## Angular-CLI

[![Join the chat at https://gitter.im/angular/angular-cli](https://badges.gitter.im/angular/angular-cli.svg)](https://gitter.im/angular/angular-cli?utm\_source=badge&utm\_medium=badge&utm\_campaign=pr-badge&utm\_content=badge)

[![Build Status][travis-badge]][travis-badge-url]

[![Dependency Status][david-badge]][david-badge-url]

[![devDependency Status][david-dev-badge]][david-dev-badge-url]

[![npm][npm-badge]][npm-badge-url]

Prototype of a CLI for Angular 2 applications based on the [ember-cli](http://www.ember-cli.com/) project.

## Note

This project is very much still a work in progress.

The CLI is now in beta.

If you wish to collaborate while the project is still young, check out [our issue list](https://github.com/angular/angular-cli/issues).

Before submitting new issues, have a look at [issues marked with the `type: faq` label](https://github.com/angular/angular-cli/issues?utf8=%E2%9C%93&q=is%3Aissue%20label%3A%22type%3A%20faq%22%20).

## Webpack update

We changed the build system between beta.10 and beta.14, from SystemJS to Webpack.

And with it comes a lot of benefits.

To take advantage of these, your app built with the old beta will need to migrate.

You can update your `beta.10` projects to `beta.14` by following [these instructions](https://github.com/angular/angular-cli/wiki/Upgrading-from-Beta.10-to-Beta.14).

## Prerequisites

Both the CLI and generated project have dependencies that require Node 4 or higher, together

with NPM 3 or higher.

## Table of Contents

\* [Installation](#installation)

\* [Usage](#usage)

\* [Generating a New Project](#generating-and-serving-an-angular2-project-via-a-development-server)

\* [Generating Components, Directives, Pipes and Services](#generating-components-directives-pipes-and-services)

\* [Generating a Route](#generating-a-route)

\* [Creating a Build](#creating-a-build)

\* [Build Targets and Environment Files](#build-targets-and-environment-files)

\* [Base tag handling in index.html](#base-tag-handling-in-indexhtml)

\* [Bundling](#bundling)

\* [Running Unit Tests](#running-unit-tests)

\* [Running End-to-End Tests](#running-end-to-end-tests)

\* [Proxy To Backend](#proxy-to-backend)

\* [Deploying the App via GitHub Pages](#deploying-the-app-via-github-pages)

\* [Linting and formatting code](#linting-and-formatting-code)

\* [Support for offline applications](#support-for-offline-applications)

\* [Commands autocompletion](#commands-autocompletion)

\* [Project assets](#project-assets)

\* [Global styles](#global-styles)

\* [CSS preprocessor integration](#css-preprocessor-integration)

\* [3rd Party Library Installation](#3rd-party-library-installation)

\* [Global Library Installation](#global-library-installation)

\* [Updating angular-cli](#updating-angular-cli)

\* [Development Hints for hacking on angular-cli](#development-hints-for-hacking-on-angular-cli)

## Installation

\*\*BEFORE YOU INSTALL:\*\* please read the [prerequisites](#prerequisites)

```bash

npm install -g angular-cli

```

## Usage

```bash

ng help

```

### Generating and serving an Angular2 project via a development server

```bash

ng new PROJECT\_NAME

cd PROJECT\_NAME

ng serve

```

Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

You can configure the default HTTP port and the one used by the LiveReload server with two command-line options :

```bash

ng serve --host 0.0.0.0 --port 4201 --live-reload-port 49153

```

### Generating Components, Directives, Pipes and Services

You can use the `ng generate` (or just `ng g`) command to generate Angular components:

```bash

ng generate component my-new-component

ng g component my-new-component # using the alias

# components support relative path generation

# if in the directory src/app/feature/ and you run

ng g component new-cmp

# your component will be generated in src/app/feature/new-cmp

# but if you were to run

ng g component ../newer-cmp

# your component will be generated in src/app/newer-cmp

```

You can find all possible blueprints in the table below:

Scaffold | Usage

--- | ---

Component | `ng g component my-new-component`

Directive | `ng g directive my-new-directive`

Pipe | `ng g pipe my-new-pipe`

Service | `ng g service my-new-service`

Class | `ng g class my-new-class`

Interface | `ng g interface my-new-interface`

Enum | `ng g enum my-new-enum`

Module | `ng g module my-module`

### Generating a route

Generating routes in the CLI has been disabled for the time being. A new router and new route generation blueprints are coming.

You can read the official documentation for the new Router here: https://angular.io/docs/ts/latest/guide/router.html. Please note that even though route generation is disabled, building your projects with routing is still fully supported.

### Creating a build

```bash

ng build

```

The build artifacts will be stored in the `dist/` directory.

### Build Targets and Environment Files

`ng build` can specify both a build target (`--target=production` or `--target=development`) and an

environment file to be used with that build (`--environment=dev` or `--environment=prod`).

By default, the development build target and environment are used.

The mapping used to determine which environment file is used can be found in `angular-cli.json`:

```json

"environments": {

"source": "environments/environment.ts",

"dev": "environments/environment.ts",

"prod": "environments/environment.prod.ts"

}

```

These options also apply to the serve command. If you do not pass a value for `environment`,

it will default to `dev` for `development` and `prod` for `production`.

```bash

# these are equivalent

ng build --target=production --environment=prod

ng build --prod --env=prod

ng build --prod

# and so are these

ng build --target=development --environment=dev

ng build --dev --e=dev

ng build --dev

ng build

```

You can also add your own env files other than `dev` and `prod` by doing the following:

- create a `src/environments/environment.NAME.ts`

- add `{ "NAME": 'src/environments/environment.NAME.ts' }` to the `apps[0].environments` object in `angular-cli.json`

- use them via the `--env=NAME` flag on the build/serve commands.

### Base tag handling in index.html

When building you can modify base tag (`<base href="/">`) in your index.html with `--base-href your-url` option.

```bash

# Sets base tag href to /myUrl/ in your index.html

ng build --base-href /myUrl/

ng build --bh /myUrl/

```

### Bundling

All builds make use of bundling, and using the `--prod` flag in `ng build --prod`

or `ng serve --prod` will also make use of uglifying and tree-shaking functionality.

### Running unit tests

```bash

ng test

```

Tests will execute after a build is executed via [Karma](http://karma-runner.github.io/0.13/index.html), and it will automatically watch your files for changes. You can run tests a single time via `--watch=false` or `--single-run`.

You can run tests with coverage via `--code-coverage`. The coverage report will be in the `coverage/` directory.

Linting during tests is also available via the `--lint` flag. See [Linting and formatting code](#linting-and-formatting-code) chapter for more informations.

### Running end-to-end tests

```bash

ng e2e

```

Before running the tests make sure you are serving the app via `ng serve`.

End-to-end tests are run via [Protractor](https://angular.github.io/protractor/).

### Proxy To Backend

Using the proxying support in webpack's dev server we can highjack certain urls and send them to a backend server.

We do this by passing a file to `--proxy-config`

Say we have a server running on `http://localhost:3000/api` and we want all calls to `http://localhost:4200/api` to go to that server.

We create a file next to projects `package.json` called `proxy.conf.json`

with the content

```json

{

"/api": {

"target": "http://localhost:3000",

"secure": false

}

}

```

You can read more about what options are available here [webpack-dev-server proxy settings](https://webpack.github.io/docs/webpack-dev-server.html#proxy)

and then we edit the `package.json` file's start script to be

```json

"start": "ng serve --proxy-config proxy.conf.json",

```

now run it with `npm start`

### Deploying the app via GitHub Pages

You can deploy your apps quickly via:

```bash

ng github-pages:deploy --message "Optional commit message"

```

This will do the following:

- creates GitHub repo for the current project if one doesn't exist

- rebuilds the app in production mode at the current `HEAD`

- creates a local `gh-pages` branch if one doesn't exist

- moves your app to the `gh-pages` branch and creates a commit

- edit the base tag in index.html to support github pages

- pushes the `gh-pages` branch to github

- returns back to the original `HEAD`

Creating the repo requires a token from github, and the remaining functionality

relies on ssh authentication for all git operations that communicate with github.com.

To simplify the authentication, be sure to [setup your ssh keys](https://help.github.com/articles/generating-ssh-keys/).

If you are deploying a [user or organization page](https://help.github.com/articles/user-organization-and-project-pages/), you can instead use the following command:

```bash

ng github-pages:deploy --user-page --message "Optional commit message"

```

This command pushes the app to the `master` branch on the github repo instead

of pushing to `gh-pages`, since user and organization pages require this.

### Linting and formatting code

You can lint your app code by running `ng lint`.

This will use the `lint` npm script that in generated projects uses `tslint`.

You can modify the these scripts in `package.json` to run whatever tool you prefer.

### Support for offline applications

\*\*The `--mobile` flag has been disabled temporarily. Sorry for the inconvenience.\*\*

~~Angular-CLI includes support for offline applications via the `--` flag on `ng new`. Support is experimental, please see the angular/mobile-toolkit project and https://mobile.angular.io/ for documentation on how to make use of this functionality.~~

### Commands autocompletion

To turn on auto completion use the following commands:

For bash:

```bash

ng completion 1>> ~/.bashrc 2>>&1

source ~/.bashrc

```

For zsh:

```bash

ng completion 1>> ~/.zshrc 2>>&1

source ~/.zshrc

```

Windows users using gitbash:

```bash

ng completion 1>> ~/.bash\_profile 2>>&1

source ~/.bash\_profile

```

### Project assets

You use the `assets` array in `angular-cli.json` to list files or folders you want to copy as-is when building your project:

```json

"assets": [

"assets",

"favicon.ico"

]

```

### Global styles

The `styles.css` file allows users to add global styles and supports

[CSS imports](https://developer.mozilla.org/en/docs/Web/CSS/@import).

If the project is created with the `--style=sass` option, this will be a `.sass`

file instead, and the same applies to `scss/less/styl`.

You can add more global styles via the `apps[0].styles` property in `angular-cli.json`.

### CSS Preprocessor integration

Angular-CLI supports all major CSS preprocessors:

- sass/scss ([http://sass-lang.com/](http://sass-lang.com/))

- less ([http://lesscss.org/](http://lesscss.org/))

- stylus ([http://stylus-lang.com/](http://stylus-lang.com/))

To use these preprocessors simply add the file to your component's `styleUrls`:

```javascript

@Component({

selector: 'app-root',

templateUrl: 'app.component.html',

styleUrls: ['app.component.scss']

})

export class AppComponent {

title = 'app works!';

}

```

When generating a new project you can also define which extension you want for

style files:

```bash

ng new sassy-project --style=sass

```

Or set the default style on an existing project:

```bash

ng set defaults.styleExt scss

```

### 3rd Party Library Installation

Simply install your library via `npm install lib-name --save` and import it in your code.

If the library does not include typings, you can install them using npm:

```bash

npm install d3 --save

npm install @types/d3 --save-dev

```

If the library doesn't have typings available at `@types/`, you can still use it by

manually adding typings for it:

1. First, create a `typings.d.ts` file in your `src/` folder. This file will be automatically included as global type definition.

2. Then, in `src/typings.d.ts`, add the following code:

```typescript

declare module 'typeless-package';

```

3. Finally, in the component or file that uses the library, add the following code:

```typescript

import \* as typelessPackage from 'typeless-package';

typelessPackage.method();

```

Done. Note: you might need or find useful to define more typings for the library that you're trying to use.

### Global Library Installation

Some javascript libraries need to be added to the global scope, and loaded as if

they were in a script tag. We can do this using the `apps[0].scripts` and

`apps[0].styles` properties of `angular-cli.json`.

As an example, to use [Bootstrap 4](http://v4-alpha.getbootstrap.com/) this is

what you need to do:

First install Bootstrap from `npm`:

```bash

npm install bootstrap@next

```

Then add the needed script files to `apps[0].scripts`:

```json

"scripts": [

"../node\_modules/jquery/dist/jquery.js",

"../node\_modules/tether/dist/js/tether.js",

"../node\_modules/bootstrap/dist/js/bootstrap.js"

],

```

Finally add the Bootstrap CSS to the `apps[0].styles` array:

```json

"styles": [

"../node\_modules/bootstrap/dist/css/bootstrap.css",

"styles.css"

],

```

Restart `ng serve` if you're running it, and Bootstrap 4 should be working on

your app.

### Updating angular-cli

To update `angular-cli` to a new version, you must update both the global package and your project's local package.

Global package:

```bash

npm uninstall -g angular-cli

npm cache clean

npm install -g angular-cli@latest

```

Local project package:

```bash

rm -rf node\_modules dist tmp

npm install --save-dev angular-cli@latest

npm install

ng init

```

Running `ng init` will check for changes in all the auto-generated files created by `ng new` and allow you to update yours. You are offered four choices for each changed file: `y` (overwrite), `n` (don't overwrite), `d` (show diff between your file and the updated file) and `h` (help).

Carefully read the diffs for each code file, and either accept the changes or incorporate them manually after `ng init` finishes.

\*\*The main cause of errors after an update is failing to incorporate these updates into your code\*\*.

You can find more details about changes between versions in [CHANGELOG.md](https://github.com/angular/angular-cli/blob/master/CHANGELOG.md).

## Development Hints for hacking on angular-cli

### Working with master

```bash

git clone https://github.com/angular/angular-cli.git

cd angular-cli

npm link

```

`npm link` is very similar to `npm install -g` except that instead of downloading the package

from the repo, the just cloned `angular-cli/` folder becomes the global package.

Any changes to the files in the `angular-cli/` folder will immediately affect the global `angular-cli` package,

allowing you to quickly test any changes you make to the cli project.

Now you can use `angular-cli` via the command line:

```bash

ng new foo

cd foo

npm link angular-cli

ng serve

```

`npm link angular-cli` is needed because by default the globally installed `angular-cli` just loads

the local `angular-cli` from the project which was fetched remotely from npm.

`npm link angular-cli` symlinks the global `angular-cli` package to the local `angular-cli` package.

Now the `angular-cli` you cloned before is in three places:

The folder you cloned it into, npm's folder where it stores global packages and the `angular-cli` project you just created.

You can also use `ng new foo --link-cli` to automatically link the `angular-cli` package.

Please read the official [npm-link documentation](https://www.npmjs.org/doc/cli/npm-link.html)

and the [npm-link cheatsheet](http://browsenpm.org/help#linkinganynpmpackagelocally) for more information.

## License

MIT

[travis-badge]: https://travis-ci.org/angular/angular-cli.svg?branch=master

[travis-badge-url]: https://travis-ci.org/angular/angular-cli

[david-badge]: https://david-dm.org/angular/angular-cli.svg

[david-badge-url]: https://david-dm.org/angular/angular-cli

[david-dev-badge]: https://david-dm.org/angular/angular-cli/dev-status.svg

[david-dev-badge-url]: https://david-dm.org/angular/angular-cli?type=dev

[npm-badge]: https://img.shields.io/npm/v/angular-cli.svg

[npm-badge-url]: https://www.npmjs.com/package/angular-cli