Let's build an Angular JS app from scratch!

Alain Chautard
www.interstate21.com

What are we about to do?

- We are going to build a small weather application with Angular JS 1.4, Bootstrap and Yeoman

- This project was inspired to me by the Udacity course: *Developing Android Apps: Android Fundamentals*

- What took hours to get done with Android can be done in less than a couple hours with Angular!

Before we get started...

- All questions are welcome! Feel free to interrupt
- The source code of what we're about to do can be found here: https://github.com/alcfeoh/angular-weather
- Each step is tagged as follows: step1, step2, etc.

Step 0: Development environment

- We use Yeoman to scaffold our app: http://yeoman.io/
- Yeoman generates code for us, makes everything faster
- It also generates unit tests stubs
- Uses Grunt and Bower to automate your build, have an auto-refresh dev environment, JS lint your code, etc.

Step 0: Development environment

- `yo angular AngularWeather`
- Generates a basic Angular JS web project
- `grunt serve`
- Starts an auto-refresh dev server with Node JS
- Let’s change the toolbar text to “Weather App”

Step 1: “Add a location” feature

- Let’s do a bit of clean-up and remove the code that we won’t be using (toolbar, etc.)
- Update the user interface and add an input text asking for a zipcode
- Update our MainCtrl and add a “addLocation” method with zipcode as a parameter
- Update our main view to show our location list

Step 2: Get data from OpenWeatherMap API

- Here is a sample API call: http://api.openweathermap.org/data/2.5/weather?zip=95814,us&units=imperial
- Let's make a $http call to load that data as soon as we add a new location
- Update the HTML code to show weather information
- Add a "remove location" method to our MainCtrl
- Display a "remove location" button for each location

Step 3: Add a weather icon

- Here is the list of all weather conditions from our API: `http://openweathermap.org/weather-conditions`
- Let's add a getWeatherIcon function that returns the URL of the image to use based on a forecast parameter
- Update the HTML code to show the icon
- Test it with several zipcodes (10000, 95835, 99501)

Step 4: Let's refactor and create a directive

- `yo angular:directive weatherWidget`
- Move the HTML code to a specific template file in "views" folder
- Update directive code accordingly
- Add a controller with a local "getWeatherIcon" function
- Add params to the directive (location and on-delete callback)

Step 5: Let's write a WeatherService

- `yo angular:service WeatherService`
- Add a HTTP resource to that service
- Update the HTTP call in MainCtrl (there's a slight syntax difference)
- That's it! We have a small functional Angular app!

The End

Enter our raffle for a JetBrains software license at:

https://raffleapp.gdgsacramento.jit.su/raffle

Google DevFest on Sunday, October 18th
gdgsacramento.org