Skip to main content

Angular components overview

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 applied to the template

This topic describes how to create and configure an Angular component.

> To view or download the example code used in this topic, see the live example / download example.

Prerequisites

To create a component, verify that you have met the following prerequisites:

1. Install the Angular CLI.
2. Create an Angular workspace with initial application. If you don't have a project, create one using `ng new <project-name>`, where `<project-name>` is the name of your Angular application.

Skip to main content

Creating a component

The best way to create a component is with the Angular CLI. You can also create a component manually.

Creating a component using the Angular CLI

To create a component using the Angular CLI:

1. From a terminal window, navigate to the directory containing your application.
2. Run the `ng generate component <component-name>` command, where `<component-name>` is the name of your new component.

By default, this command creates the following:

- A directory named after the component
- A component file, `<component-name>.component.ts`
- A template file, `<component-name>.component.html`
- A CSS file, `<component-name>.component.css`
- A testing specification file, `<component-name>.component.spec.ts`

Where `<component-name>` is the name of your component.

> You can change how `ng generate component` creates new components. For more information, see ng generate component in the Angular CLI documentation.

Creating a component manually

Although the Angular CLI is the best way to create an Angular component, you can also create a component manually. This section describes how to create the core component file within an existing Angular project.

Skip to main content

onent manually:

Skip to main content

1. Angular project directory.
2. Create a new file, `<component-name>.component.ts`.
3. At the top of the file, add the following import statement.

```
import { Component } from '@angular/core';
```

4. After the `import` statement, add a `@Component` decorator.

```
@Component({
})
```

5. Choose a CSS selector for the component.

```
@Component({
  selector: 'app-component-overview',
})
```

For more information on choosing a selector, see Specifying a component's selector.

6. Define the HTML template that the component uses to display information. In most cases, this template is a separate HTML file.

```
@Component({
  selector: 'app-component-overview',
  templateUrl: './component-overview.component.html',
})
```

For more information on defining a component's template, see Defining a component's template.

7. Select the styles for the component's template. In most cases, you define the styles for your component's template in a separate file.

Skip to main content

```
({
  : 'app-component-overview',
    templateUrl: './component-
overview.component.html',
    styleUrls: ['./component-
overview.component.css']
})
```

8. Add a `class` statement that includes the code for the component.

```
export class ComponentOverviewComponent {

}
```

Specifying a component's CSS selector

Every component requires a CSS *selector*. A selector instructs Angular to instantiate this component wherever it finds the corresponding tag in template HTML. For example, consider a component `hello-world.component.ts` that defines its selector as `app-hello-world`. This selector instructs Angular to instantiate this component any time the tag `<app-hello-world>` appears in a template.

Specify a component's selector by adding a `selector` statement to the `@Component` decorator.

```
@Component({
  selector: 'app-component-overview',
})
```

Skip to main content

Defining a component's template

A template is a block of HTML that tells Angular how to render the component in your application. Define a template for your component in one of two ways: by referencing an external file, or directly within the component.

To define a template as an external file, add a `templateUrl` property to the `@Component` decorator.

```
@Component({
  selector: 'app-component-overview',
  templateUrl: './component-overview.component.html',
})
```

To define a template within the component, add a `template` property to the `@Component` decorator that contains the HTML you want to use.

```
@Component({
  selector: 'app-component-overview',
  template: '<h1>Hello World!</h1>',
})
```

If you want your template to span multiple lines, use backticks (`` ` ``). For example:

```
@Component({
  selector: 'app-component-overview',
  template: `
    <h1>Hello World!</h1>
    <p>This template definition spans multiple lines.
</p>
  `
})
```

Skip to main content

ponent requires a template defined using `template` or `templateUrl`. You cannot have both statements in a component.

Declaring a component's styles

Declare component styles used for its template in one of two ways: By referencing an external file, or directly within the component.

To declare the styles for a component in a separate file, add a `styleUrls` property to the `@Component` decorator.

```
@Component({
  selector: 'app-component-overview',
  templateUrl: './component-overview.component.html',
  styleUrls: ['./component-overview.component.css']
})
```

To declare the styles within the component, add a `styles` property to the `@Component` decorator that contains the styles you want to use.

```
@Component({
  selector: 'app-component-overview',
  template: '<h1>Hello World!</h1>',
  styles: ['h1 { font-weight: normal; }']
})
```

The `styles` property takes an array of strings that contain the CSS rule declarations.

Next steps

- For an architectural overview of components, see Introduction to components and templates
- For additional options to use when creating a component, see Component in the API Reference
- For more information on styling components, see Component styles
- For more information on templates, see Template syntax

Last reviewed on Mon Feb 28 2022