Code Norman Angular Workshop

Table of Contents

Code Norman Angular Workshop

Installations

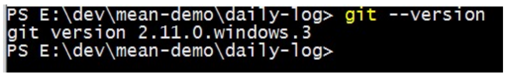
git

git will be used for version control. Installation is here: https://git-scm.com/download/win

To verify git installation, open a command prompt

```
git --version
```

If successful, the result will display the git version installed.



```
PS E:\dev\mean-demo\daily-log> git --version
git version 2.11.0.windows.3
PS E:\dev\mean-demo\daily-log>
```

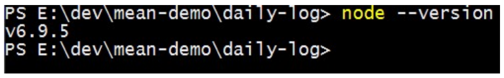
NodeJS

Download and install the LTS (Long Term Support) version of nodejs from https://nodejs.org (version 6.9.5 is used in the tutorial)

To verify node installation, open a command prompt

```
node -v
```

If successful, the result will be the version number installed.



```
PS E:\dev\mean-demo\daily-log> node --version
v6.9.5
PS E:\dev\mean-demo\daily-log>
```

npm update

npm is the Node Package Manager. This is used to install projects dependencies.

The Node install includes a version of npm, but npm is its own project and upgraded on its own schedule. Upgrade to the latest version of npm. The -g parameter will install npm globally. Available in all projects.

```
npm install -g npm
```

Postman

Download the postman installer from https://www.getpostman.com/ and start the application. This tool can send http requests (get, post, put) and view the response.

npm Global Dependences

Some npm modules are installed globally to make them available for all projects.

The project may also install a local version as a dependency. The global version make the commands available on the command line and the local dependency is used for the specific project. This allows each project to have a dependency on a different version.

angular-cli

```
npm install -g @angular/cli@1.0.0-beta.31
```

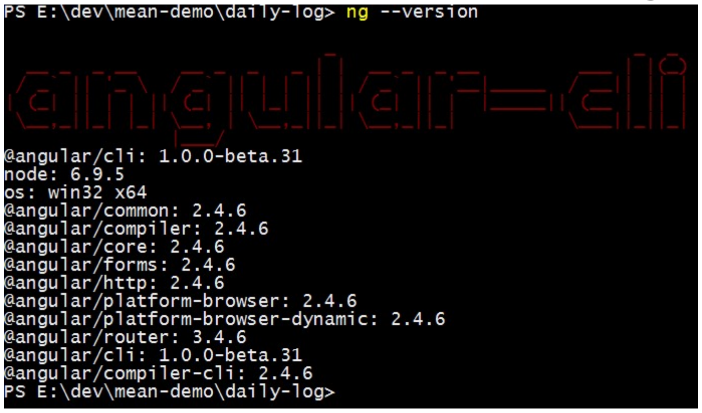
Note: installation of a specific version of the angular-cli to make sure compatibility with this tutorial.

npm install -g @angular/cli@latest to install the latest version.

To verify angular-cli installation, open a command prompt

```
ng -v
```

If successful, it will show version information.



```
PS E:\dev\mean-demo\daily-log> ng --version

@angular/cli: 1.0.0-beta.31
node: 6.9.5
os: win32 x64
@angular/common: 2.4.6
@angular/compiler: 2.4.6
@angular/core: 2.4.6
@angular/forms: 2.4.6
@angular/http: 2.4.6
@angular/platform-browser: 2.4.6
@angular/platform-browser-dynamic: 2.4.6
@angular/router: 3.4.6
@angular/cli: 1.0.0-beta.31
@angular/compiler-cli: 2.4.6
PS E:\dev\mean-demo\daily-log>
```

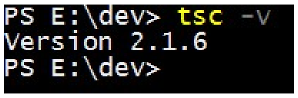
Typescript

Typescript is a typed superset of JavaScript that compiles to plain JavaScript

```
npm install -g typescript
```

To verify typescript installation, on a command prompt

```
tsc -v
```



```
PS E:\dev> tsc -v
Version 2.1.6
PS E:\dev>
```

Code Editor

Install the editor of your choice. Some options are:

Jetbrain's WebStorm

Microsoft Visual Studio

Microsoft Visual Studio Code

Atom

Eclipse

Augury

Augury is a Chrome Extension for debugging Angular 2 applications.

What is Angular?

Angular is a platform, created by Google, optimized for writing both mobile and web applications. Building components that extend HTML through creation of new elements and attributes. For example, instead of creating a div section for adding a User, create a <user></user> element.

An application = component + component + component + component

A component = template + Class (properties + methods) + metadata

The Angular-cli is a command-line tool that generates the basic plumbing of the application. The cli has an upgrade process to help upgrade to future versions of Angular.

Angular components follow the W3C Web Component Specification https://www.w3.org/standards/techs/components#w3c_all. The four specifications are:

Custom Elements: This document describes the method for enabling the author to define and use new types of DOM elements in a document.

HTML Imports: This document defines a way to include and reuse HTML documents in other HTML documents.

HTML Templates: Describes a method for declaring inert DOM subtrees in HTML and manipulating them to instantiate document fragments with identical contents

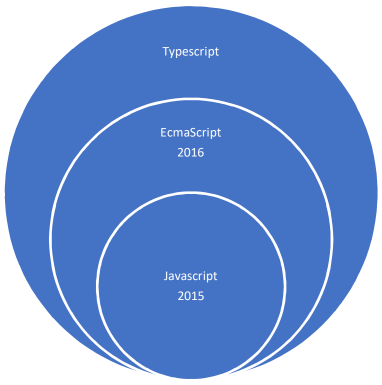
Shadow DOM: Describes a method of establishing and maintaining functional boundaries between DOM subtrees and how these subtrees interact with each other within a document tree.

Angular can be written using different languages. TypeScript, a superset of JavaScript, ES2016, JavaScript 2015 and Dart. TypeScript will be used and currently the language supported by the angular-cli.

TypeScript Concepts

TypeScript is a superset of ECMAScript[1] and is transpiled into ECMAScript. TypeScript Playground (https://www.typescriptlang.org/play/index.html) shows TypeScript (left) and the resulting JavaScript (right).

In addition to Typescript features, ECMAScript ES2017 and later features have been incorporated. This allows these future standards to be used today. TypeScript will transpile to today's ECMAScript until browsers support these future language features.



Types

JavaScript is an untyped language. Changing the type of a variable at runtime is allowed. TypeScript can verify that the type is not changed. This verification is completed at compile time, therefore, there is no performance penalty at runtime. This can avoid programmer error where a string is passed when a number is expected, resulting in more reliable software.

Typescript typing includes creation of custom types.

```
// Javascript
var x = 42;
x = "Helloworld";  // valid JavaScript
```

```
// TypeScript
var x:number = 42;  // x is defined as number
x = "HelloWorld"; // compile time error: Type "HelloWorld" is not assignable to
type 'number'
```

Classes

TypeScript introduces classes.

```
class Person {
   lastname: string;
```

[1] JavaScript was renamed to EcmaScript

```
   firstname: string;
   constructor (lastname: string, firstname:string){
         this.lastname = lastname;
         this.firstname = firstname;
      }
}
```

Decorators

A decorator (or annotation) can be added to one of the following types: Class, Property, Method, and Parameter.

A decorator example that adds @log before the method myMethod. The @log decorator adds logging messages to myMethod.

```
class MyClass{
   @log()
      myMethod(){
   }
}
```

Decorators are powerful and can add reusable features throughout the application or multiple applications.

Decorator Examples:

@readonly: Property Decorator. Makes the property not writable.

@debounce(wait): Method Decorator: Makes the method invoked after wait milliseconds (default 300) since the last time it was invoked.

@throttle (wait): Method Decorator: Makes the method invoked every wait (300ms default) milliseconds.

@time: Method Decorator: Times how long it takes a method to execute.

Generics

A generic allows generation of an object of a specific type. It is a template for types.

```
// Observable
var daily-log:Observable<DailyLog> // Observable of DailyLog Type
```

Instead of creating a SortInt, SortString, SortDate, etc… A Sort generic can be created. Using Sort<int> can limit to integers. It also allows custom types Sort<User>.

Angular Basics

Modules

An Angular module is used to package related Directives, Components and Services into a package for inclusion in an application.

Components

A component is a special Class with HTML visual properties.

Templates

View part of a component.

Metadata

Configuration data

Data Bindings

Interpolation, property, event and 2-way binding

Directives

Add a new behavior to HTML.

Services

A service is a singleton that provides common shared object to multiple pieces of the application. Dependency Injection is used.

For example, A database service:

Component1: connect to database

Component2: connect to database

Dependency Injection

Pattern used by Angular to provide services into other objects. A database service does the connection to the database and provides methods to get data. Inject this into a User component. The user component calls getUsers, but doesn't have to know how to connect to the database. The user component only needs to know the getUsers method, its parameters and response.

The database service can be change to a different database or a mock that responds the same way the database does for testing.

Routing

Routing is used for Single Page Applications. Using the URI location stay on the page, but route to a different component.

www.myap.com/home routes to home component

www.myapp.com/users routes to the users component

It is also possible to nest routes. The user component may have own navigation to view, edit, list users.

Project Setup

Angular

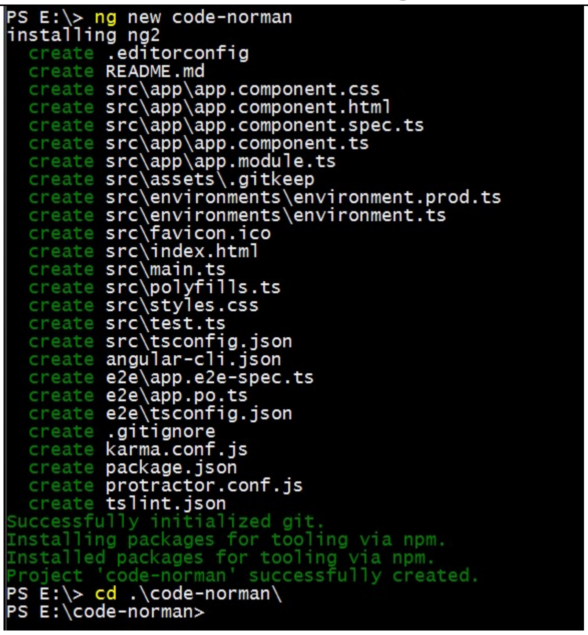
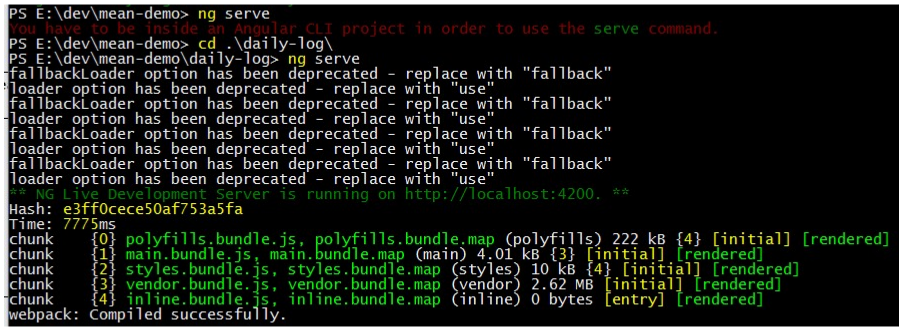
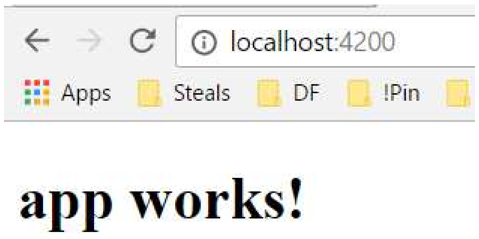
Have an issue or want to start from here checkout branch

```
git checkout cors
cd server
npm install
npm run dev
```

Using the angular-cli will create quick work of generating the application.

Project Setup

Initialize angular application using angular-cli. The cli will create the directory, add files and	**`ng new code-norman`** The cli will install files and run npm for the project This will take a few minutes.

perform npm install for dependencies.	 (terminal output below)
Enter created project	`cd code-norman`
Test application	`ng serve` the cli will start the app with a development server. This server is not intended to be used in production.  (terminal output below) Once build the application will be available at **localhost:4200**
Test in browser	`Open chrome and enter the url` http://localhost:4200 You should see **app works!** in the browser. 

```
PS E:\> ng new code-norman
installing ng2
  create .editorconfig
  create README.md
  create src\app\app.component.css
  create src\app\app.component.html
  create src\app\app.component.spec.ts
  create src\app\app.component.ts
  create src\app\app.module.ts
  create src\assets\.gitkeep
  create src\environments\environment.prod.ts
  create src\environments\environment.ts
  create src\favicon.ico
  create src\index.html
  create src\main.ts
  create src\polyfills.ts
  create src\styles.css
  create src\test.ts
  create src\tsconfig.json
  create angular-cli.json
  create e2e\app.e2e-spec.ts
  create e2e\app.po.ts
  create e2e\tsconfig.json
  create .gitignore
  create karma.conf.js
  create package.json
  create protractor.conf.js
  create tslint.json
Successfully initialized git.
Installing packages for tooling via npm.
Installed packages for tooling via npm.
Project 'code-norman' successfully created.
PS E:\> cd .\code-norman\
PS E:\code-norman>
```

```
PS E:\dev\mean-demo> ng serve
You have to be inside an Angular CLI project in order to use the serve command.
PS E:\dev\mean-demo> cd .\daily-log\
PS E:\dev\mean-demo\daily-log> ng serve
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
** NG Live Development Server is running on http://localhost:4200. **
Hash: e3ff0cece50af753a5fa
Time: 7775ms
chunk    {0} polyfills.bundle.js, polyfills.bundle.map (polyfills) 222 kB {4} [initial] [rendered]
chunk    {1} main.bundle.js, main.bundle.map (main) 4.01 kB {3} [initial] [rendered]
chunk    {2} styles.bundle.js, styles.bundle.map (styles) 10 kB {4} [initial] [rendered]
chunk    {3} vendor.bundle.js, vendor.bundle.map (vendor) 2.62 MB [initial] [rendered]
chunk    {4} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.
```

Review of the Application Structure

e2e	End 2 end testing
node_modules	Dependencies.
src	application source. **Most changes will occur here.**
.editorconfig	Editor preferences
.gitignore	Files and directories for git to ignore
angular-cli.json	angular-cli configuration information
karma.config.js	Karma configuration
package.json	npm project configuration
protractor.conf.js	Protractor e2e configuration
README.md	Markdown file containing instructions from angular-cli
tslint.json	Typescript linter configuration

src directory

app	Contains the app component. Other components will be created under this directory
assets	For any files
environments	Configure settings for different environments (development and production are default environments)

Review index.html

Index.html	Open src/index.html in editor In the <body> element is an element <app-root>. The <app-root> element is where angular will put the first component. The loading… statement will be displayed while the application loads.

Commit work to git

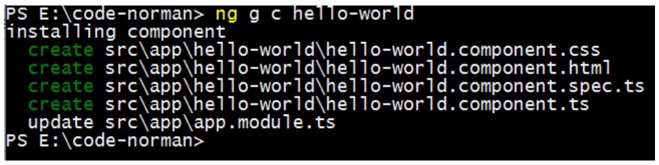
create new branch angular has already committed files to master branch. Creating a branch will to start from this point in the future.	**`git checkout -b angular-setup`**

Create First Component

Have an issue or want to start from here checkout branch

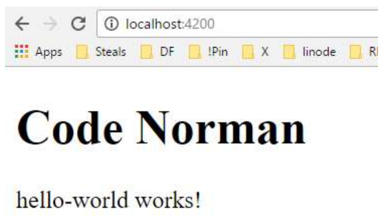
```
git checkout angular-setup
cd server
npm install
new command line
cd code-norman
ng serve
```

Using the angular cli generate a hello world component.

Open another command line	2 command prompt windows opened One for the angular development server One to perform angular-cli commands
Cd into daily-log project	`cd code-norman`
Generate hello-world component	`ng generate component hello-world`  `PS E:\code-norman> ng g c hello-world` `installing component` `  create src\app\hello-world\hello-world.component.css` `  create src\app\hello-world\hello-world.component.html` `  create src\app\hello-world\hello-world.component.spec.ts` `  create src\app\hello-world\hello-world.component.ts` `  update src\app\app.module.ts` `PS E:\code-norman>`
Review what has been generated	In editor open src/app/hello-world/hello-world.component.ts
import Component & OnInit from @angular.core	`import { Component, OnInit } from '@angular/core';`
@Component is a decorator (annotation). The 3 parameters are:	`@Component({`
selector: HTML tag used in the HTML file.	`  selector: 'app-hello-world',`
templateUrl: path to the html partial to replace the selector.	`  templateUrl: './hello-world.component.html',`
styleUrls: an array of style files. In this case only one style specified.	`  styleUrls: ['./hello-world.component.css']` `})` `export class HelloWorldComponent implements OnInit {` `  constructor() { }` `  ngOnInit() {` `  }` `}`
Note: These files are relative to the hello-world-component.ts file.	
Open template	In the editor open the src/app/hello-world/hello-world.component.html file
A simple template with a `<p>` tag.	`<p>` `  hello-world works!`

	`</p>`
Styles	The src/app/hello-world/hello-world.component.css is empty, but can contain styles specific to this component.
Spec	Spec is used to write tests.
Open browser	Open browser to **localhost:4200**

Add component to application.	Keep ng serve and the browser open.
	Edit /src/app/app.component.html and add the app-hello-world selector for the component which is
	`<h1>`
	`  {{title}}`
	`</h1>`
	`<app-hello-world></app-hello-world>`
	ng serve will notice the file change and recompile the application. The browser will be automatically refreshed showing changes.
	This process is very productive. With two monitors. One for the code window and one for the browser. Edit Code, save, confirm changes. Commit changes and start next feature.

Let's modify the title.	open code-norman/src/app/app.component.ts
	
	on about line 9 from
	`title = 'app works!'`
	to
	`title = 'CODE Norman'`
	ng server will recompile the application and the browser will automatically refresh
	 localhost:4200
	Apps
	Steals
	DF
	!Pin
	X
	linode
	Ri
	Code Norman
	hello-world works!

Commit work to git

create new branch	`git checkout -b hello-world`
add new files to git	`git add *`
commit changes	`git commit -m "added hello world component"`

Add Project Dependencies

Have an issue or want to start from here checkout branch

```
git checkout hello-world
cd server
npm install
new command line
cd code-norman
ng serve
```

Install primeng and font-awesome PrimeNG is a collection of UI Components for Angular 2 http://www.primefaces.org/primeng These components will be used in the application being built. font-awesome is used by PrimeNG.	**`npm install --save primeng font-awesome`**

Add primeng styles	Open code-norman/src/angular-cli.json and update the styles(~ line 21) as follows
	`"styles": [`
	`  "styles.css",`
	`"../node_modules/primeng/resources/themes/ludvig/theme.css",`
	`  "../node_modules/font-awesome/css/font-awesome.min.css",`
	`  "../node_modules/primeng/resources/primeng.min.css"`
	`],`

Import PrimeNG components that are going to be used into our app.module.ts file	Open code-norman/src/app.module.ts Imports ~Line 5 add **`import {PanelModule} from 'primeng/primeng';`** Inject imports ~line 20 (after HttpModule) **`, PanelModule`**
Note: ng server does not restart based on changes to the angular-cli.json file.	**`stop and restart ng serve`**

Commit work to git

create new branch	**`git checkout -b add-dependencies`**
add new files to git	**`git add *`**
commit changes	**`git commit -m "project dependencies"`**

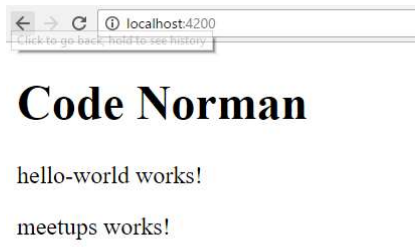
Code Norman Angular Workshop

Meetup component

Have an issue or want to start from here checkout branch

```
git checkout add-dependencies
cd code-norman
npm install
ng serve
new command line
cd code-norman
```

Hello-world component is static. Just an overview of the basic component structure. Let's build the component that displays upcoming meetups.

Generate component	`ng generate component meetups`
Angular-cli will rebuild the app, but the component has only been created and is not used. Let's add it to the app.	Open code-norman/src/app/app.component.html and add the following to the end of the file, after <app-hello-world> `<app-meetups></app-meetups>`  localhost:4200 Click to go back, hold to see history Code Norman hello-world works! meetups works!
Let's change the meetups class to hold the properties of a meetup. For now, the data will be initialized in our component. Later we will request the upcoming meetups from the meetup site api.	First let's create a Class (custom type) to hold the data for a meetup. In the free command prompt `ng generate class meetup`

Add properties to the meetup definition.
 	
properties:
 	
constructor to create a new meetup
	Open the new file code-norman/src/meetup.ts
	
	`export class Meetup {`
	`  name:string;`
	`  link:string;`
	`  description:string;`
	`  constructor(name, link, description) {`
	`    this.name = name;`

```
    this.link = link;
    this.description = description;
  }
}
```

Modify the meetups component to hold an array of meetups and initialize two meetups.

Open code-norman/src/app/meetup/meetup.component.ts and modify as follows:

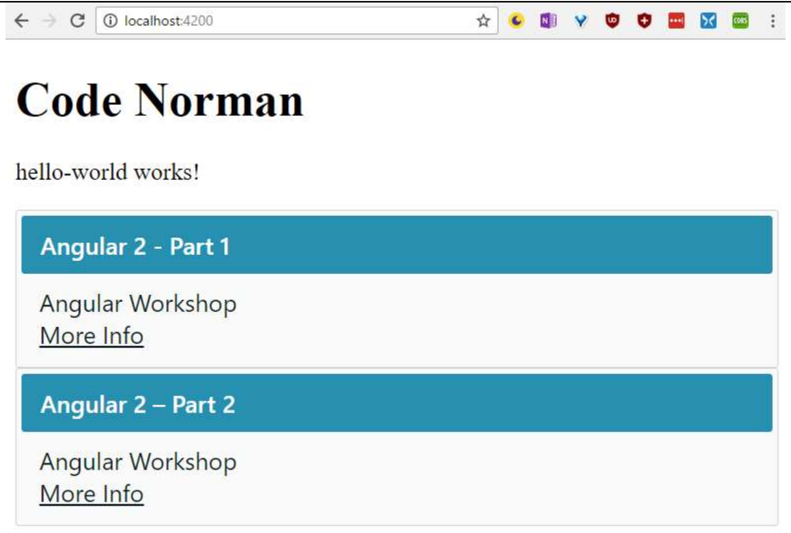
```
import { Component, OnInit } from '@angular/core';
import {Meetup} from '../meetup';

@Component({
  selector: 'app-meetups',
  templateUrl: './meetups.component.html',
  styleUrls: ['./meetups.component.css']
})
export class MeetupsComponent implements OnInit {
  meetups: Meetup[];
  constructor() { }

  ngOnInit() {
    this.meetups = []; // initialize array
    this.meetups.push(new Meetup('Angular 2 - Part 
1','http://meetup.com', 'Angular Workshop'));
    this.meetups.push(new Meetup('Angular 2 – Part 
2','http://meetup.com', 'Angular Workshop'));
  }
}
```

Let’s modify the component view to display the meetups.

Open code-norman/src/app/meetup/meetup.component.html and replace the text with the following

```
<p-panel *ngFor="let meetup of meetups" 
header="{{meetup.name}}">
  <div innerHtml="{{meetup.description}}"></div> <a 
href="{{meetup.link}}">More Info</a>
</p-panel>
```

Save files and view on browser

	 localhost:4200 Code Norman hello-world works! Angular 2 - Part 1 Angular Workshop More Info Angular 2 – Part 2 Angular Workshop More Info

Commit work to git

create new branch	`git checkout -b meetups`
add new files to git	`git add *`
commit changes	`git commit -m "added meetup component"`

Create Meetup Service

Have an issue or want to start from here checkout branch

```
git checkout meetups
cd code-norman
npm install
ng serve
new command line
cd code-norman
```

Create a service that will load the meetups. This will call the api in future steps but for now we will move the two meetups being created.	**`ng generate service meetup`**
Move the initialization of meetups to a service	Open code-norman/src/app/meetups/meetups.component.ts Cut the code in the ngOnInit function to the clipboard. Open code-norman/src/meetup.service.ts and paste the code into the constructor & add the additional lines in bold. `import { Injectable } from '@angular/core';` **`import {Meetup} from './meetup';`**

```
@Injectable()
export class MeetupsService {
  meetups: Meetup[];
  constructor() {
    this.meetups = []; // initialize array

    this.meetups.push(new Meetup('Angular 2 - Part 
1','http://meetup.com', 'Angular Workshop'));
    this.meetups.push(new Meetup('Angular 2 – Part 
2','http://meetup.com', 'Angular Workshop'));
  }
  futureMeetings(){
    return this.meetups;
  }
}
```

Use the service

Open code-norman/src/app/meetups/meetups.component.ts again and add the lines in bold

Import the MeetupService.

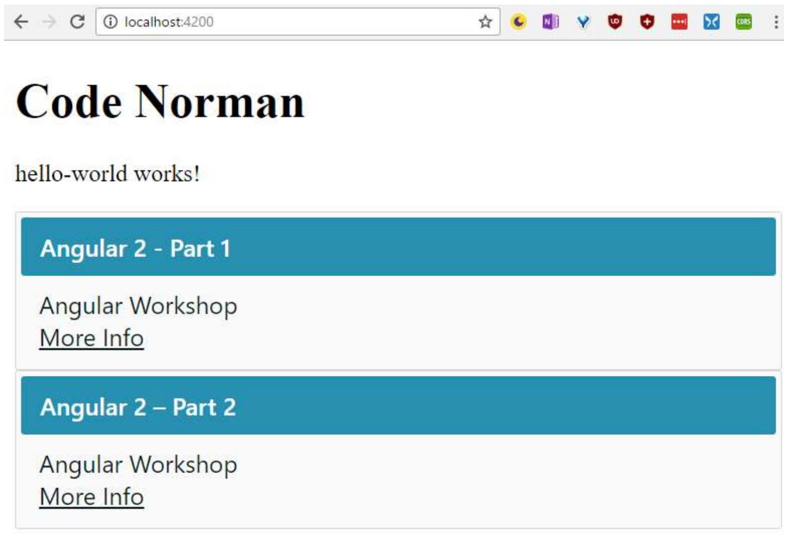
A service is a type of provider.

Let the @Component decorator know about the provider.

Create a private property for the service in the constructor

Call the meetupsService to load the meetups on

```
import { Component, OnInit } from '@angular/core';
import {Meetup} from '../meetup';
import {MeetupService} from "../meetup.service";

@Component({
  selector: 'app-meetups',
  templateUrl: './meetups.component.html',
  styleUrls: ['./meetups.component.css'],
  providers: [MeetupService]
})
export class MeetupsComponent implements OnInit {
  meetups: Meetup[];
  constructor(private meetupService: MeetupService) { }

  ngOnInit() {
    this.meetups = this.meetupService.futureMeetings()
  }

}
```

Initialization of the component	
Save files	There is no visual change in the application. We just created a service in preparation to call the meetups api.  localhost:4200 Code Norman hello-world works! Angular 2 - Part 1 Angular Workshop More Info Angular 2 – Part 2 Angular Workshop More Info

Commit work to git

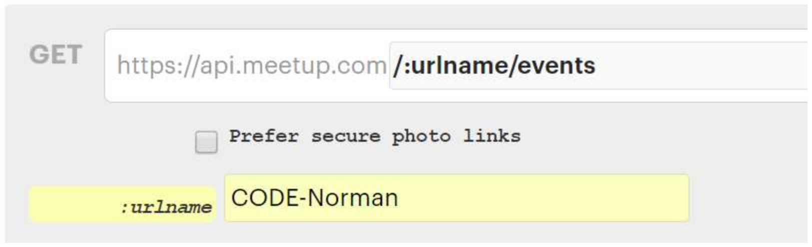
create new branch	`git checkout -b meetup-service`
add new files to git	`git add *`
commit changes	`git commit -m "added meetup service"`

Get Meetup API Key

Have an issue or want to start from here checkout branch

```
git checkout meetup-service
cd code-norman
npm install
ng serve
new command line
cd code-norman
```

Get Meetup API Key	**Goto https://secure.meetup.com/meetup_api/key/** **And Generate a Key if one doesn't show.** **Goto https://secure.meetup.com/meetup_api/console/?path=/:urlname/events** **In the first :urlName field type CODE-Norman**



GET https://api.meetup.com /:urlname/events

☐ Prefer secure photo links

:urlname CODE-Norman

```
Click the Show Response button and copy the Signed URL
```

Call the meetups api using http.get.

http.get will return an observable. This will be subscribed to in the component.

Open code-norman/src/meetups.service.ts

after export class line (~line 5)

```
  private meetupURL = '<YOUR SECURE API URL>';
```

remove the **meetups: Meetup[];**

Import Observable

~Line 2

```
import {Http, Response} from '@angular/http';import
{Observable} from 'rxjs/Rx';
```

Import map & catch 2 of the Observable functions

after rest of imports (~line 4)

```
import 'rxjs/add/operator/map';
import 'rxjs/add/operator/catch';
```

Add the URL to the called

Inject Http

Create private http property and **remove initialization of meetups**;

```
constructor(private http:Http){}
```

http.get returns an observable

map Response to json

map to extract just the name, link and description from the http.get request. There are other fields

Change futureMeetups to call http

```
futureMeetings(): Observable<Meetup[]> {
    return this.http.get(this.meetupURL)
      .map((res: Response) => res.json())
      .catch((error: any) =>
Observable.throw(error.json().error || 'Server Error'))
  }
```

Update meetups component to use the observable results

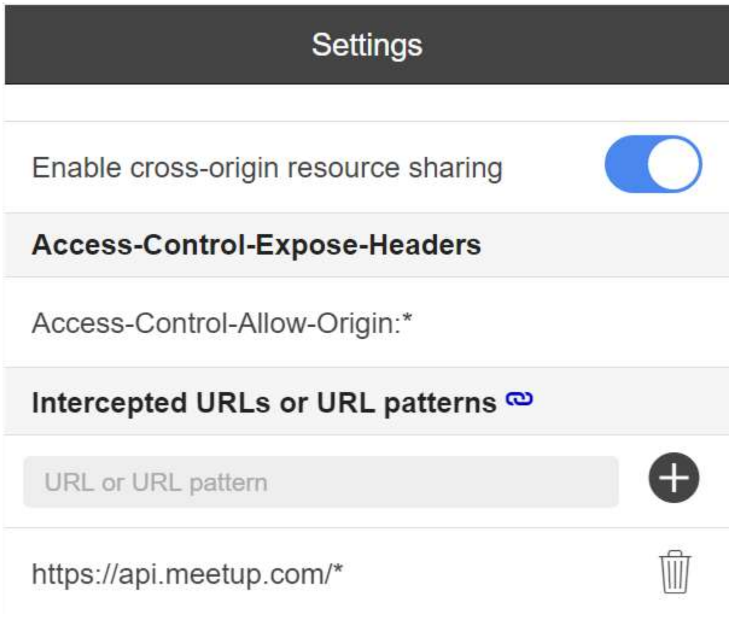
Open code-norman/src/meetups/meetups.component.ts

Change ngOnInit to subscribe to the Observable service

```
ngOnInit() {
    this.meetupService.futureMeetings()
      .subscribe( meetups => {
        this.meetups = meetups;
      },
      err => {
        console.log(err)
      })
  }
```

Download and install Chrome Extension: Allow-Control-Allow-Origin: *

Configure as follows



Settings

Enable cross-origin resource sharing

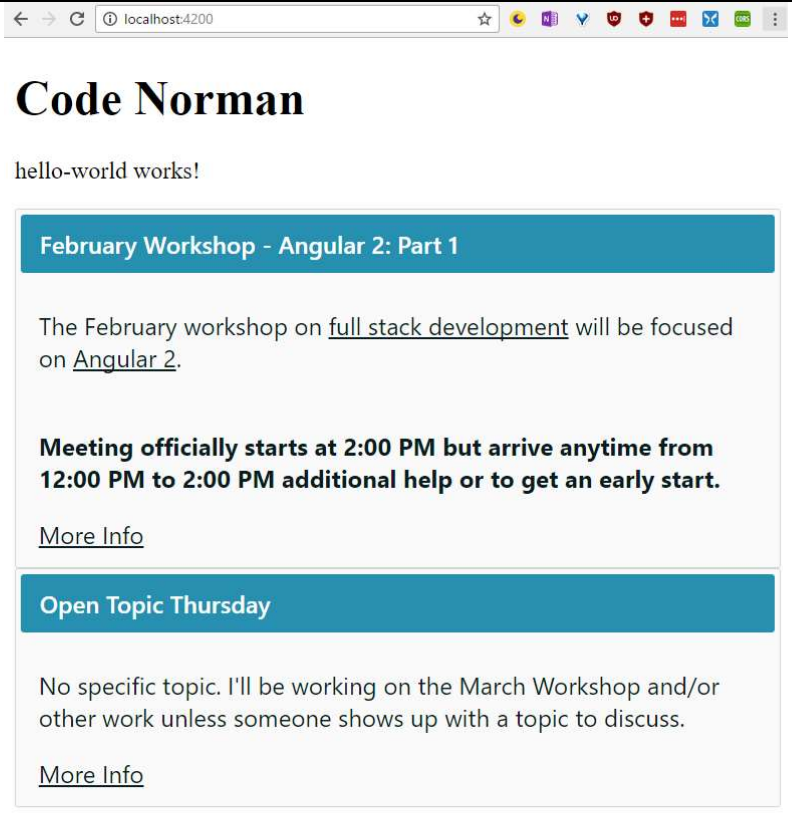
Access-Control-Expose-Headers

Access-Control-Allow-Origin:*

Intercepted URLs or URL patterns

URL or URL pattern

https://api.meetup.com/*



localhost:4200

Code Norman

hello-world works!

February Workshop - Angular 2: Part 1

The February workshop on full stack development will be focused on Angular 2.

Meeting officially starts at 2:00 PM but arrive anytime from 12:00 PM to 2:00 PM additional help or to get an early start.

More Info

Open Topic Thursday

No specific topic. I'll be working on the March Workshop and/or other work unless someone shows up with a topic to discuss.

More Info

Commit work to git

create new branch	`git checkout -b meetup-http`
add new files to git	`git add *`
commit changes	`git commit -m "add meetup http request"`

Create meetup-suggestion Component

Have an issue or want to start from here checkout branch

```
git checkout meetup-http
cd code-norman
npm install
ng serve
new command line
cd code-norman
```

Generate a Meetup Suggestion Component, Service, and Class	`ng generate component meetup-suggestion`
	`ng generate service meetup-suggestion`
	`ng generate class meetup-suggestion`

Edit the meetup-suggestion class
 	
This will contain information on meetup suggestions that members can vote up.
It extends a meetup class.
 	
super must be called
	Open the code-norman/src/app/meetup-suggestion.ts
	`import {Meetup} from './meetup';`
	
	`export class MeetupSuggestion extends Meetup {`
	`  willPresent: boolean;`
	`  email: string;`
	`  votes: number;`
	
	`  constructor(name:string, description:string, willPresent:boolean, email:string) {`
	`    super(name,null,description);`
	`    this.willPresent = willPresent;`
	`    this.email = email;`
	`    this.votes = 0;`
	`  }`
	`}`

Edit Service
 	
Add a few functions add and AddVote that will be used later
	Open the code-norman/src/app/meetup-suggestions.service.ts
	`import {Injectable} from '@angular/core';`
	`import {MeetupSuggestion} from "./meetup-suggestion";`
	
	`@Injectable()`
	`export class MeetupSuggestionService {`
	`  suggestions: MeetupSuggestion[];`

	```
	  constructor() {
	    this.suggestions = [
	      new MeetupSuggestion('Docker', 'A workshop on Docker', true, ''),
	      new MeetupSuggestion('lets encrypt', 'A workshop on lets encrypt', false, ''),
	      new MeetupSuggestion('Linux command  line', 'workshop on bash', false, ''),
	      new MeetupSuggestion('MongoDB', 'A workshop on MongoDB', true, ''),
	      new MeetupSuggestion('RabbitMQ', 'A workshop on RabbitMQ', true, ''),
	      new MeetupSuggestion('Linode', 'A workshop on using a VPS Server on Linode. Linode has a new $5 month server.', true, '')
	    ]
	  }

	  getSuggestions(): MeetupSuggestion[] {
	    return this.suggestions;
	  }
	  add(){
	    this.suggestions.push( new MeetupSuggestion('', '', false, ''));
	  }
	  addVote(index){
	    this.suggestions[index].votes +=1;
	    this.suggestions.sort( (a,b) => {
	      return a.votes < b.votes ? 1 : -1;
	    });
	  }
	}
	```

Update Component	Open code-norman/src/app/meetup-suggestions/meetup-suggestions.ts
	```
	import { Component, OnInit, Input, Output } from '@angular/core';
	import {MeetupSuggestionService} from "../meetup-suggestion.service";
	import {MeetupSuggestion} from "../meetup-suggestion";

	@Component({
	  selector: 'app-meetup-suggestion',
	  templateUrl: './meetup-suggestion.component.html',
	```

```
  styleUrls: ['./meetup-suggestion.component.css'],
  providers: [MeetupSuggestionService]
})
export class MeetupSuggestionComponent implements OnInit
{
  @Input()
  @Output()
  suggested: MeetupSuggestion[];

  constructor(private meetupSuggestionService:
MeetupSuggestionService) {   }

  ngOnInit() {
    this.suggested =
this.meetupSuggestionService.getSuggestions();
  }

  addSuggestion(){
    console.log('called');
    this.meetupSuggestionService.add();
    this.suggested =
this.meetupSuggestionService.getSuggestions();
  }
  addVote(index){
    console.log('index', index);
    this.meetupSuggestionService.addVote(index);
    this.suggested =
this.meetupSuggestionService.getSuggestions();

  }
}
```

Before we update the view add a few more controls from PrimeNG

Open code-norman/src/app/app.module.ts

~line 5

```
import {PanelModule, ButtonModule, InputTextModule,
InputTextareaModule, InputSwitchModule, ToolbarModule}
from 'primeng/primeng';
```

~line 23 (after PanelModule)

```
    ButtonModule, InputTextModule, InputTextareaModule,
    InputSwitchModule, ToolbarModule
```

Update Component View

Open code-norman/src/app/meetup-suggestions/meetup-suggestions.html

```
<button pButton type="button" (click)="addSuggestion()"
label="Add Suggestion"></button>
```

```
<h3>Suggestions</h3>
<p-panel *ngFor="let s of suggested; let i=index">
  <p-header>
    <span>{{s.name}}</span>
    <div style="float: right">
      <span>Votes: {{s.votes}}
        <button pButton type="button"
(click)="addVote(i)" icon="fa fa-arrow-up"></button>
      </span>
    </div>
  </p-header>
  <label>Name</label><input type="text" pInputText
[(ngModel)]="s.name" placeholder="Name"/>
  <br>
  <div>
    <label>Description</label>
    <br>
    <textarea pInputTextarea  rows="5" cols="80"
[(ngModel)]="s.description" placeholder="Enter a
description"></textarea>
  </div>
  <p-inputSwitch [(ngModel)]="s.willPresent"
onLabel="Will Present" offLabel="Won't Present"></p-
inputSwitch>
  <br>
  <label>Email</label><input type="text" pInputText
disabled="{{!s.willPresent}}" [(ngModel)]="s.email"
placeholder="email"/>
</p-panel>
```

Add router and navigation

routerLink is an angular directive that will update the <router-outlet> area.

Edit code-norman/src/app/app.component.html

```
<p-toolbar>
  <div class="ui-toolbar-group-left">
    <button pButton type="button" label="Upcoming
Meetings" routerLink="upcoming-meetups"></button>
    <button pButton type="button" label="Suggest/Vote
for future Meeting" routerLink="suggest-
meetup"></button>
  </div>
</p-toolbar>
<router-outlet></router-outlet>
```

Import Router Outlet and set routes

Open code-norman/app/app.module.ts

after imports(~line 10)

```
const routes:Routes = [
  {path: '', redirectTo:'upcoming-meetups',
pathMatch:"full"},
  {path: 'upcoming-meetups', component:
MeetupsComponent},
  {path: 'suggest-meetup', component:
MeetupSuggestionComponent},
  {path: 'hello-world', component: HelloWorldComponent},
  {path: 'vote', redirectTo: 'suggest-meetup'}
];
```

~line 34 (In imports)

```
  RouterModule.forRoot(routes)
```

Commit work to git

create new branch	`git checkout -b meetup-suggestion`
add new files to git	`git add *`
commit changes	`git commit -m "add meetup-suggestion component"`