Slide Note

Learning Consolidation

Develop SPA Using Angular Components

Menu

00:01:00:00:29

01/ 29

Slide Note

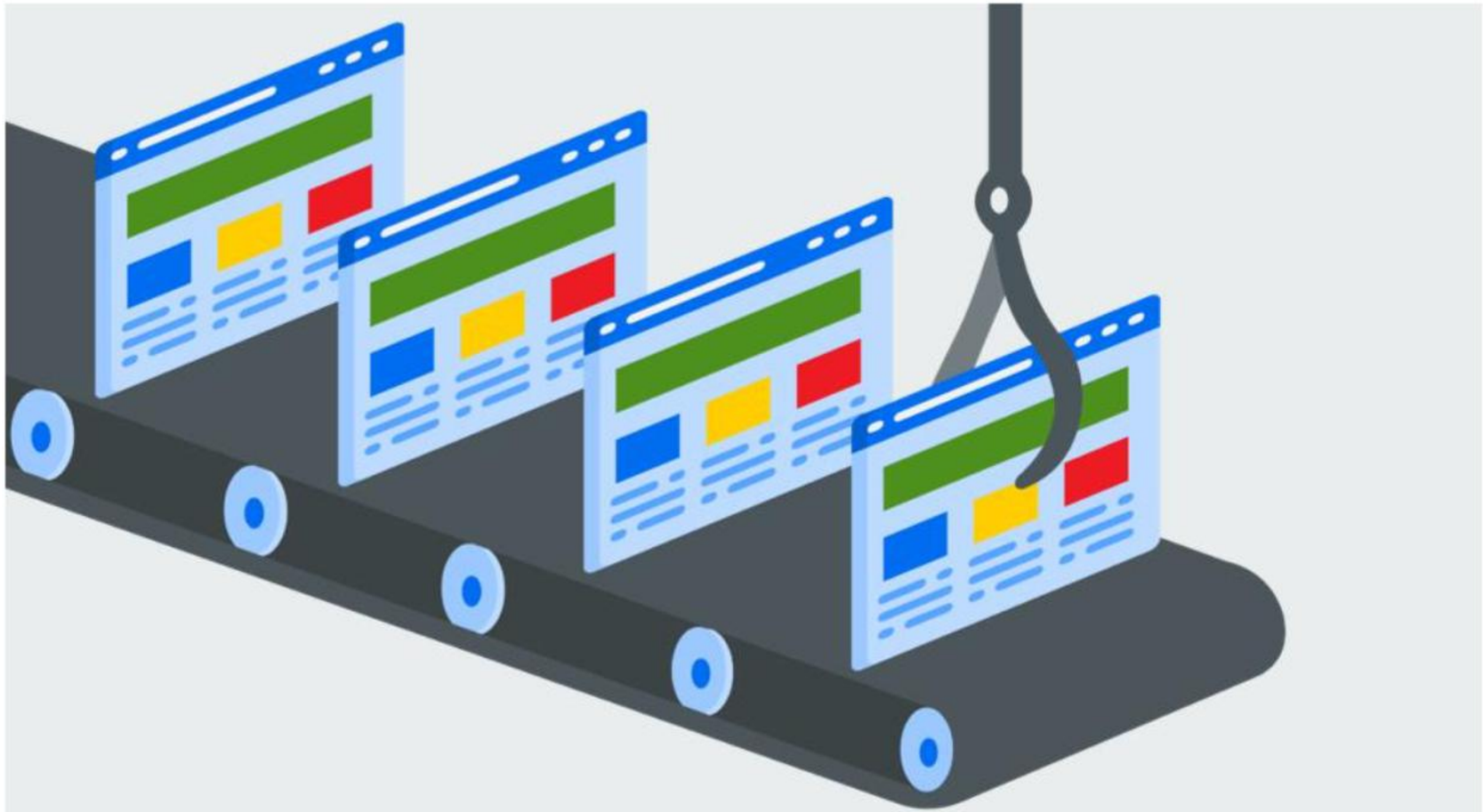
In this sprint, you learned to:

- Differentiate between Multi-page Application (MPA) and Single-page Application (SPA)
- Create an Angular project using the CLI tool
- Create Angular components, the main building block for Angular applications
- Design user interface for views using component templates
- Explore the file and folder structure of an Angular project
- Add behavior and data to views using interpolation and property binding in templates
- Handle the events raised by user actions using event binding
- Explain two-way binding using ngModel

Slide Note

Multi-page applications consists of several pages with static information (text, images, etc.) and links to the other pages with the same content.

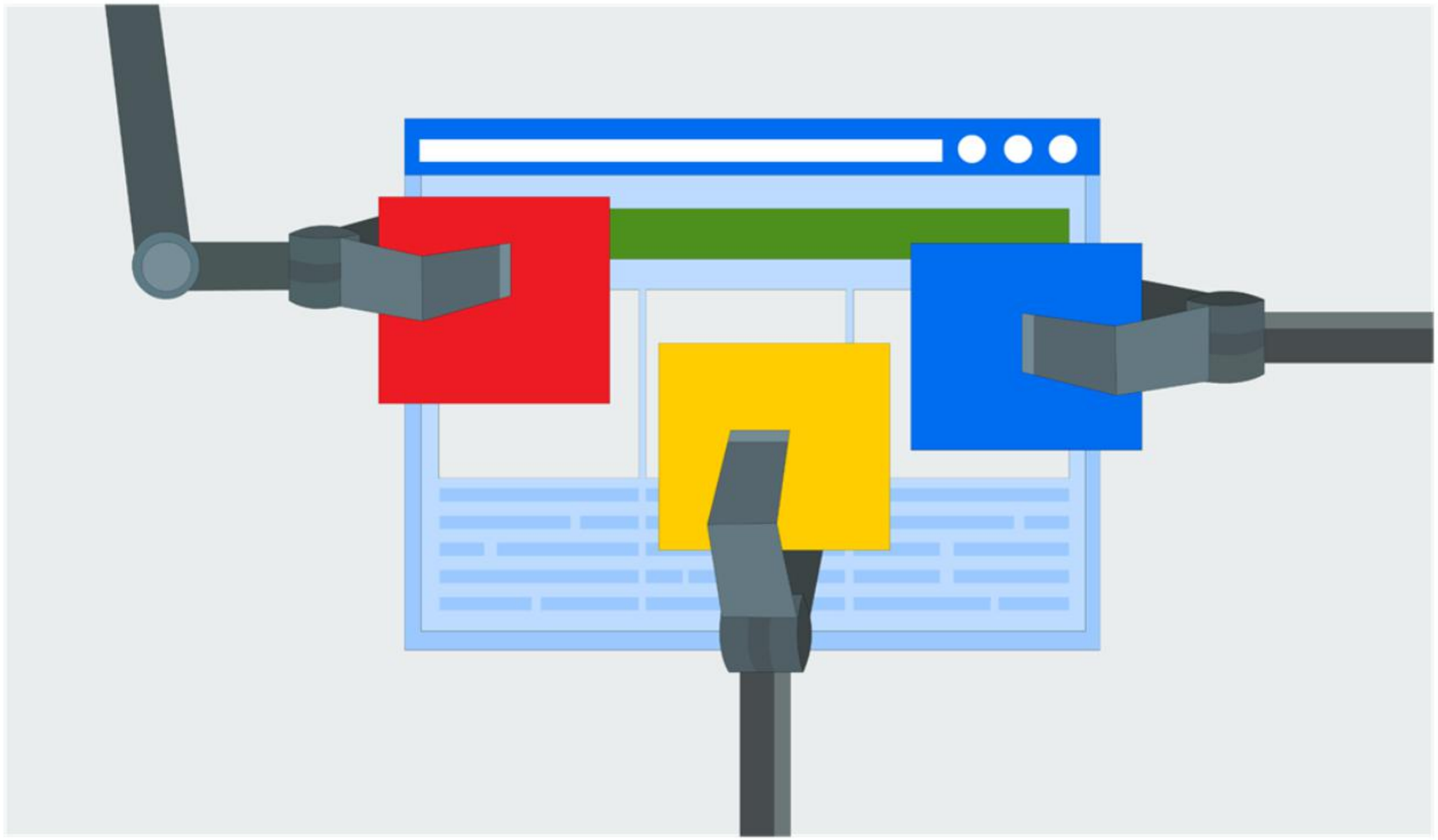
MPA Approach



Slide Note

Single-page applications consist of just one single page that works inside a browser and does not require page reloading during use.

SPA Approach



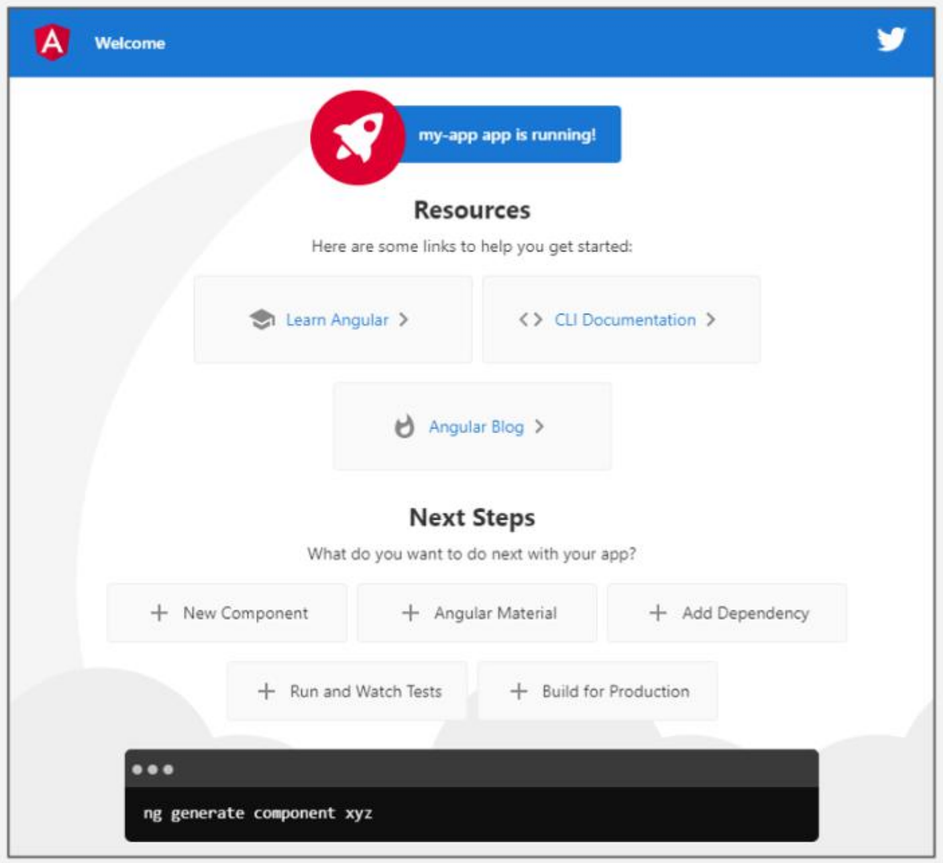
Slide Note

Angular Framework

- Angular is a component-based framework for building scalable web applications.
- It is a development platform built on TypeScript for creating efficient and sophisticated single-page apps.
- It is a well-integrated library collection that covers various features, including routing, forms management, client-server communication, and much more.
- Angular includes a suite of developer tools to help you develop, build, test, and update your code.

Slide Note

Delete the default content present in the app.component.html to start working with your project.



Angular Application Output created using Angular CLI command

Install Angular CLI

- Install Angular CLI using the command in the terminal window:

  npm install -g @angular/cli

- To create a new workspace and initial starter app, use the "ng new" command, which prompts for information about features to include in the initial app.

  ng new first-app

- Run the application with the included server after navigating to the workspace folder.

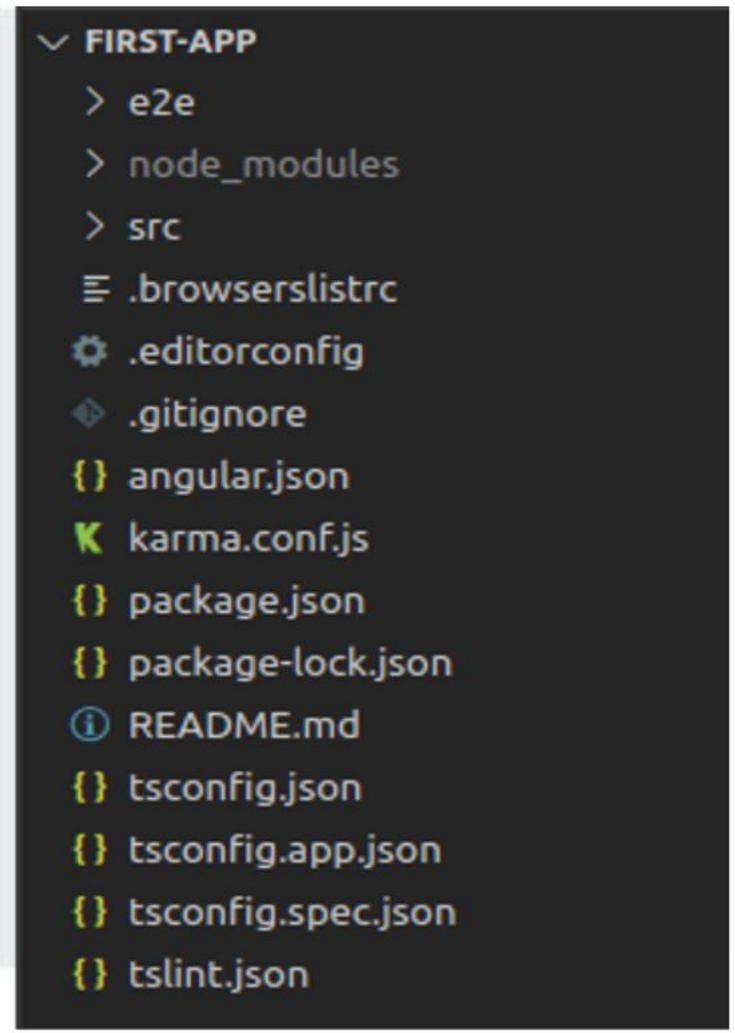
  cd first-app

  ng serve --open

Slide Note

Workspace and Project Files

- The ng new command creates an Angular workspace folder and generates a new application skeleton.
- A workspace can contain multiple applications and libraries.
- The initial application created by the ng new command is at the top level of the workspace.
- Generating an additional application or library in a workspace goes into a project/subfolder.
- A newly generated application contains the source files for a root module with a root component and template.
- Each application has a src folder that contains the logic, data, and assets.



Slide Note

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
build	b	Compiles an Angular app into an output directory named dist/ at the given output path
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		Builds and serves an Angular app, then runs end-to-end tests
extract-i18n	i18n-extract xi18n	Extracts i18n messages from source code

Slide Note

Angular CLI Commands (Contd.)

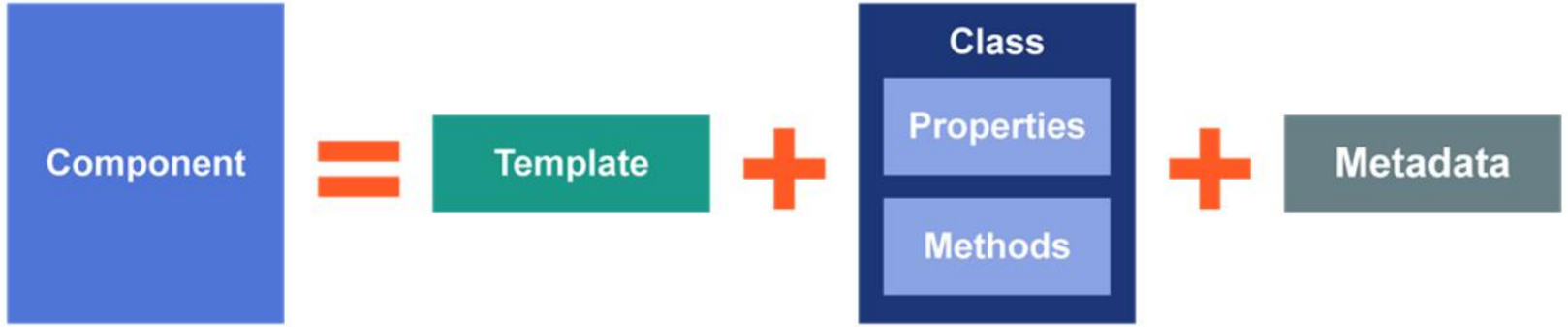
Command	Alias	Description
generate	g	Generates and/or modifies files based on a schematic
help		Lists available commands and their short descriptions
lint	l	Runs linting tools on the Angular app code in a given project folder
new	n	Creates an Angular workspace
run		Runs an Architect target with an optional 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
version	v	Outputs Angular CLI version

Slide Note

Angular Component

Components are the main building block for Angular applications. Each component consists of:

- An HTML template that declares what renders on the page.
- A TypeScript class that defines behavior.
- A CSS selector that defines how the component is used in a template.
- Optionally, CSS styles are applied to the template.



Slide Note

The above image lists the various terms used in defining an Angular component.

• The import statement in the first line is used to import the Angular Component decorator from the @angular/core library.

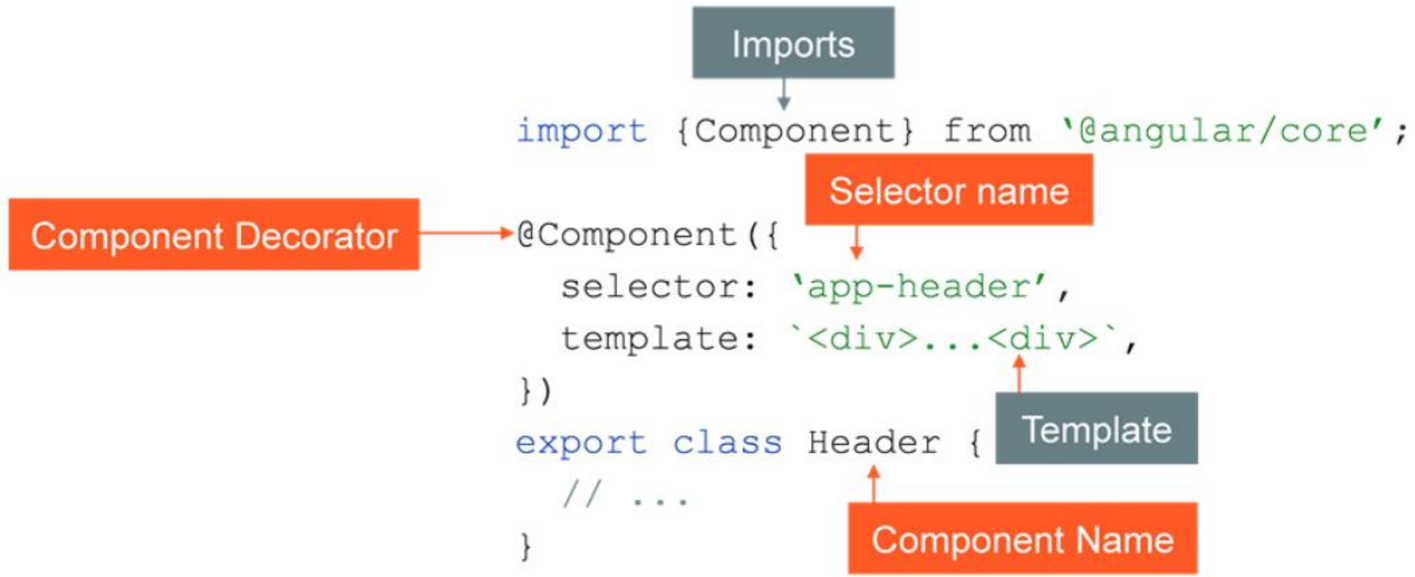
• @Component decorator: The metadata for a component tells Angular where to get the major building blocks needed to create and present the component and its view.

• Inside the @Component decorator, the component's CSS selector is added using a selector property.

• A template property is added inside the @Component decorator to define the template within the component.

• The component class Header is exported to be used in other modules.

Angular Component Class



Slide Note

tslint.json: Configuration file for typescript lint errors.

Folders and Files in an Angular Project

Folder/File Name	Description
angular.json	Provides workspace-wide and project-specific configuration defaults for build and development tools provided by the Angular CLI
package.json	Includes *a starter* set of packages used by all projects in the workspace
package.lock.json	Automatically generates for any operations where npm modifies either the node_modules tree or package.json
tsconfig.json	Specifies the base TypeScript and Angular compiler options that all projects in the workspace inherit
tsconfig.app.json	Additional config file that allows you to adjust your config on an app basis
src	Contains the main code files related to the angular application
e2e	Contains all End-to-End (e2e) test script files to perform e2e testing
node_modules	Has all downloaded packages from NPM listed in packed.json as dependencies

Slide Note

tslint.json: Configuration file for typescript lint errors.

Folders and Files in an Angular Project

Folder/File Name	Description
angular.json	Provides workspace-wide and project-specific configuration defaults for build and development tools provided by the Angular CLI
package.json	Includes *a starter* set of packages used by all projects in the workspace
package.lock.json	Automatically generates for any operations where npm modifies either the node_modules tree or package.json
tsconfig.json	Specifies the base TypeScript and Angular compiler options that all projects in the workspace inherit
tsconfig.app.json	Additional config file that allows you to adjust your config on an app basis
src	Contains the main code files related to the angular application
e2e	Contains all End-to-End (e2e) test script files to perform e2e testing
node_modules	Has all downloaded packages from NPM listed in packed.json as dependencies

Slide Note

Go to the angular.json file and explore the important properties like:

• **Workspace configuration**: version, newProjectRoot, deafultProject, schematics, and projects.

• **Project Configuration Options**: root, sourceRoot, projectType, prefix, schematics, and architect.

• **Project Tool configuration Options**: The architect is the tool that the CLI uses to perform complex tasks, such as compilation and test running. Architect is a shell that runs a specified builder to perform a given task according to a target configuration.

Angular defines default builders for use with specific CLI commands or with the general ng run command.

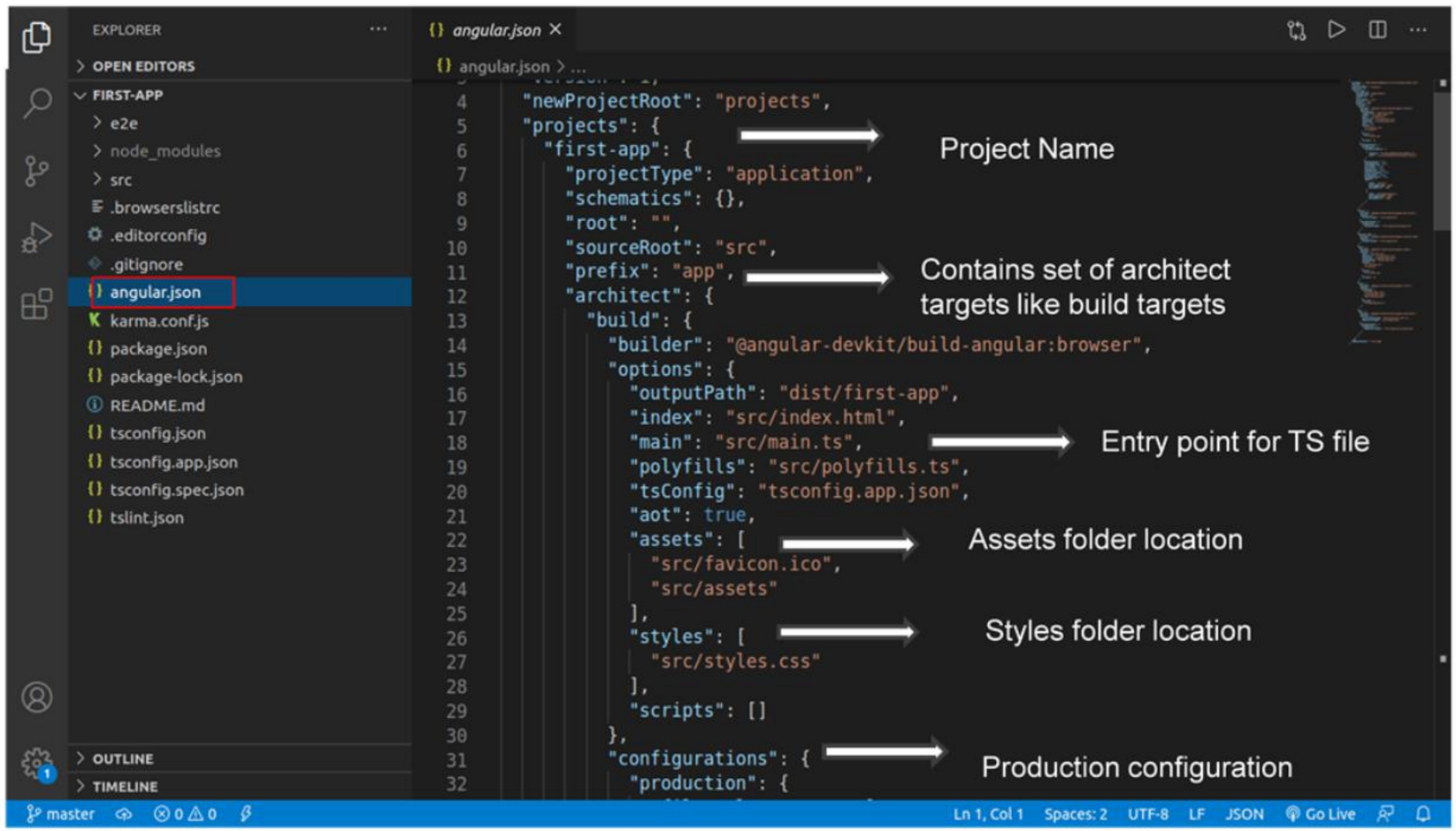
The architect section of angular.json contains a set of Architect targets. Many of the targets correspond to the CLI commands that run them.

Each target specifies a builder for that target, Options, and Configuration properties for setting defaults.

Some of the targets are:

"architect": { "build": { }, "serve": { }, "e2e" : { }, "test": { }, "lint": { }, "extract-i18n": { }, "server": { }, "app-shell": { } }

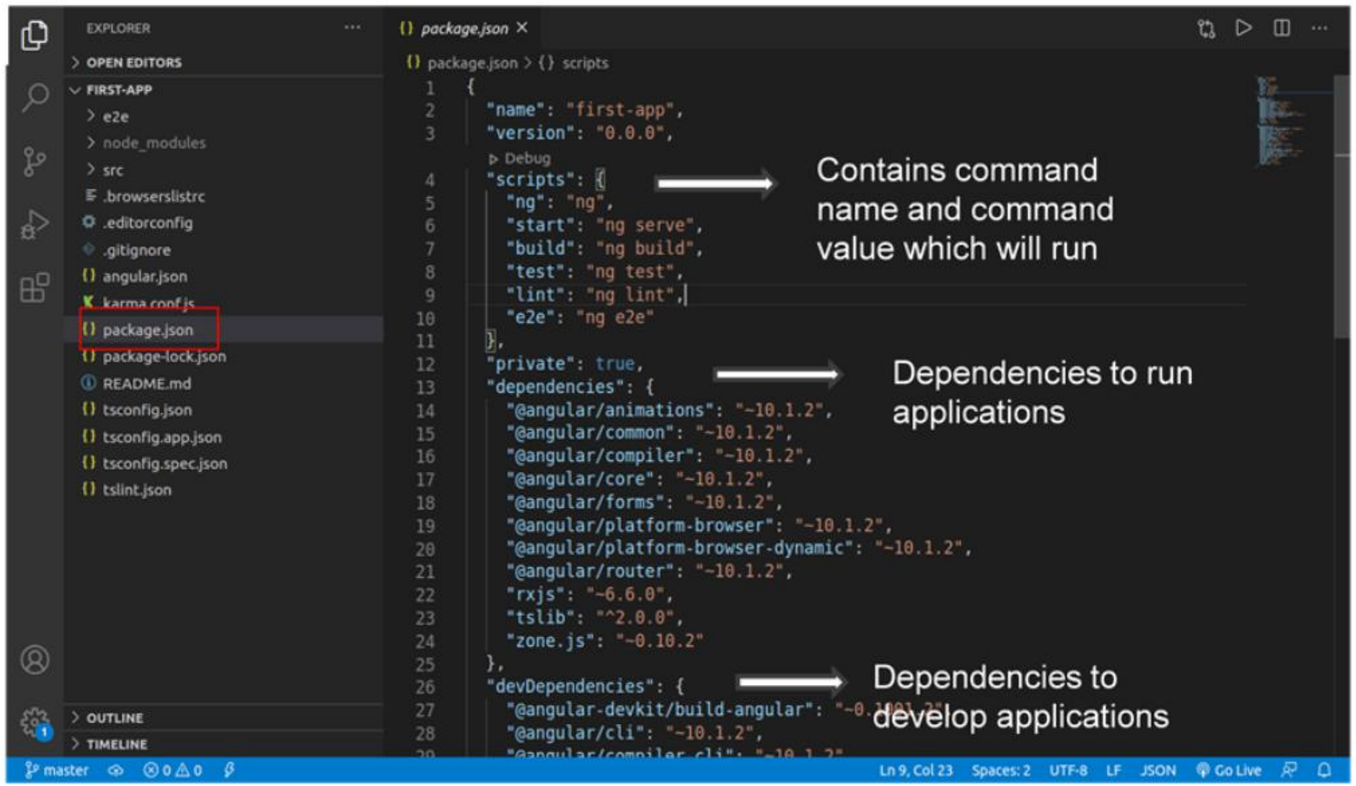
angular.json Configuration File Overview



Slide Note

Initially, the package.json includes *a starter set of packages*, some of which are required by Angular and others that support common application scenarios.

package.json Configuration File



The package.json is organized into two groups of packages:

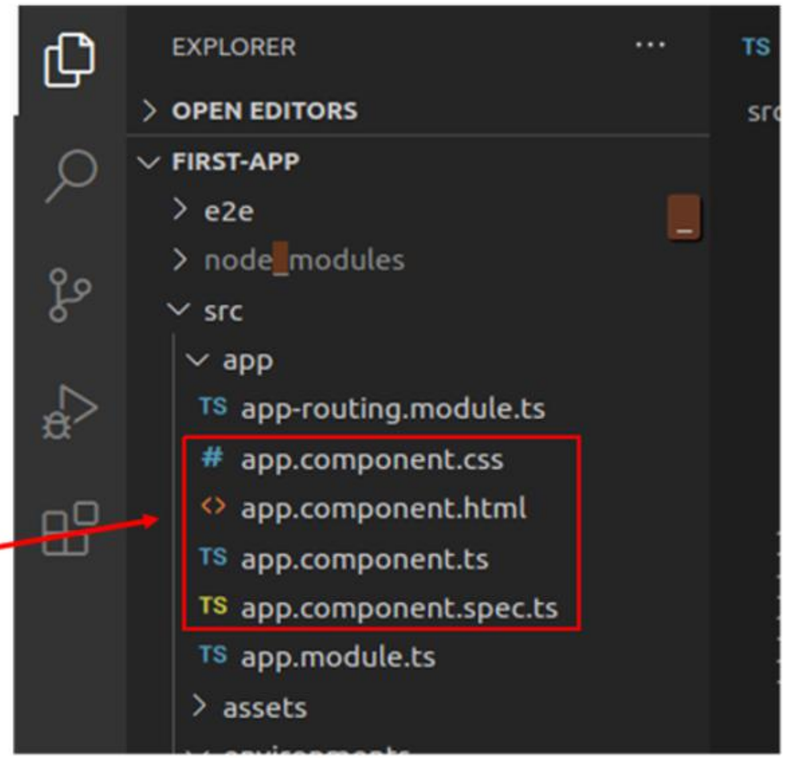
- Dependencies are essential for *running* applications.
- DevDependencies are only necessary to *develop* applications.

Slide Note

When the angular app is served and opened in the browser, the index.html file in the src folder calls the app-root root component. The root component is defined in app.components.ts which targets app.component.html.

Index.html

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>First App</title>
<base href="/">
<meta name="viewport" content="width=device-
width, initial-scale=1">
<link rel="icon" type="image/x-icon"
href="favicon.ico">
</head>
<body>
<app-root></app-root>
</body>
</html>
```



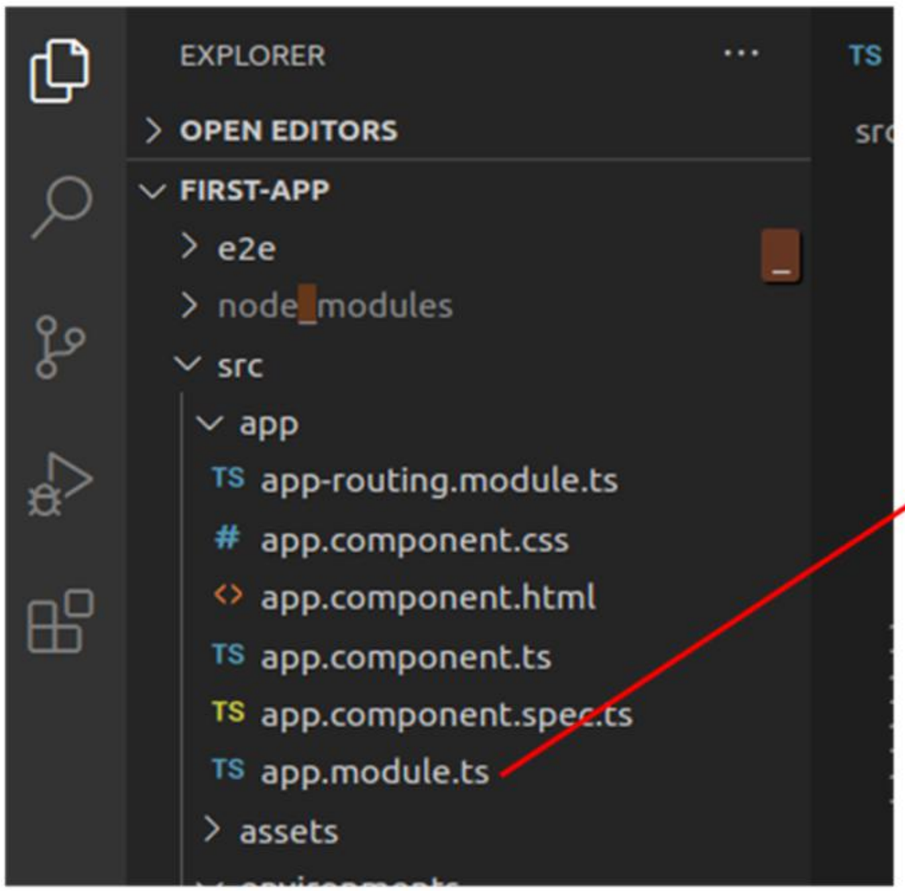
Slide Note

The **main.ts** file is the first code that gets executed. The job of **main.ts** is to bootstrap the application. It loads everything and controls the startup of the application.

platformBrowserDyanmic().bootstrap (AppModule) tells the builder to start the app from the AppModule. AppModule refers to the **app.module.ts** file.

environment.production is a boolean constant declared in environment.ts whose value will be true for the production environment.

main.ts File




```
import { enableProdMode } from
'@angular/core';
import { platformBrowserDynamic } from
'@angular/platform-browser-dynamic';

import { AppModule } from
'./app/app.module';
import { environment } from
'./environments/environment';

if (environment.production) {
enableProdMode();
}

platformBrowserDynamic().bootstrapModule(App
Module)
.catch(err => console.error(err));
```

Slide Note

app.module.ts

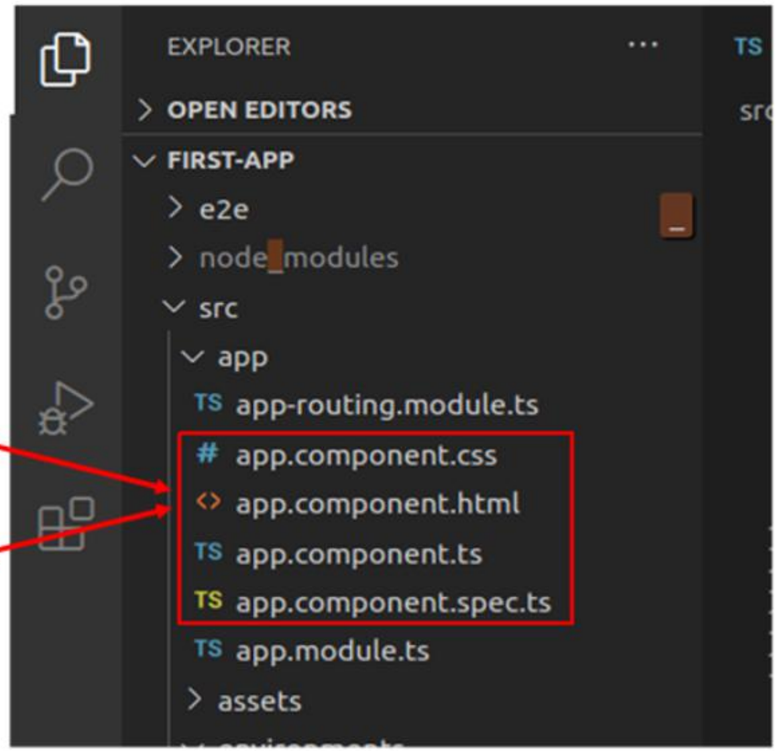
This module created with the **@NgModule decorator,** which has declarations of all the components we are creating within the app module so that angular is aware of them. Here, we also have an imports array where we can import other modules and use them in our app.

There is a list of all components that should be known to Angular when it analyzes our **index.html** file.

app.module.ts

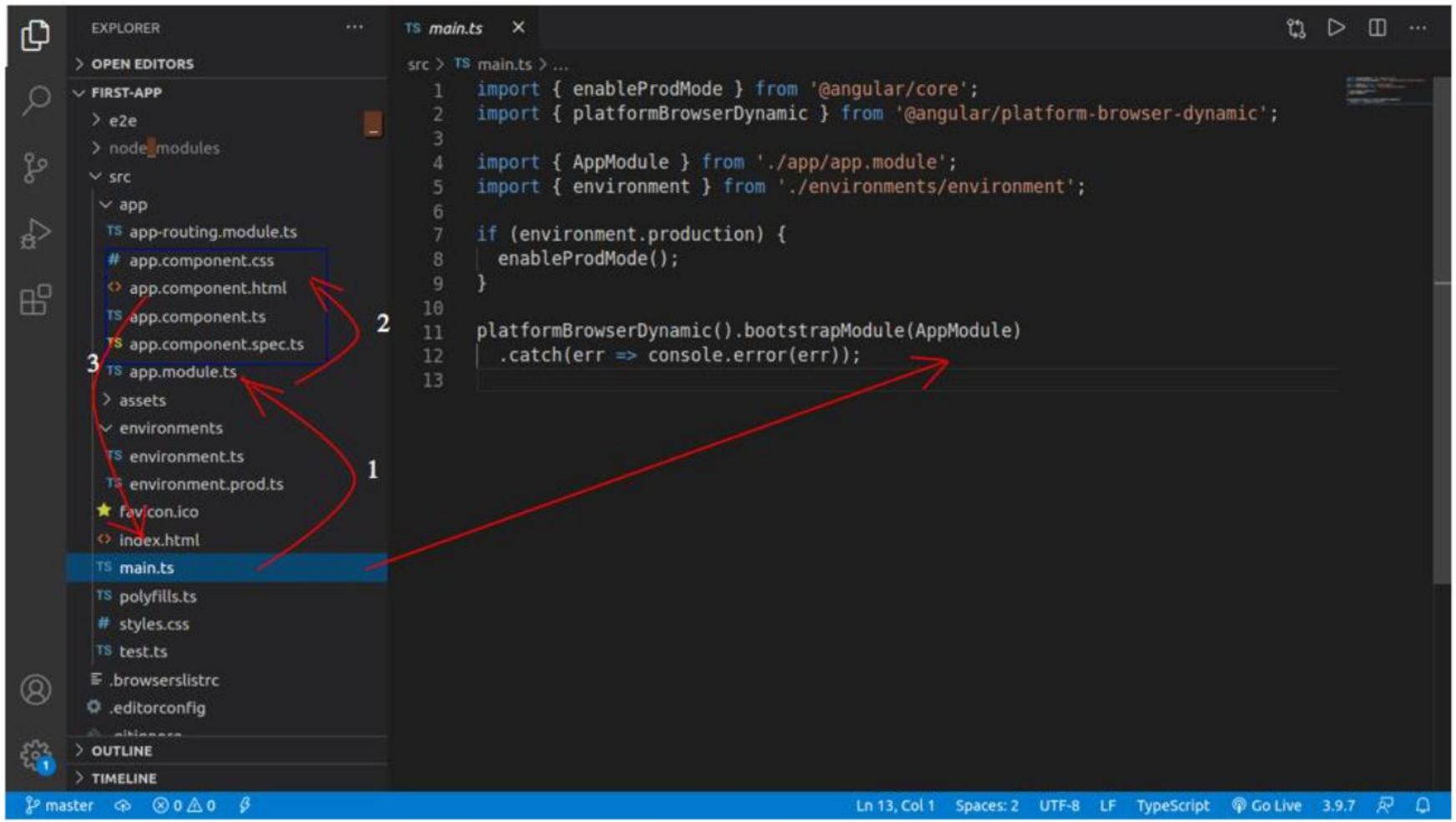
```
import { browserModule} from '@angular/platform-
browser';
import { NgModule} from '@angular/corec';

import { AppRoutingModule } from './app-
routing.module';
import { AppComponent} from './app/componnet';

@NgModule({
declarations: [
AppComponent
],
imports: [
BrowserModule,
AppRoutingModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```

Start with AppComponent



Slide Note

How Does Angular Trigger index.html and Start Working?



Slide Note

Built-in Directives

- Directives are classes that add additional behavior to elements in your Angular applications.
- Angular built-in directives manage forms, lists, styles, etc.
- Different types of directives:
  - Components: Most common directive used with the template.
  - Attribute Directives: Change the appearance or behavior of an element, component, or another directive.
  - Structural Directives: Change the DOM layout by adding and removing DOM elements.
- Built-in Structural Directives:
  - *NgIf: Conditionally creates or disposes of subviews from the template.

```
<div *ngIf="fruit" class="name">{{fruit.name}}</div>
```

  - *NgFor: Repeat a node for each item in a list.

```
<div *ngFor="let fruit of fruits">{{fruit.name}}</div>
```

  - *NgSwitch: A set of directives that switch among alternative views.

Check this link to know more about built-in attribute directives.

Slide Note

Data Binding Markup: Summary

DOM

{{expression}}

Interpolation

[property] = “expression”

One-way Binding

(event) = “statement”

Event Binding

[(ngModel)] = “property”

Two-way Binding

Component

Slide Note

Self-Check

Which Angular module is needed to use NgIf, NgFor?

1. BrowserModule
2. FormsModule
3. CommonModule
4. HttpClientModule

Slide Note

Self-Check: Solution

Which Angular module is needed to use NgIf, NgFor?

1. **BrowserModule**
2. FormsModule
3. CommonModule
4. HttpClientModule

Slide Note

Self-Check

When an Angular application is built using ng build in the Angular CLI, where are the built/compiled files stored?

1. In the /out folder
2. In memory
3. In /dist folder
4. Angular CLI cannot serve the files

Slide Note

Self-Check: Solution

When an Angular application is built using ng build in the Angular CLI, where are the built/compiled files stored?

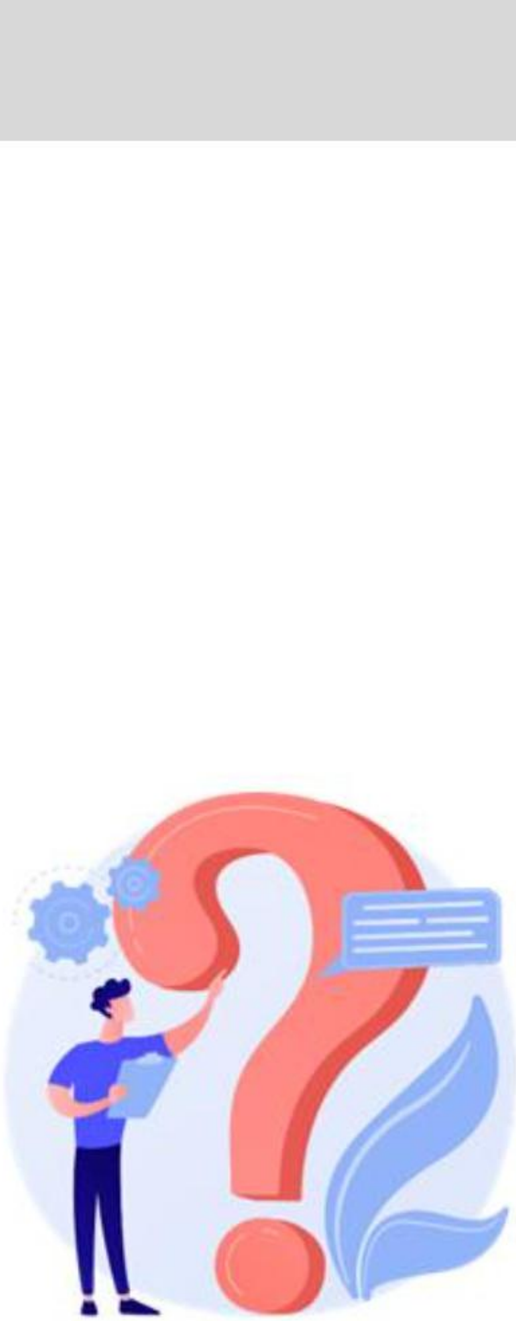
1. In the /out folder
2. In memory
3. **In /dist folder**
4. Angular CLI cannot serve the files

Slide Note

Self-Check

What is the use of angular.json?

1. Used to compile Angular project.
2. Used to link external files.
3. Used to install required project packages.
4. Used to configure Angular project.



Slide Note

Self-Check: Solution

What is the use of angular.json?

1. Used to compile Angular project.
2. Used to link external files.
3. Used to install required project packages.
4. **Used to configure Angular project.**

Slide Note

Self-Check

Which of the following statements are true? Choose multiple options if applicable.

1. "ng generate" command can be used to generate only components and services.
2. "ng generate" component creates a component with the default name.
3. "ng serve" bundles all the code, builds the application, and makes it available in the development server at 4200 port.
4. "ng new new-app" creates a project with new-app as the name.

Slide Note

Self-Check: Solution

Which of the following statements are true? Choose multiple options if applicable.

1. "ng generate" command can be used to generate only components and services.
2. "ng generate" component creates a component with the default name.
3. **"ng serve" bundles all the code, builds the application, and makes it available in the development server at 4200 port.**
4. **"ng new new-app" creates a project with new-app as the name.**