ANGULAR 6

Lesson 02

OBJECTIVES

Introduction to Angular Framework

- Introduction to Angular Framework, History & Overview
- Environment Setup
- Angular CLI, Installing Angular CLI
- NPM commands & package.json
- Bootstrapping Angular App, Components, AppModule
- Project Setup, Editor Environments
- First Angular App & Directory Structure
- Angular Fundamentals, Building Blocks
- MetaData

INTRODUCTION TO ANGULAR FRAMEWORK

Angular Framework…

- Angular is an open source JavaScript library that is sponsored and maintained by Google.
- Angular applications are built around a design pattern called *Model-View-Controller* (MVC)

Angular Framework…

- *Extendable:* It is easy to figure out how even a complex Angular app works once you understand the basics—and that means you can easily enhance applications to create new and useful features for your users.
- *Maintainable*: Angular apps are easy to debug and fix, which means that long-term maintenance is simplified.
- *Testable:* Angular has good support for unit and end-to-end testing, meaning that you can find and fix defects before your users do.
- *Standardized:* Angular builds on the innate capabilities of the web browser without getting in your way, allowing you to create standards compliant web apps that take advantage of the latest features (such as HTML5 APIs) and popular tools and frameworks.

Angular-History

- Angular was created in the year 2009 as AngularJS

It is **large** and **fastest-growing** community

It is **open source** and **supported by Google**

The first stable version

- Patch Releases
- Bug Fixes
- Small Features

Angular 1.3

Angular 1.4

Performance Boost

- 30% faster digest times
- New Router
- Internationalization
- Ng-animate, ng-messages

Additional concepts from angular

- Concept of Component based architecture

Angular 1.5

Angular 1.6

- Sandbox support removed
- Uses new router
- Uses ES6 Modules

- Performance
- The changing web
- Mobile
- Ease to use

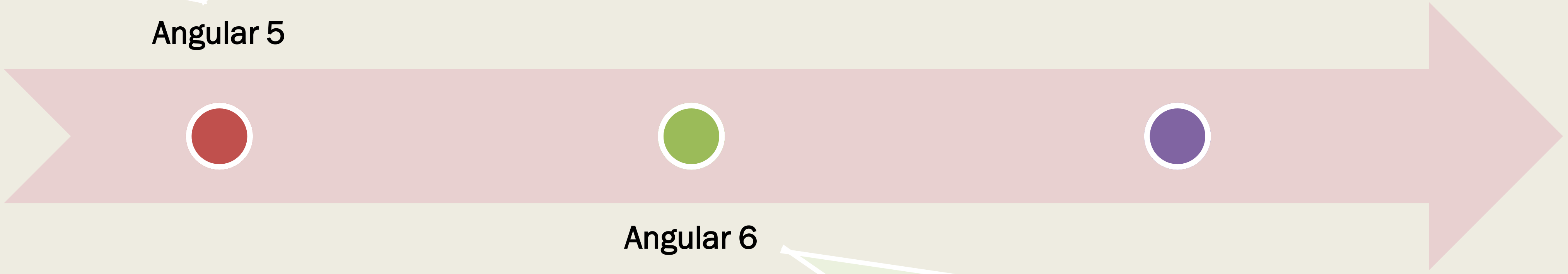
Angular 2.0

Angular 4

- Smaller and Faster
- Animation package
- Angular Universal
- Typescript 2.1 and 2.2 compatibility
- New ParamMap interface

- Simpler Progressive Web Applications
- Material Design
- Build optimizer
- Angular Universal API and DOM
- Improved Compiler and Typescript
- CLI v1.5
- Router Hooks
- Number, date and currency pipes update

Angular 5



Angular 6

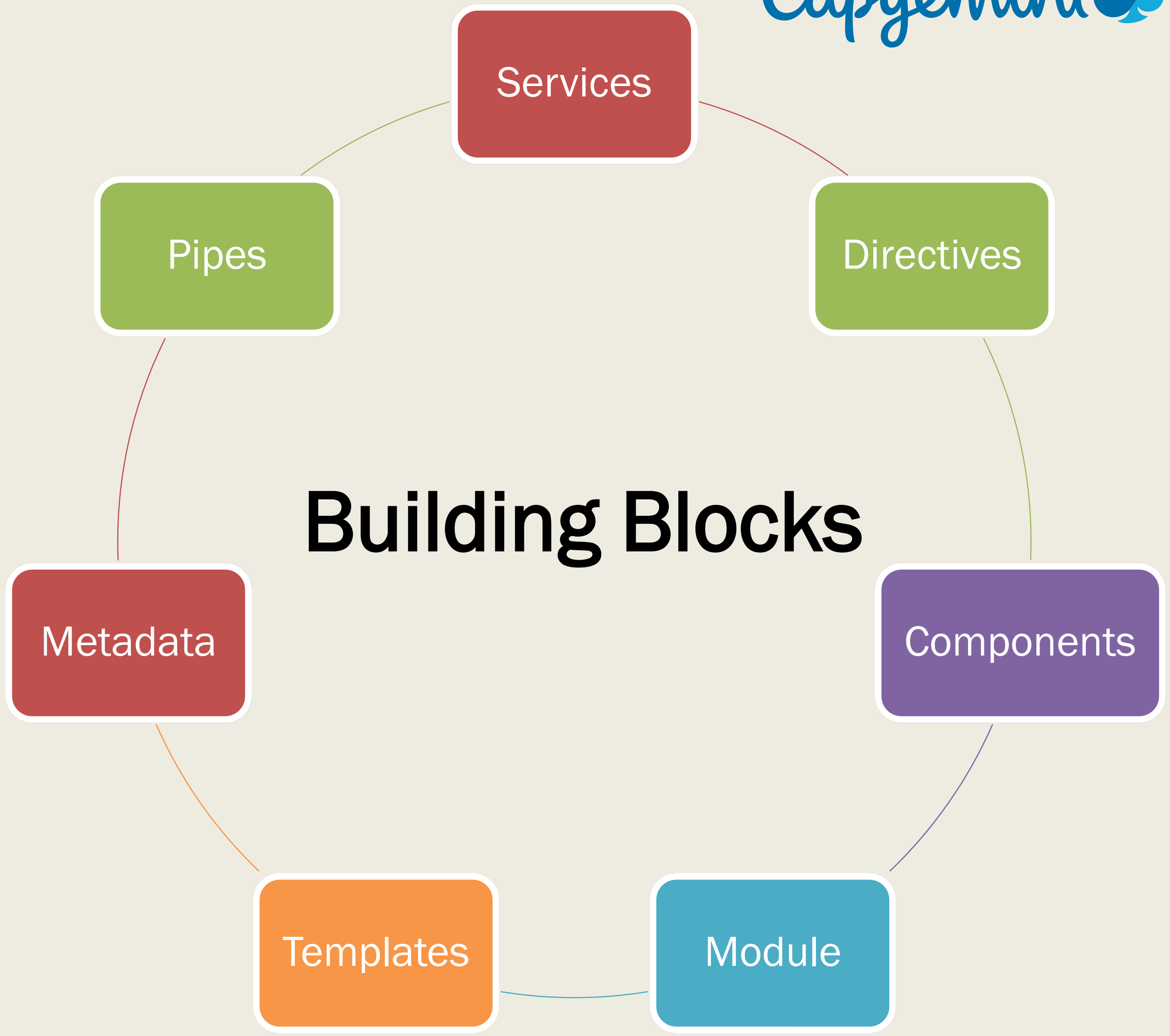
- Improved CLI
- RxJS Revamp
- Library Generation Support
- Ivy – the cutting-edge rendering pipeline
- Angular Elements

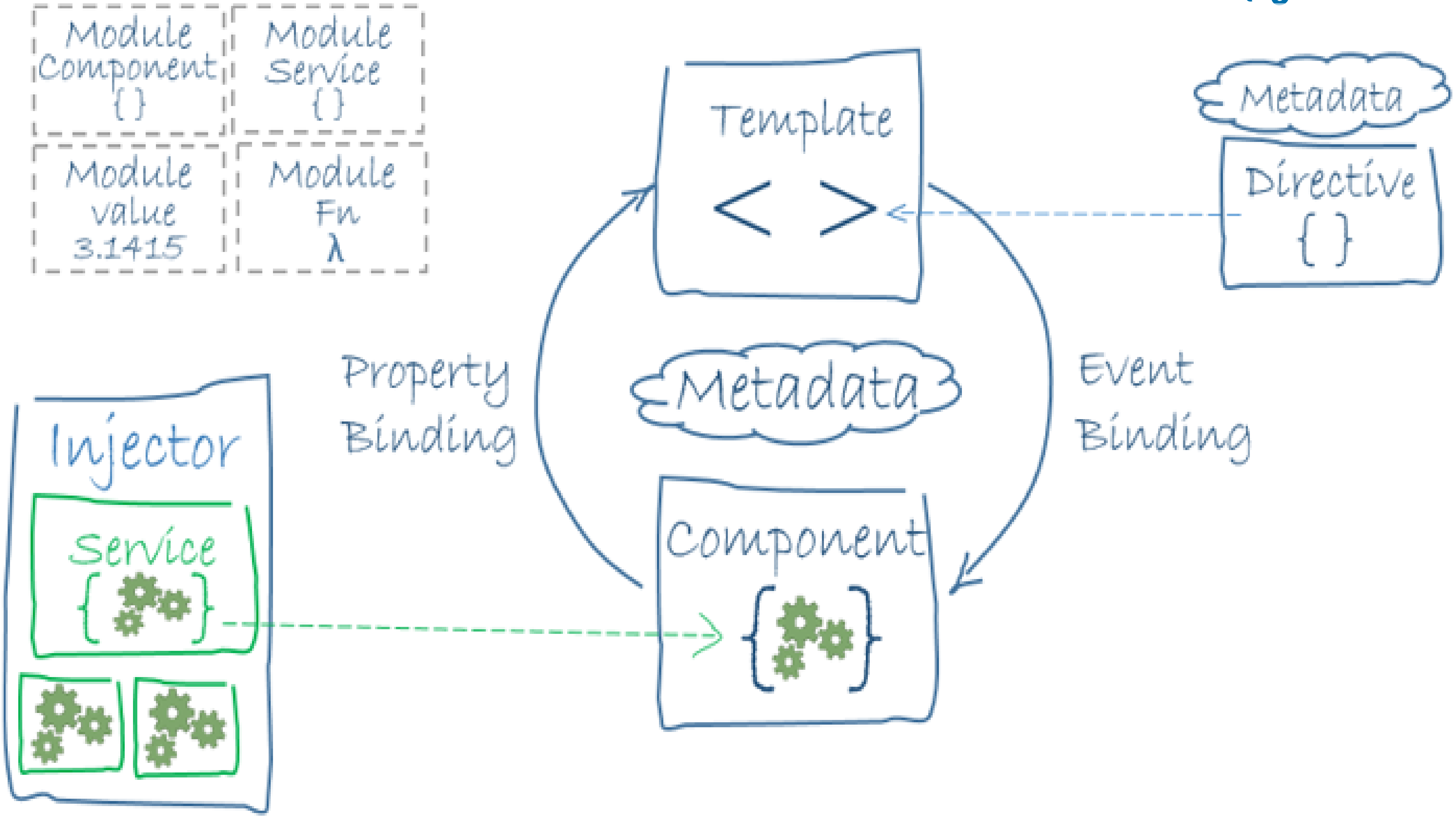
ANGULAR FEATURES

- Cross platform
- Speed and performance
- Productivity
- Full development story

Building Blocks

- Components
- Modules
- Templates
- Metadata
- Pipes
- Services
- Directives





Module Component { }
Module Service { }
Module value 3.1415
Module Fn λ
Template < >
Metadata
Directive { }
Property Binding
Metadata
Event Binding
Injector
Service
Component

Modules

- Angular apps are modular and Angular has its own modularity system called NgModule.
- Every Angular app has at least one NgModule class, the root module, conventionally named AppModule.

Angular libraries

- Angular ships as a collection of JavaScript modules. You can think of them as library modules.
- Each Angular library name begins with the @angular prefix. You install them with the **npm** package manager and import parts of them with JavaScript import statements.

– *Component*	*@angular/core*
– *BrowserModule*	*@angular/platform-browser*
– *FormsModule*	*@angular/forms*

Component

- A component controls a patch of screen called a view and used to manage templates.
- You define a component's application logic—what it does to support the view—inside a class.
- The class interacts with the view through an API of properties and methods.

- TEMPLATE
  - *A template is a form of HTML that tells Angular how to render the component*
- METADATA
  - *Metadata tells Angular how to process a class. We can attach metadata to a class by using a* ***decorator.*** *@Component****,*** *@Injectable, @Input, and @Output are a few of the more popular decorators*

Environment Setup

- **Node.Js and npm**
  - *To get started with Angular, you'll need to have Node.js installed.*
  - *There are a couple of different ways you can install Node.js, so please refer to the Node.js website for detailed information https://nodejs.org/download/*
- **Angular CLI**
  - *CLI – Command Line Interface – This is simply a tool set, which is used for creating, managing and building angular applications quickly.*
  - *It quickly creates new angular projects, and then you can use some commands to build that project for production and so on.*

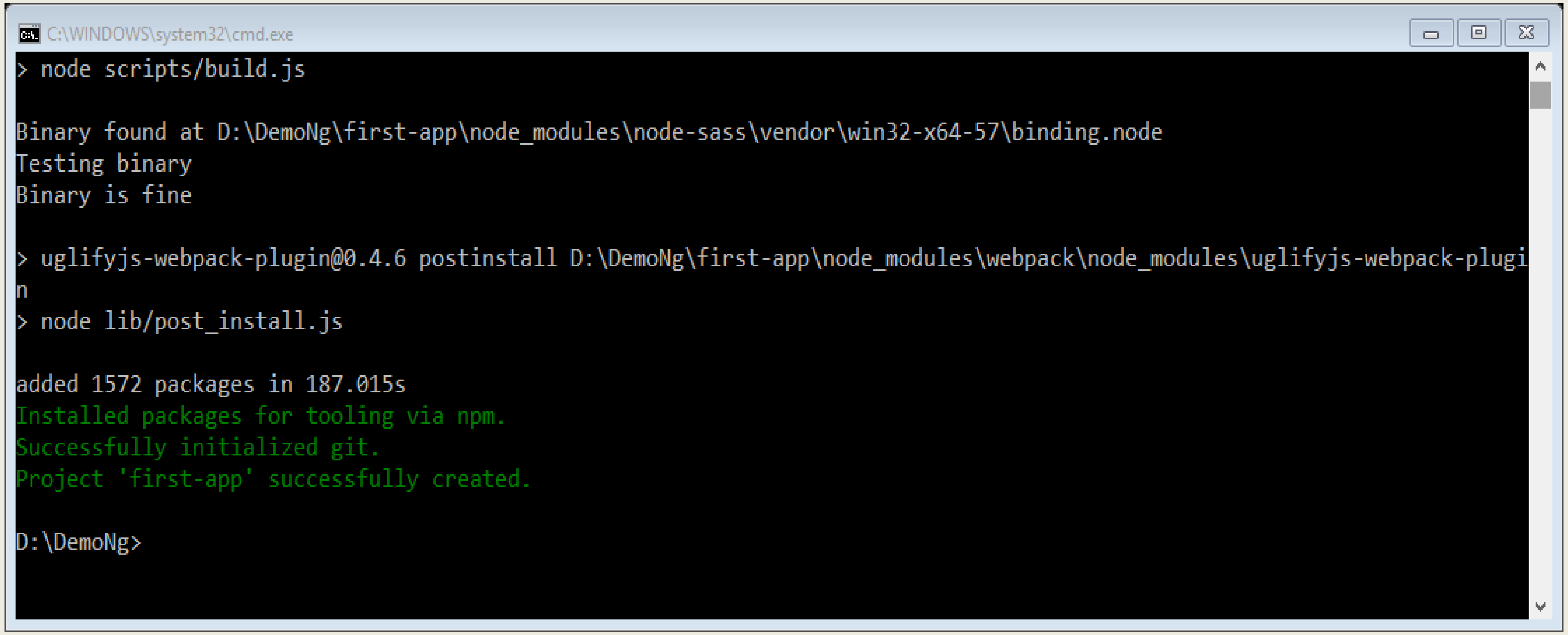
```
> npm install -g @angular/cli
```

- **VS Code**
  - *Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux.*
  - *Download VS Code - Quickly find the appropriate install for your platform (Windows, macOS and Linux).*

CREATE A NEW ANGULAR APPLICATION

- AngularJS is based on the model view controller, whereas Angular 2/4/5/6 is based on the components structure.
- To install Angular framework, the Angular team came up with Angular CLI which eases the installation. You need to run through a few commands to install Angular.
- Go to this site https://cli.angular.io to install Angular CLI.

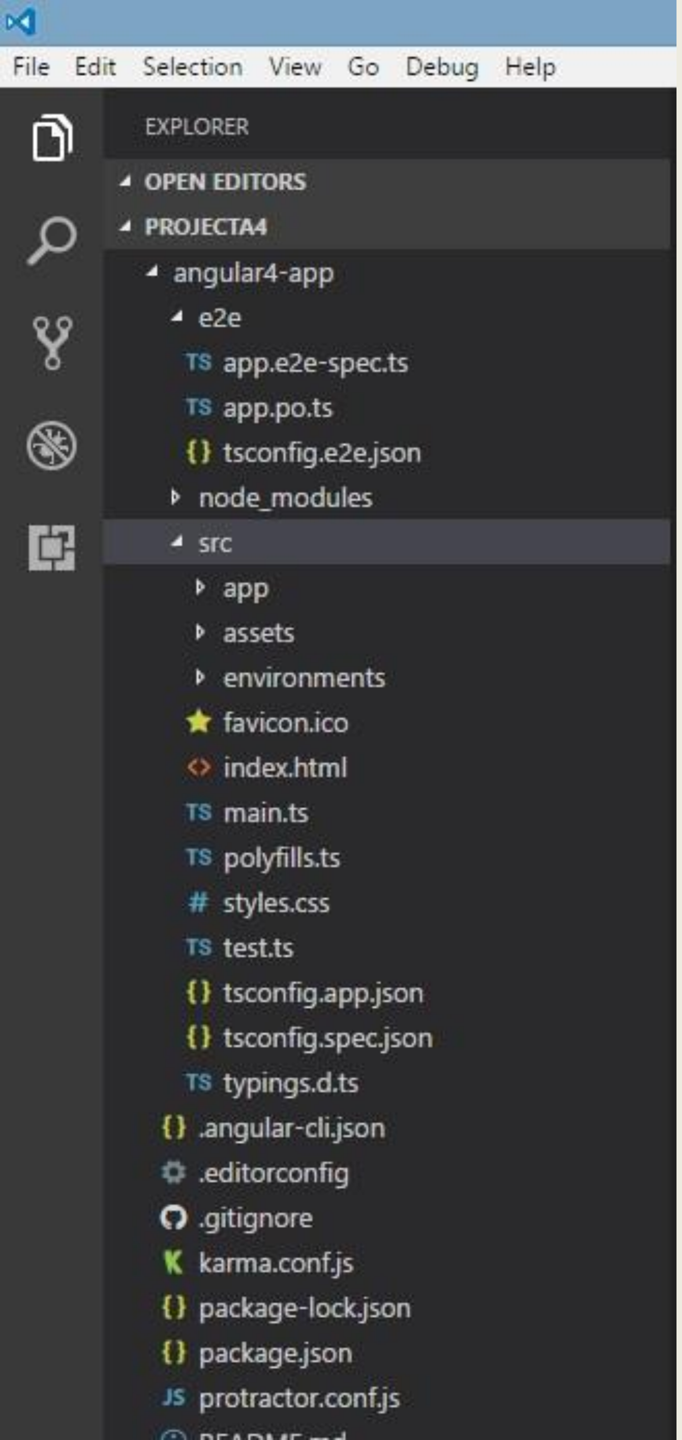
- Create a new project named first-app with this CLI command.
  - *> ng new first-app*



C:\WINDOWS\system32\cmd.exe

```
> node scripts/build.js

Binary found at D:\DemoNg\first-app\node_modules\node-sass\vendor\win32-x64-57\binding.node
Testing binary
Binary is fine

> uglifyjs-webpack-plugin@0.4.6 postinstall D:\DemoNg\first-app\node_modules\webpack\node_modules\uglifyjs-webpack-plugi
n
> node lib/post_install.js

added 1572 packages in 187.015s
Installed packages for tooling via npm.
Successfully initialized git.
Project 'first-app' successfully created.

D:\DemoNg>
```

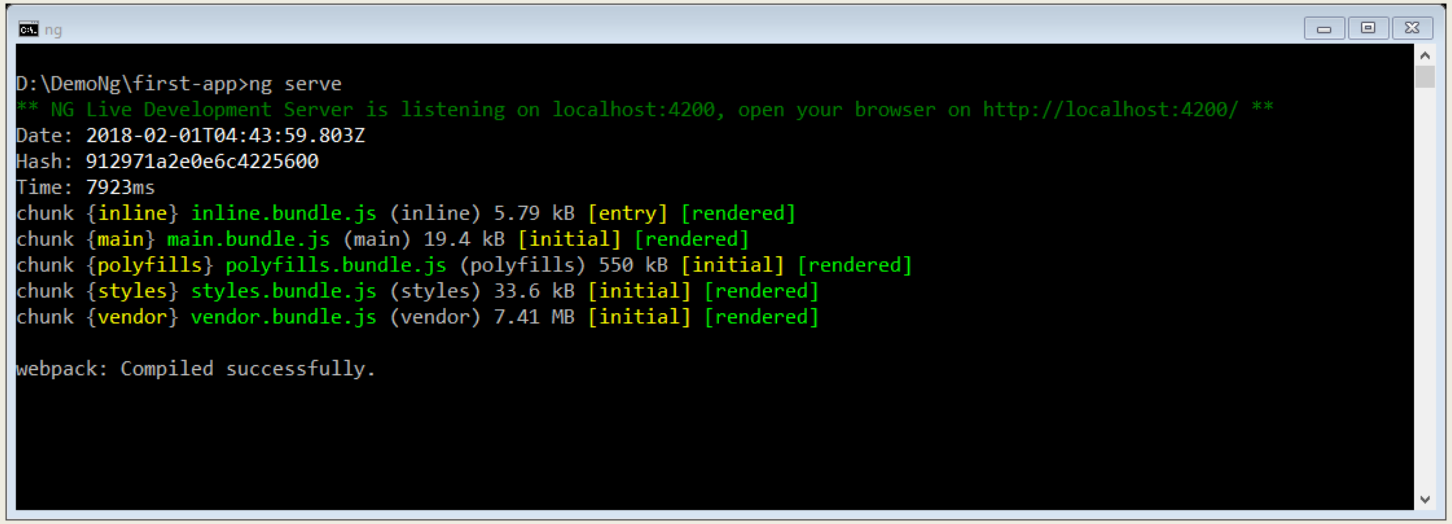
- We will use Visual Studio Code IDE for working with Angular; you can use any IDE, i.e., Atom, WebStorm, etc.
- To download Visual Studio Code, go to https://code.visualstudio.com/ and click Download for Windows.



- Now that we have the file structure for our project, let us compile our project with the following command –

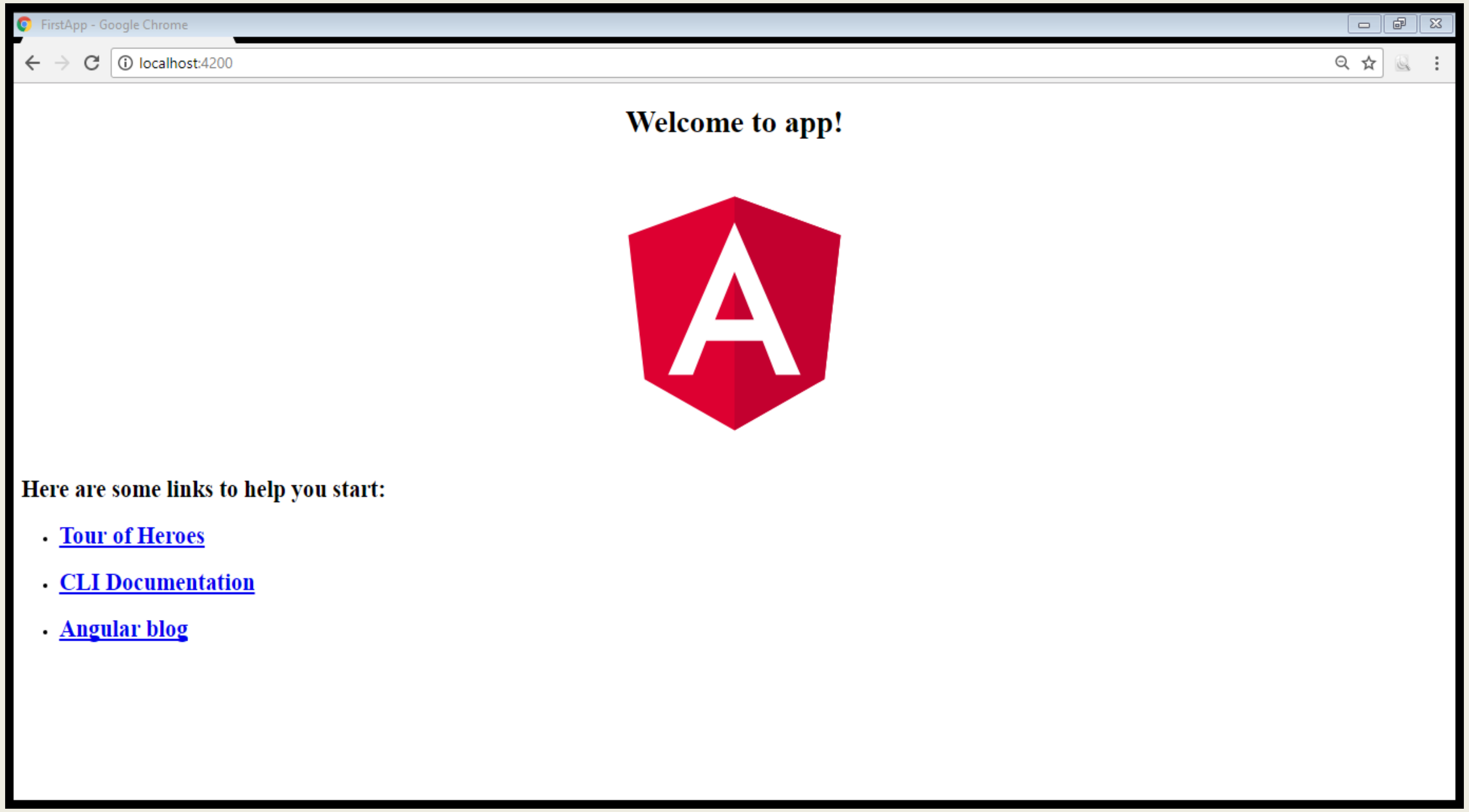
  > *ng serve*

- The ng serve command builds the application and starts the web server.



```
D:\DemoNg\first-app>ng serve
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
Date: 2018-02-01T04:43:59.803Z
Hash: 912971a2e0e6c4225600
Time: 7923ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 19.4 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 550 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 33.6 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.41 MB [initial] [rendered]

webpack: Compiled successfully.
```

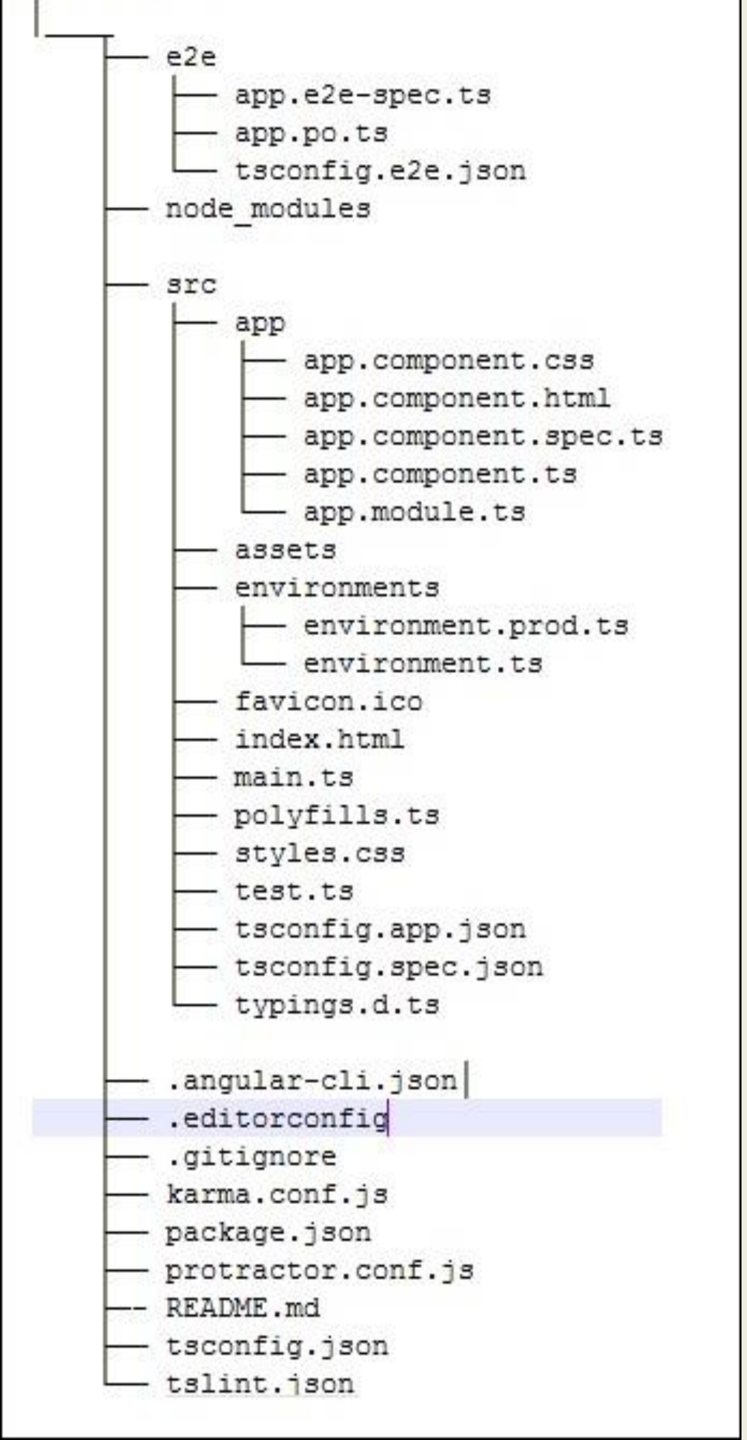
- The web server starts on port 4200. Type the url **http://localhost:4200/** in the browser and see the output. Once the project is compiled, you will receive the following output –
- Once you run **http://localhost:4200/** in the browser, you will be directed to the following screen –



Welcome to app!

Here are some links to help you start:

- Tour of Heroes
- CLI Documentation
- Angular blog



```
|
├── e2e
│   ├── app.e2e-spec.ts
│   ├── app.po.ts
│   └── tsconfig.e2e.json
├── node_modules
│
├── src
│   ├── app
│   │   ├── app.component.css
│   │   ├── app.component.html
│   │   ├── app.component.spec.ts
│   │   ├── app.component.ts
│   │   └── app.module.ts
│   ├── assets
│   ├── environments
│   │   ├── environment.prod.ts
│   │   └── environment.ts
│   ├── favicon.ico
│   ├── index.html
│   ├── main.ts
│   ├── polyfills.ts
│   ├── styles.css
│   ├── test.ts
│   ├── tsconfig.app.json
│   ├── tsconfig.spec.json
│   └── typings.d.ts
│
├── .angular-cli.json
├── .editorconfig
├── .gitignore
├── karma.conf.js
├── package.json
├── protractor.conf.js
├── README.md
├── tsconfig.json
└── tslint.json
```

Project folder structure

- The Angular app folder has the following **folder structure** –
  - **e2e** – *end to end test folder. Mainly e2e is used for integration testing and helps ensure the application works fine.*
  - ***node_modules*** – *The npm package installed is node_modules. You can open the folder and see the packages available.*
  - ***src*** – *This folder is where we will work on the project using Angular.*
    - **app** - The Angular **app** folder has the following **file structure** –
  - ***.angular-cli.json*** – *It basically holds the project name, version of cli, etc.*
  - ***.editorconfig*** – *This is the config file for the editor.*
  - ***.gitignore*** – *A .gitignore file should be committed into the repository, in order to share the ignore rules with any other users that clone the repository.*
  - ***karma.conf.js*** – *This is used for unit testing via the protractor. All the information required for the project is provided in karma.conf.js file.*
  - ***package.json*** – *The package.json file tells which libraries will be installed into node_modules when you run npm install.*
  - ***protractor.conf.js*** – *This is the testing configuration required for the application.*
  - ***tsconfig.json*** – *This basically contains the compiler options required during compilation.*
  - ***tslint.json*** – *This is the config file with rules to be considered while compiling.*

App

- The App directory contains the files described below. These files are installed by angular-cli by default.
- **app.module.ts** – If you open the file, you will see that the code has reference to different libraries, which are imported. Angular-cli has used these default libraries for the import – angular/core, platform-browser. The names itself explain the usage of the libraries.
- They are imported and saved into variables such as **declarations, imports, providers**, and **bootstrap**.

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { AppComponent } from './app.component';
 @NgModule({
   declarations: [
      AppComponent
   ],
   imports: [
      BrowserModule
   ],
   providers: [],
   bootstrap: [AppComponent]
})
 export class AppModule { }
```

AppModule

- **declarations** – In declarations, the reference to the components is stored. The Appcomponent is the default component that is created whenever a new project is initiated. We will learn about creating new components in a different section.
- **imports** – This will have the modules imported as shown above. At present, BrowserModule is part of the imports which is imported from @angular/platform-browser.
- **providers** – This will have reference to the services created. The service will be discussed in a subsequent chapter.
- **bootstrap** – This has reference to the default component created, i.e., AppComponent.

- **app.component.css** – You can write your css structure over here. Right now, we have added the background color to the div as shown below.
- **app.component.html** – The html code will be available in this file.
- **app.component.spec.ts** – These are automatically generated files which contain unit tests for source component.
- **app.component.ts** – The class for the component is defined over here. You can do the processing of the html structure in the .ts file. The processing will include activities such as connecting to the database, interacting with other components, routing, services, etc.

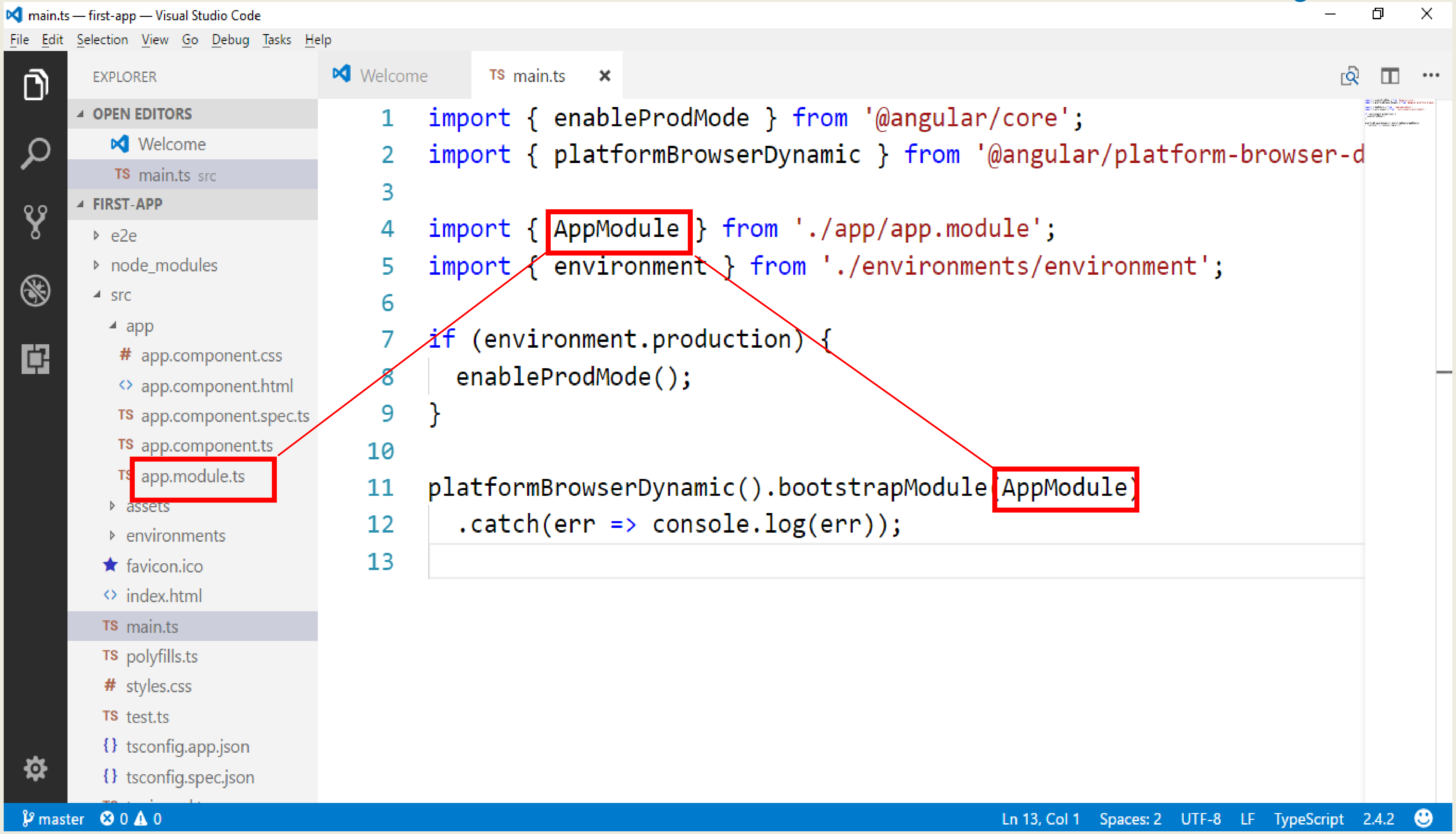
HOW ANGULAR WORKS

How Angular Works

- The first big idea is that an Angular application is made up of Components.
- One way to think of Components is a way to teach the browser new tags.
- If you have an Angular 1 background, Components are analogous to directives in AngularJS 1.x

How Angular Works

- Every app has a main entry point. This application was built using Angular CLI (which is built on a tool called Webpack). We run this app by calling the command:
- > ng serve
  - *ng will look at the file .angular-cli.json to find the entry point to our app*
  - *.angular-cli.json specifies a "main" file, which in this case is main.ts*
  - *main.ts is the entry-point for our app and it bootstraps our application*
  - *The bootstrap process boots an Angular module ("AppModule")*
  - *We use the AppModule to bootstrap the app. AppModule is specified in src/app/app.module.ts*
  - *AppModule specifies which component to use as the top-level component. In this case it is AppComponent*
  - *AppComponent has <app-root> tags in the template and this renders output*



```typescript
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic } from '@angular/platform-browser-d

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.log(err));
```