Angular CLI

With all of the new features Angular 2 takes advantage of, like static typing, decorators and ES6 module resolution, comes the added cost of setup and maintenance. Spending a lot of time with different build setups and configuring all of the different tools used to serve a modern JavaScript application can really take a lot of time and drain productivity by not being able to actually work on the app itself.

Seeing the popularity of ember-cli, Angular 2 decided they would provide their own CLI to solve this problem. Angular CLI is geared to be the tool used to create and manage your Angular 2 app. It provides the ability to:

- create a project from scratch
- scaffold components, directives, services, etc.
- lint your code
- serve the application
- run your unit tests and end to end tests.

> The Angular 2 CLI currently only generates scaffolding in TypeScript, with other dialects to come later.

Setup

Prerequisites

Angular CLI is currently only distributed through npm and requires Node version 4 or greater.

Installation

The Angular 2 CLI can be installed with the following command:

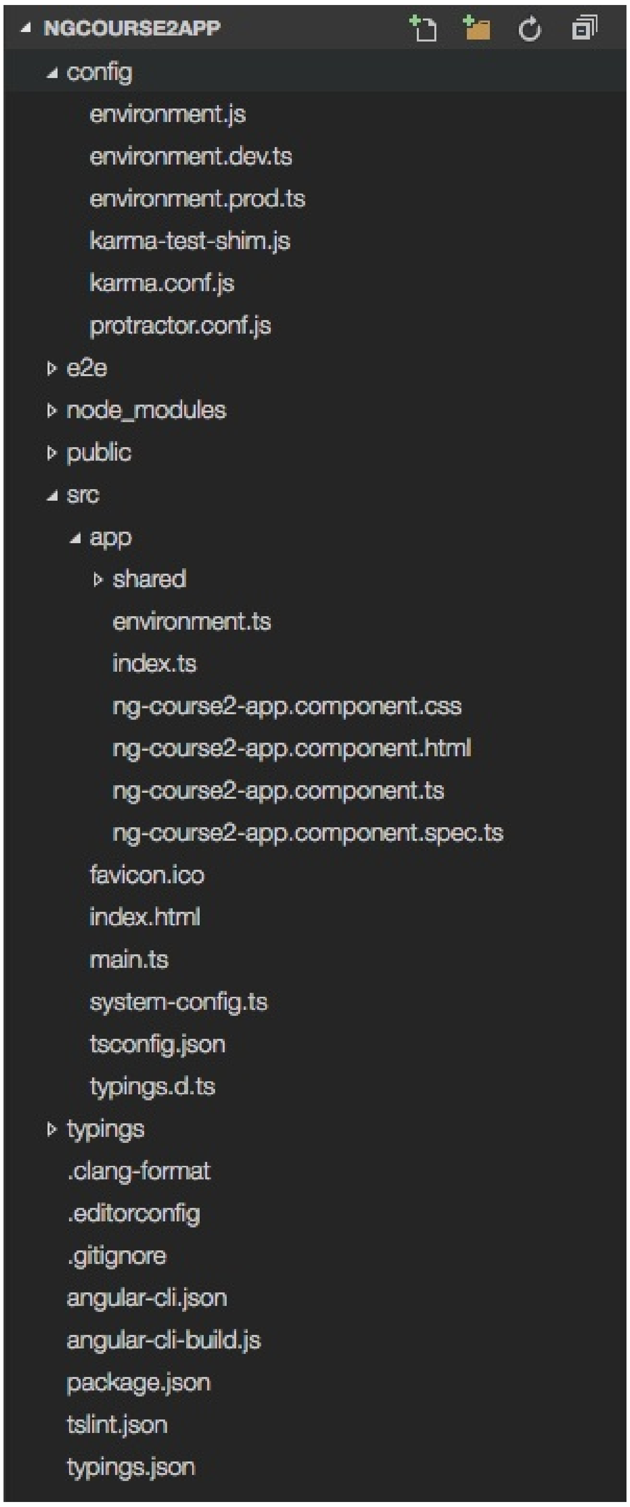
```
npm install -g angular-cli
```

Creating a New App

Use the `ng new [app-name]` command to create a new app. This will generate a basic app in the folder of the app name provided. The app has all of the features available to work with the CLI commands. Creating an app may take a few minutes to complete since npm will need to install all of the dependencies. The directory is automatically set up as a new **git** repository as well. If git is not your version control of choice, simply remove the *.git* folder and *.gitignore* file.

File and Folder Setup

The generated app folder will look like this:



```
NGCOURSE2APP
  config
    environment.js
    environment.dev.ts
    environment.prod.ts
    karma-test-shim.js
    karma.conf.js
    protractor.conf.js
  e2e
  node_modules
  public
  src
    app
      shared
      environment.ts
      index.ts
      ng-course2-app.component.css
      ng-course2-app.component.html
      ng-course2-app.component.ts
      ng-course2-app.component.spec.ts
    favicon.ico
    index.html
    main.ts
    system-config.ts
    tsconfig.json
    typings.d.ts
  typings
  .clang-format
  .editorconfig
  .gitignore
  angular-cli.json
  angular-cli-build.js
  package.json
  tslint.json
  typings.json
```

Figure: App folder

Application configuration is stored in different places, some located in the *config* folder, such as test configuration, and some being stored in the project root such as linting information and build information. The CLI stores application-specific files in the *src* folder and Angular 2-specific code in the *src/app* folder. Files and folders generated by the CLI will follow the official style guide.

> Warning: The CLI relies on some of the settings defined in the configuration files to be able to execute the commands. Take care when modifying them, particularly the package.json file.

The CLI has installed everything a basic Angular 2 application needs to run properly. To make sure everything has run and installed correctly we can run the server.

Serving the App

The CLI provides the ability to serve the app with live reload. To serve an application, simply run the command `ng serve` . This will compile the app and copy all of the application-specific files to the *dist* folder before serving.

By default, `ng serve` serves the application locally on port 4200 (http://localhost:4200) but this can be changed by using a command line argument: `ng serve --port=8080` .

Creating Components

The CLI can scaffold Angular 2 components through the `generate` command. To create a new component run:

```
ng generate component [component-name]
```

Executing the command creates a folder, [component-name], in the project's `src/app` path or the current path the command is executed in if it's a child folder of the project. The folder has the following:

- `[component-name].component.ts` the component class file
- `[component-name].component.css` for styling the component
- `[component-name].component.html` component html
- `[component-name].component.spec.ts` tests for the component
- `index.ts` which exports the component

Creating Routes

The `ng g route [route-name]` command will spin up a new folder and route files for you.

At the time of writing this feature was temporarily disabled due to ongoing changes happening with Angular 2 routing.

Creating Other Things

The CLI can scaffold other Angular 2 entities such as services, pipes and directives using the generate command.

```
ng generate [entity] [entity-name]
```

This creates the entity at `src/app/[entity-name].[entity].ts` along with a spec file, or at the current path if the command is executed in a child folder of the project. The CLI provides blueprints for the following entities out of the box:

Item	Command	Files generated
Component:	`ng g component [name]`	component, HTML, CSS, test spec files
Directive:	`ng g directive [name]`	component, test spec files
Pipe:	`ng g pipe [name]`	component, test spec files
Service:	`ng g service [name]`	component, test spec files
Class:	`ng g class [name]`	component, test spec files
Route:	`ng g route [name]`	component, HTML, CSS, test spec files (in new folder)

Testing

Apps generated by the CLI integrate automated tests. The CLI does this by using the Karma test runner.

Unit Tests

To execute unit tests, run `ng test`. This will run all the tests that are matched by the Karma configuration file at `config/karma.conf.js`. It's set to match all TypeScript files that end in `.spec.ts` by default.

End-to-End Tests

End-to-end tests can be executed by running `ng e2e`. Before end-to-end tests can be performed, the application must be served at some address. Angular CLI uses protractor. It will attempt to access `localhost:4200` by default; if another port is being used, you will have to update the configuration settings located at `config/protractor.conf.js`.

Linting

To encourage coding best practices Angular CLI provides built-in linting. By default the app will look at the project's `tslint.json` for configuration. Linting can be executed by running the command `ng lint`.

For a reference of tslint rules have a look at: https://palantir.github.io/tslint/rules/.

CLI Command Overview

One of the advantages of using the Angular CLI is that it automatically configures a number of useful tools that you can use right away. To get more details on the options for each task, use `ng --help`.

Linting

`ng lint` lints the code in your project using tslint. You can customize the rules for your project by editing `tslint.json`.

> You can switch some of these to use your preferred tool by editing the scripts in `package.json`.

Testing

`ng test` triggers a build and then runs the unit tests set up for your app using Karma. Use the `--watch` option to rebuild and retest the app automatically whenever source files change.

Build

`ng build` will build your app (and minify your code) and place it into the default output path, `dist/`.

Serve

`ng serve` builds and serves your app on a local server and will automatically rebuild on file changes. By default, your app will be served on http://localhost:4200/.

> Include `--port [number]` to serve your app on a different HTTP port.

E2E

Once your app is served, you can run end-to-end tests using `ng e2e`. The CLI uses Protractor for these tests.

Deploy

`ng deploy` deploys to GitHub pages or Firebase.

Adding Third Party Libraries

The CLI generates development automation code which has the ability to integrate third party libraries into the application. Packages are installed using `npm` and the development environment is setup to check the installed libraries mentioned in `package.json` and bundle these third party libraries within the application. For more information see https://github.com/angular/angular-cli#3rd-party-library-installation

Integrating an Existing App

Apps that were created without CLI can be integrated to use CLI later on. This is done by going to the existing app's folder and running `ng init`.

Since the folder structure for an existing app might not follow the same format as one created by the CLI, the `init` command has some configuration options.

- `--source-dir` identifies the relative path to the source files (default = src)
- `--prefix` identifies the path within the source dir that Angular 2 application files reside (default = app)
- `--style` identifies the path where additional style files are located (default = css).