etc.
- Cordova Plugins


Cordova Application
Web App
HTML
config.xml
JS
CSS
Resources
Cordova Plugins
Accelerometer
Geolocation
Camera
Media
Device
Network
Contacts
Storage
Custom Plugins
HTML APIs
Cordova APIs
HTML Rendering Engine(WebView)
Cordova Native APIs
OS APIs
OS APIs
Mobile OS
Services
Input
Sensors
Graphics


Browser
Renderer
Blink
V8
WebKit
Your App
HTML JS CSS
Your App
HTML JS CSS
Cordova Native iOS / Android Application
WebView
Renderer
Blink
V8
WebKit
Your App
HTML JS CSS

Basic Ionic & Angular Concepts

- Single Responsibility Principle - not an Angular concept, but important
- Components
- Pages - Ionic Concept, Special Components
- Services / Providers
- Pipes

Review: Ionic CLI

- ionic start
- ionic serve
- Ionic generate
- ionic cordova
  - Add platforms
  - Add plugins
  - Build for a platform

Pro Tip: —help gives help, partial commands give options for you

Tour: The Code So Far…

Lab: Layout Pages

Styling

- Uses SASS
- Global Styling - src/app/app.scss
- Theming - variables in src/theme/variables.scss
- Each component and page has its own SCSS file
- Best Practice:
  - Style and theme globally
  - Use separate files and @import
  - Tweak locally and minimally

Lab: Styling

Using Libraries

- NPM - started out as Node Package Manager
- Used to install libraries
- Used to determine libraries to update
- Dependencies managed via package.json file

Dependencies

```
"dependencies": {
  "@angular/animations": "5.2.11",
  ...
  "@ionic-native/core": "~4.12.0",
  "@ionic-native/geolocation": "4.12.2",
  "@ionic-native/splash-screen": "^4.12.0",
  "@ionic-native/status-bar": “~4.12.0"
"devDependencies": {
  "@ionic/app-scripts": "3.2.0",
  "typescript": "~2.6.2"
},
```

Basic semver

- Three digit version - major.minor.patch (ex: 4.3.7)
- Version change signifies type of change
  - Major - breaking changes
  - Minor - new features
  - Patch - bug fixes, refactoring, speed or size enhancements, etc.
- NPM Prefixes
  - None - use specified version
  - ‘~’ - take new patch releases
  - ‘^’ - take new minor releases

Lab: Use a Library

Services / Providers

- Often used to get data
- Business logic goes here
- Instantiated based on Angular’s NgModule system
- Injected via constructors
- Create with CLI: ionic generate provider provider-name

HttpClient

- Default Angular Service for HTTP Requests
- Uses RxJS Observables
- Typical use

```
In Service:
getConfig(): Observable<Config> {
  return this.http.get<Config>(this.configUrl);
}

In Consumer:
showConfig() {
  this.configService.getConfig()
    .subscribe((data: Config) => this.config = { ...data });
}
```

Models

- Define the structure of your data
- Can be classes, interfaces, types, etc

```
export interface Weather {
  temperature: number;
  condition: number;
  date?: Date;
}

export type Forecast = Array<Array<Weather>>;
```

Observables

- Like Lodash for events
- Angular uses RxJS Library

```
getEmployee(id: number): Observable<Employee> {
  return this.http
    .get(`${this.baseUrl}/employees/${id}`)
    .pipe(map((res: any) => this.unpackEmlpoyee(res)));
}

Caller:
this.data.getEmployee(42).subscribe(e => this.employee = e);
```

NOTE: If getEmployee() is called without the subscribe, the HTTP call will not be made.

Lab: Getting the Data

Lab: Using the Data

Promises

An object that represents the eventual completion of an asynchronous event

```
function foo () {
  return new Promise((resolve, reject) => {
    setTimeout(() => resolve('foo'), 1000);
  });
}

function bar () {
  return foo().then(x => `${x}bar`);
}

function wut () {
  return bar().then(x => foo());
}

wut().then(x => console.log(`wut: ${x}`));
foo().then(x => console.log(x));
bar().then(x => console.log(x));
console.log('baz');
```

Async / Await

- Makes async code more synchronous
- Can be very helpful, but need to use with care

```
async function writeIt () {
  console.log(`wut: ` + await wut());
  console.log(await foo());
  console.log(await bar());
  console.log(`baz`);
}

wut().then(x => {
  console.log(`wut: ${x}`));
  return foo().then(x => {
    console.log(x));
    return bar().then(x => {
      console.log(x));
      console.log('baz');
    }
  }
}
```

Platform Service

- this.platform.ready().then() - most common usage
- Information about the device
- Information about runtime environment

Cordova Plugins

- JavaScript interface to native functionality and APIs
- Like ogres and onions, plugins have layers
  - JavaScript layer
  - Java Layer - Android
  - Objective C Layer - iOS
  - Other layers for other devices

Lab: Using Plugins

Modals

- ModalController
- ViewController
- Many overlays follow this same pattern

```
export class ModalComponent {
  constructor(private modal: ViewController) {}

  dismiss() {
    this.modal.dismiss();
  }
}

Caller:
openModal() {
  const m = this.modal.create(ModalComponent);
  m.present();
}
```

Lab: User Preferences Modal

Subjects

- Similar to Observables
- Like EventEmitters, maintains multiple subscribers

```
class EmittingService
  changed: Subject<void>;

  constructor(private storage: Storage) {
    this.changed = new Subject();
  }

  foo() {
    this.changed.next();
  }
}
```

Lab: Finish User Preferences

Lazy Loading

- Each page in own module
- Each page needs IonicPage() decorator
- No direct references to pages
- Only works at the page level (in Ionic v3)
- Essentially all-or-nothing

Lab: Lazy Loading

Ionic Version 4

- Standards Based - Web Components, Shadow DOM
- Lazy Loaded by Default
- Smaller Angular Footprint
- Aligned with Angular CLI Projects

Changes in the Ionic Framework

- ionic-angular becomes @ionic/core and @ionic/angular
- No more app-scripts, uses the NG CLI
- Defaults to Angular Version 6
- Tight integration with the Angular component router
- Angular directives become first-class elements
- Shadow DOM - CSS Custom Properties
- Unit Testing and End-to-end Testing Infrastructure

Areas of Largest Change

- Project Structure and Tooling
- Navigation
- Styling - Shadow DOM and Custom Properties
- HTML Markup - Directives Becoming Elements

Suggested Upgrade Path

- Generate a New Project - new project structure and build tools
- Create a foundation - configure libraries, plugins, etc.
- Copy the pure code - copies over with minor modifications
- Copy the components and pages - use the lint tool to help you
- Apply the styles and themes
- Copy to clean branch in your project’s repo

Ionic Resources

- https://beta.ionicframework.com/docs
- https://beta.ionicframework.com/docs/building/migration
- https://github.com/ionic-team/ionic/blob/master/angular/BREAKING.md
- https://github.com/ionic-team/v4-migration-tslint

Let’s Upgrade Ionic Weather