**What is Angular?**

Angular is a framework for building client applications in HTML, CSS, and either JavaScript or a language like TypeScript that can be compiled (more accurately, transpiled) to JavaScript

* It gives our applications a proper, maintainable and testable structure.
* At the core of every Angular application, we have a tree of components.
* Each component encapsulates the state (data), the HTML markup and the behavior for a view.
* At a minimum, a component is implemented using a class. This class may include a bunch of fields for displaying data and methods, which will be called in response to events raised from the DOM events.
* A component may have an external template stored in a separate HTML file.
* It may also include one or more external stylesheets.
* A component also has a selector. That tells Angular where in the DOM to place this component.
* We use the property binding syntax (square brackets) to bind properties of DOM elements to fields/properties in our components.
* When the value of these fields/properties change, Angular automatically updates the DOM (view).
* We use the event binding syntax (parenthesis) to bind events of DOM elements to methods in our component.
* Angular automatically calls these methods to handle events raised from the DOM elements.

### What is TypeScript?

TypeScript is a superset of JavaScript. That means any valid JavaScript code is valid TypeScript code. But many prefer TypeScript because it has additional features that we don’t have in the current version of JavaScript that most browsers understand. So, when building Angular applications, we need to have our TypeScript code converted into JavaScript code that browsers can understand. This process is called transpilation which is the combination of translate and compile. And that’s the job of  the TypeScript compiler.

**Why do we need angular Framework?**

nothing inherently wrong with using vanilla JavaScript/jQuery. In fact, a lot of web applications out there are built this way. But as your application grows, structuring your code in a clean and maintainable and more importantly, testable way, becomes harder and harder. I’m not saying this is impossible. But using a framework like Angular, makes your life far easier.

Sure, there is a learning curve involved, but once you master Angular, you’ll be able to build client applications faster and easier.

### Is Angular better than React/Vue.js?

These days there is a lot of debate between Angular vs React vs Vue.js. Whether Angular is better than React or Vue.js depends on how you define “better”. Each of these frameworks have strengths and weaknesses

## Your First Angular App

So with Angular CLI in place, now we can create a new Angular project using the following command:

ng new hello-world

As you can see, we can access Angular CLI using **ng**. We provide a command (in this case **new**) to instruct Angular CLI on what we want it to do for us. Here, we are telling Angular CLI to generate a new project called “hello-world” and store it in a folder with the same name. We’ll be looking at these generated files shortly. For now, let’s finish the first step and see our new Angular app in the browser.

So, run the following commands in the terminal:

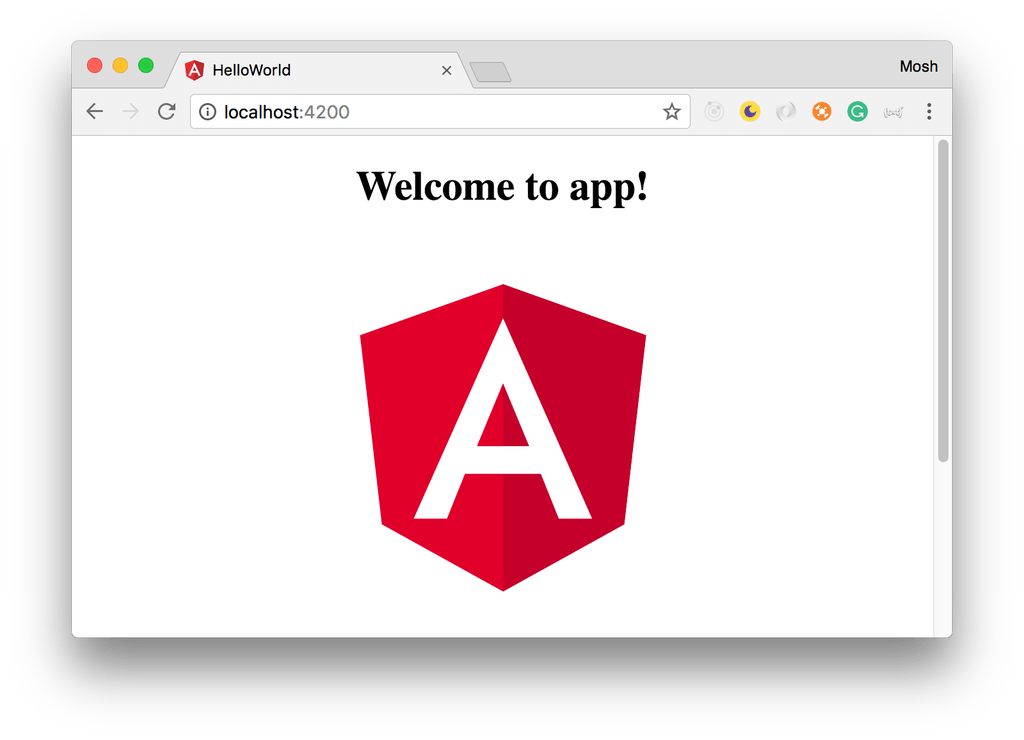
cd hello-world

npm install

ng serve

With the second command (**npm install**) we install all the dependencies of our application. The last command (**ng serve**)compiles our application and hosts it using a lightweight web server. Now, we can access our application at **http://localhost:4200.**

So, open up your browser and navigate to this address. You should see the following page:



Congratulations! You generated and served your first Angular 4 application. Now, let’s have a quick look at the files and folders in our new Angular project.

## Structure of Angular Projects

Inside the generated folder, you’ll find the following top-level folders:

* **e2e**: includes end-to-end tests.
* **node\_modules**: all the third-party libraries that our project is dependent upon.
* **src**: the actual source code of our Angular application.

99.9% of the time you’ll be working with the files inside the **src**folder. But let’s quickly overview the other files we have in this project:

**angular-cli.json:**a configuration file for Angular CLI. We use this file to import third-party stylesheets or define additional environments (eg testing environment) for our application.

**package.json:**a standard file for Node-based projects. It contains metadata about our project, such as its name, version as well as the list of its dependencies.

**protractor.conf.js:**Protractor is a tool for running end-to-end tests for Angular projects. We hardly need to modify this file.

**karma.conf.js:**Karma is a test runner for JavaScript applications. This file contains some configuration for Karma. We rarely need to modify this file.

**tsconfig.json:**includes setting for the TypeScript compiler. Again, we hardly, if ever, need to modify this file.

**tslint.json:**includes the settings for TSLint which is a popular tool for linting TypeScript code. That means it checks the quality of our TypeScript code based on a few configurable parameters. This is especially important in a team environment to ensure that everyone follows the same conventions and produces code of the same quality. We can run linting on our code using Angular CLI in the terminal or we can add it to our editor as a plug-in.