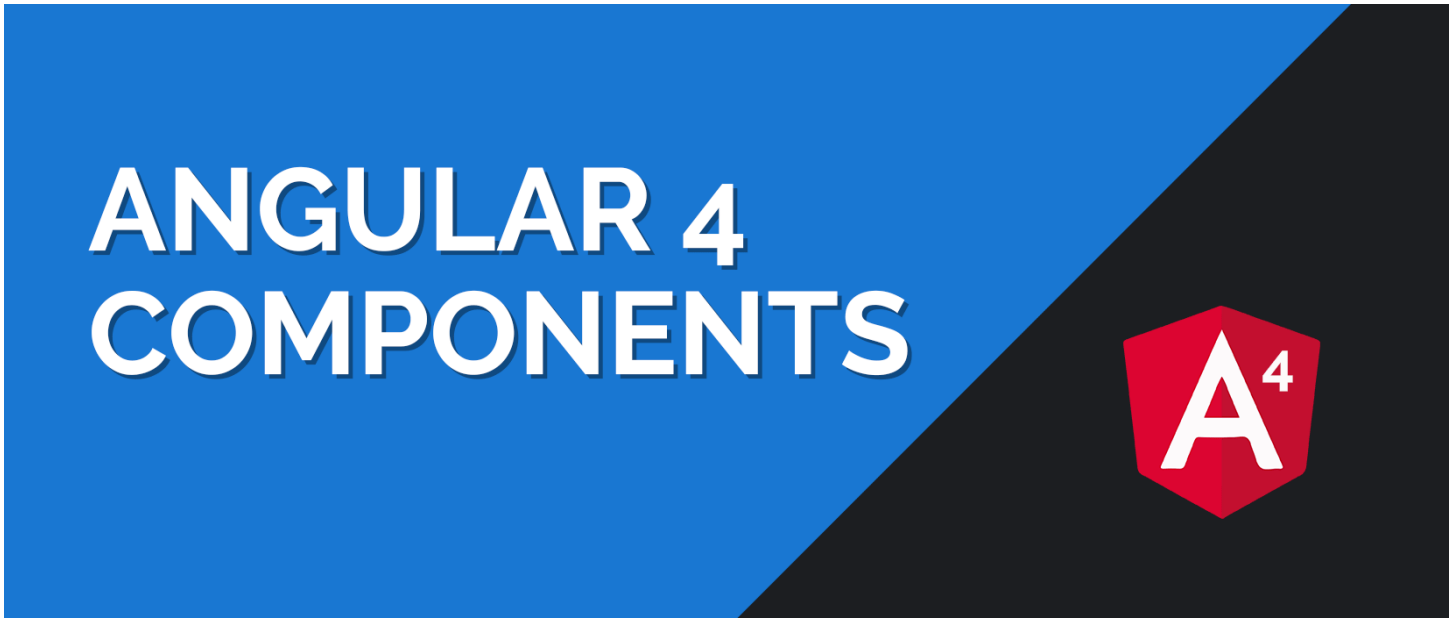
Angular 4 Components Tutorial



In Angular, components are where you will likely spend the bulk of time when developing your application. Angular 4 components are simply classes that are designated as a component with the help of a component decorator.

Every component has a defined template which can communicate with the code defined in the component class. So, let's take a close look at how components are structured.

Get a Project Setup to Follow Along!

Install the Angular CLI using **npm install -g @angular/cli** at the console. The stable release of 1.x will install an Angular 4 app by default.

Then start a new Angular 4 project by running **ng new components**. Then head over to the **/src/app/app.component.ts** file to follow along.

This is what the **app.component.ts** file will look like by default:

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

// Component Decorator
@Component({
  selector: 'app-root',
```

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

// Component Class
export class AppComponent {
  title = 'app works!';
}
```

Let's take a look at each of these 3 defining sections in detail.

1. Component Imports

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

The way you make a class a component is by importing the **Component** member from the @angular/core library.

Some components may have dozens of imports based on the needs of the component. This is also where you import any services that your component may use through dependency injection.

2. The Component Decorator

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

The next section is what's known as the Component Decorator. The **@Component** is using **Component** that was imported from the above import line.

This is what makes this given class a component.

Within the component, you have a variety of configuration properties that help define this given component.

- **selector**: This is the name of the tag that the component is applied to. For instance: **<app-root>Loading...</app-root>** within index.html.
- **templateUrl** & **styleUrls**: These define the HTML template and stylesheets associated with this component. You can also use **template** and **styles** properties to define inline HTML and CSS.

There are other properties you can define within the component decorator depending on the needs of your component. It is also where you define animations.

3. The Component Class

```
export class AppComponent {
  title = 'app works!';
}
```

Finally, we have the core of the component, which is the component class.

This is where the various properties and methods are defined. Any properties and methods defined here are accessible from the template. And likewise, events that occur in the template are accessible within the component.

It's also where dependency injection occurs within a constructor, which gives you access to various services.

Let's make our own component with the help of the CLI.

Creating a Component

Fortunately, the Angular CLI makes creating components a cinch. In the console within the project folder, simply run:

```
> ng g component my-new-component
```

- **ng** invokes the Angular CLI
- **g** is short for generate (note, you can use the full word generate if you wish)
- **component** is the type of element you're about to generate (others include directive, pipe, service, class, guard, interface, enum and module)
- and then the **component name**

So, what does generating a new component in Angular 4 actually do?

1. It creates a new folder inside of the **/app** folder based on the name you gave the component.
2. Inside, it generates 4 files: CSS, HTML, TS (component class) and a .spec.ts file for unit tests.
3. Inside **/src/app/app.module.ts** it imports the new component, and adds it to the declarations.

You could generate a component manually, as long as you do all of the legwork outlined in the above 3 points (though, only a component class is actually required in step 2).

In a Nutshell

So, components are the basic building blocks of your application. Every component has an associated template, stylesheet, and a class with logic.

Now that you understand components, let's take a closer look at templating in Angular 4.