Full-stack with Angular and Spring boot.

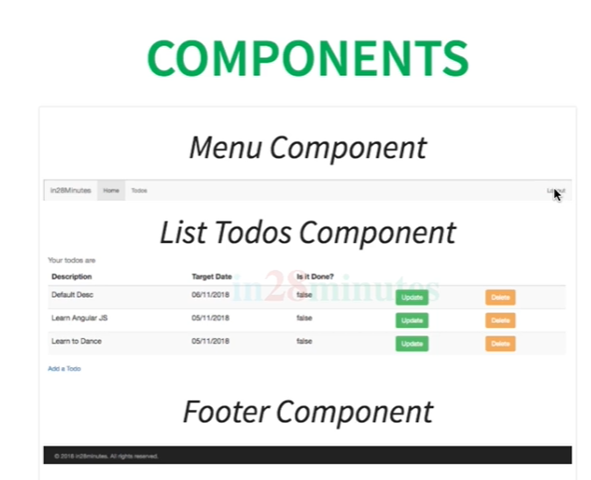
# Exploring Angular CLI commands

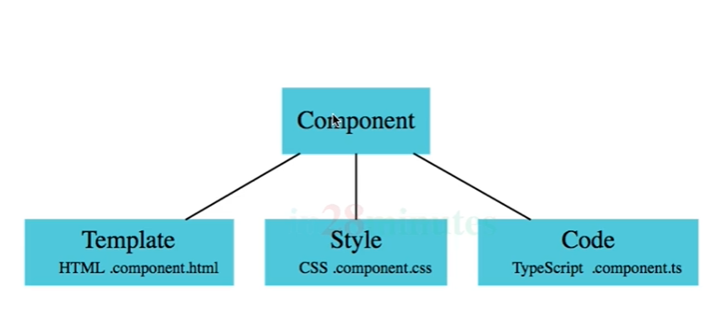
|  |  |  |
| --- | --- | --- |
| **COMMAND** | **ALIAS** | **DESCRIPTION** |
| add |  | Adds support for an external library to your project. |
| analytics |  | Configures the gathering of Angular CLI usage metrics. See https://angular.io/cli/usage-analytics-gathering. |
| build | b | Compiles an Angular app into an output directory named dist/ at the given output path. Must be executed from within a workspace directory. |
| config |  | Retrieves or sets Angular configuration values in the angular.json file for the workspace. |
| deploy |  | Invokes the deploy builder for a specified project or for the default project in the workspace. |
| doc | d | Opens the official Angular documentation (angular.io) in a browser, and searches for a given keyword. |
| e2e | e | Builds and serves an Angular app, then runs end-to-end tests using Protractor. |
| generate | g | Generates and/or modifies files based on a schematic. |
| help |  | Lists available commands and their short descriptions. |
| lint | l | Runs linting tools on Angular app code in a given project folder. |
| new | n | Creates a new workspace and an initial Angular app. |
| run |  | Runs an Architect target with an optional custom builder configuration defined in your project. |
| serve | s | Builds and serves your app, rebuilding on file changes. |
| test | t | Runs unit tests in a project. |
| update |  | Updates your application and its dependencies. See https://update.angular.io/ |
| version | v | Outputs Angular CLI version. |
| xi18n | i18n-extract | Extracts i18n messages from source code. |

## Angular Project Structure

|  |  |
| --- | --- |
| **WORKSPACE CONFIG FILES** | **PURPOSE** |
| .editorconfig | [Configuration for code editors. See EditorConfig.](https://editorconfig.org/) |
| .gitignore | [Specifies intentionally untracked files that Git should ignore.](https://git-scm.com/) |
| README.md | Introductory documentation for the root app. |
| angular.json | CLI configuration defaults for all projects in the workspace, including configuration options for build, serve, and test tools that the CLI uses, such as TSLint, Karma, and Protractor. For details, see Angular Workspace Configuration. |
| package.json | Configures npm package dependencies that are available to all projects in the workspace. See npm documentation for the specific format and contents of this file. |
| package-lock.json | Provides version information for all packages installed into node\_modules by the npm client. See npm documentation for details. If you use the yarn client, this file will be yarn.lock instead. |
| src/ | Source files for the root-level application project. |
| node\_modules/ | [Provides npm packages to the entire workspace. Workspace-wide node\_modules dependencies are visible to all projects.](https://angular.io/guide/npm-packages) |
| tsconfig.json | [The tsconfig.json file is a "Solution Style" TypeScript configuration file. Code editors and TypeScript’s language server use this file to improve development experience. Compilers do not use this file.](https://www.typescriptlang.org/docs/handbook/release-notes/typescript-3-9.html#support-for-solution-style-tsconfigjson-files) |
| tsconfig.base.json | The base TypeScript configuration for projects in the workspace. All other configuration files inherit from this base file. For more information, see the Configuration inheritance with extends section of the TypeScript documentation. |
| tslint.json | [Default TSLint configuration for projects in the workspace.](https://palantir.github.io/tslint/) |

# Angular components





Refer : <https://angular.io/guide/displaying-data>

# **Displaying data in views**

Angular [components](https://angular.io/guide/glossary#component) form the data structure of your application. The HTML [template](https://angular.io/guide/glossary#template) associated with a component provides the means to display that data in the context of a web page. Together, a component's class and template form a [view](https://angular.io/guide/glossary#view) of your application data.

The process of combining data values with their representation on the page is called [data binding](https://angular.io/guide/glossary#data-binding). You display your data to a user (and collect data from the user) by binding controls in the HTML template to the data properties of the component class.

In addition, you can add logic to the template by including [directives](https://angular.io/guide/glossary#directive), which tell Angular how to modify the page as it is rendered.

Angular defines a template language that expands HTML notation with syntax that allows you to define various kinds of data binding and logical directives. When the page is rendered, Angular interprets the template syntax to update the HTML according to your logic and current data state. Before you read the complete [template syntax guide](https://angular.io/guide/template-syntax), the exercises on this page give you a quick demonstration of how template syntax works.

## **Showing component properties with interpolation**

The easiest way to display a component property is to bind the property name through interpolation. With interpolation, you put the property name in the view template, enclosed in double curly braces: {{myHero}}.

Use the CLI command [ng new displaying-data](https://angular.io/cli/new) to create a workspace and app named displaying-data.

Delete the app.component.html file. It is not needed for this example.

Then modify the app.component.ts file by changing the template and the body of the component.

When you're done, it should look like this:

src/app/app.component.ts

content\_copyimport { [Component](https://angular.io/api/core/Component) } from '@angular/core';

@[Component](https://angular.io/api/core/Component)({

selector: 'app-root',

template: `

<h1>{{title}}</h1>

<h2>My favorite hero is: {{myHero}}</h2>

`

})

export class AppComponent {

title = 'Tour of Heroes';

myHero = 'Windstorm';

}

You added two properties to the formerly empty component: title and myHero.

The template displays the two component properties using double curly brace interpolation:

src/app/app.component.ts (template)

content\_copytemplate: `

<h1>{{title}}</h1>

<h2>My favorite hero is: {{myHero}}</h2>

`

The template is a multi-line string within ECMAScript 2015 backticks (`). The backtick (`)—which is not the same character as a single quote (')—allows you to compose a string over several lines, which makes the HTML more readable.

Angular automatically pulls the value of the title and myHero properties from the component and inserts those values into the browser. Angular updates the display when these properties change.

More precisely, the redisplay occurs after some kind of asynchronous event related to the view, such as a keystroke, a timer completion, or a response to an HTTP request.

Notice that you don't call **new** to create an instance of the AppComponent class. Angular is creating an instance for you. How?

The CSS selector in the @[Component](https://angular.io/api/core/Component) decorator specifies an element named <app-root>. That element is a placeholder in the body of your index.html file:

src/index.html (body)

content\_copy<body>

<app-root></app-root>

</body>

# Create new component

ng generate component welcome

This command will create a WelcomeComponent in your workspace and automatically links it to your existing AppComponent.

To render the welcome component in the main page. Add the WelcomeComponent in the app.module.ts file. Go back to your browser to see the changes.

