## Step 1. Set up the Development Environment

You need to set up your development environment before you can do anything.

Install **Node.js® and npm** if they are not already on your machine.

**Verify that you are running at least node**6.9.x**and npm**3.x.x by running node -v and npm -v in a terminal/console window. Older versions produce errors, but newer versions are fine.

Then **install the Angular CLI** globally.

npm install -g @angular/cli

## Step 2. Create a new project

Open a terminal window.

Generate a new project and skeleton application by running the following commands:

ng new my-app

Patience, please. It takes time to set up a new project; most of it is spent installing npm packages.

## Step 3: Serve the application

Go to the project directory and launch the server.

cd my-app

ng serve --open

The ng serve command launches the server, watches your files, and rebuilds the app as you make changes to those files.

Using the --open (or just -o) option will automatically open your browser on http://localhost:4200/.

Your app greets you with a message:



## Step 4: Edit your first Angular component

The CLI created the first Angular component for you. This is the root component and it is named app-root. You can find it in ./src/app/app.component.ts.

Open the component file and change the title property from 'app' to 'My First Angular App!'.

src/app/app.component.ts

export class AppComponent {

title = 'My First Angular App!';

}

The browser reloads automatically with the revised title. That's nice, but it could look better.

Open src/app/app.component.css and give the component some style.

src/app/app.component.css

h1 {

color: #369;

font-family: Arial, Helvetica, sans-serif;

font-size: 250%;

}



Looking good!

## What's next?

That's about all you'd expect to do in a "Hello, World" app.

You're ready to take the Tour of Heroes Tutorial and build a small application that demonstrates the great things you can build with Angular.

Or you can stick around a bit longer to learn about the files in your brand new project.

## Project file review

An Angular CLI project is the foundation for both quick experiments and enterprise solutions.

The first file you should check out is README.md. It has some basic information on how to use CLI commands. Whenever you want to know more about how Angular CLI works make sure to visit the Angular CLI repository and Wiki.

Some of the generated files might be unfamiliar to you.

### The src folder

Your app lives in the src folder. All Angular components, templates, styles, images, and anything else your app needs go here. Any files outside of this folder are meant to support building your app.

src

app

app.component.css

app.component.html

app.component.spec.ts

app.component.ts

app.module.ts

assets

.gitkeep

environments

environment.prod.ts

environment.ts

favicon.ico

index.html

main.ts

polyfills.ts

styles.css

test.ts

tsconfig.app.json

tsconfig.spec.json

|  |  |
| --- | --- |
| **File** | **Purpose** |
| app/app.component.{ts,html,css,spec.ts} | Defines the AppComponent along with an HTML template, CSS stylesheet, and a unit test. It is the **root**component of what will become a tree of nested components as the application evolves. |
| app/app.module.ts | Defines AppModule, the root module that tells Angular how to assemble the application. Right now it declares only the AppComponent. Soon there will be more components to declare. |
| assets/\* | A folder where you can put images and anything else to be copied wholesale when you build your application. |
| environments/\* | This folder contains one file for each of your destination environments, each exporting simple configuration variables to use in your application. The files are replaced on-the-fly when you build your app. You might use a different API endpoint for development than you do for production or maybe different analytics tokens. You might even use some mock services. Either way, the CLI has you covered. |
| favicon.ico | Every site wants to look good on the bookmark bar. Get started with your very own Angular icon. |
| index.html | The main HTML page that is served when someone visits your site. Most of the time you'll never need to edit it. The CLI automatically adds all js and css files when building your app so you never need to add any <script> or <link> tags here manually. |
| main.ts | The main entry point for your app. Compiles the application with the JIT compiler and bootstraps the application's root module (AppModule) to run in the browser. You can also use the AOT compilerwithout changing any code by appending the--aot flag to the ng build and ng serve commands. |
| polyfills.ts | Different browsers have different levels of support of the web standards. Polyfills help normalize those differences. You should be pretty safe with core-js and zone.js, but be sure to check out the Browser Support guide for more information. |
| styles.css | Your global styles go here. Most of the time you'll want to have local styles in your components for easier maintenance, but styles that affect all of your app need to be in a central place. |
| test.ts | This is the main entry point for your unit tests. It has some custom configuration that might be unfamiliar, but it's not something you'll need to edit. |
| tsconfig.{app|spec}.json | TypeScript compiler configuration for the Angular app (tsconfig.app.json) and for the unit tests (tsconfig.spec.json). |

### The root folder

The src/ folder is just one of the items inside the project's root folder. Other files help you build, test, maintain, document, and deploy the app. These files go in the root folder next to src/.

my-app

e2e

app.e2e-spec.ts

app.po.ts

tsconfig.e2e.json

node\_modules/...

src/...

.angular-cli.json

.editorconfig

.gitignore

karma.conf.js

package.json

protractor.conf.js

README.md

tsconfig.json

tslint.json

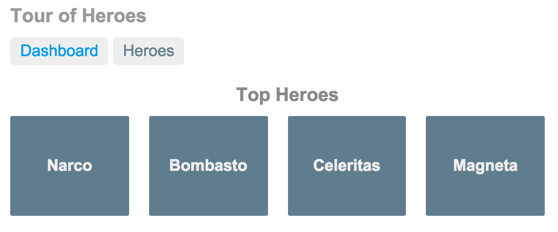
|  |  |
| --- | --- |
| **File** | **Purpose** |
| e2e/ | Inside e2e/ live the end-to-end tests. They shouldn't be inside src/ because e2e tests are really a separate app that just so happens to test your main app. That's also why they have their own tsconfig.e2e.json. |
| node\_modules/ | Node.js creates this folder and puts all third party modules listed in package.json inside of it. |
| .angular-cli.json | Configuration for Angular CLI. In this file you can set several defaults and also configure what files are included when your project is built. Check out the official documentation if you want to know more. |
| .editorconfig | Simple configuration for your editor to make sure everyone that uses your project has the same basic configuration. Most editors support an .editorconfig file. See http://editorconfig.org for more information. |
| .gitignore | Git configuration to make sure autogenerated files are not commited to source control. |
| karma.conf.js | Unit test configuration for the Karma test runner, used when running ng test. |
| package.json | npm configuration listing the third party packages your project uses. You can also add your own custom scripts here. |
| protractor.conf.js | End-to-end test configuration for Protractor, used when running ng e2e. |
| README.md | Basic documentation for your project, pre-filled with CLI command information. Make sure to enhance it with project documentation so that anyone checking out the repo can build your app! |
| tsconfig.json | TypeScript compiler configuration for your IDE to pick up and give you helpful tooling. |
| tslint.json | Linting configuration for TSLint together with Codelyzer, used when running ng lint. Linting helps keep your code style consistent. |

By the end of the tutorial you will be able to do the following:

* Use built-in Angular directives to show and hide elements and display lists of hero data.
* Create Angular components to display hero details and show an array of heroes.
* Use one-way data binding for read-only data.
* Add editable fields to update a model with two-way data binding.
* Bind component methods to user events, like keystrokes and clicks.
* Enable users to select a hero from a master list and edit that hero in the details view.
* Format data with pipes.
* Create a shared service to assemble the heroes.
* Use routing to navigate among different views and their components.

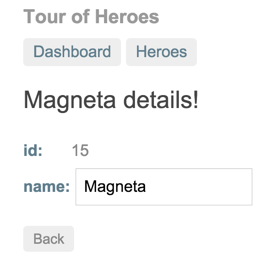
## What you'll build

Here's a visual idea of where this tutorial leads, beginning with the "Dashboard" view and the most heroic heroes:



You can click the two links above the dashboard ("Dashboard" and "Heroes") to navigate between this Dashboard view and a Heroes view.

If you click the dashboard hero "Magneta," the router opens a "Hero Details" view where you can change the hero's name.



Clicking the "Back" button returns you to the Dashboard. Links at the top take you to either of the main views. If you click "Heroes," the app displays the "Heroes" master list view.



When you click a different hero name, the read-only mini detail beneath the list reflects the new choice.

You can click the "View Details" button to drill into the editable details of the selected hero.

The following diagram captures all of the navigation options.



Here's the app in action:

