**Angular Components: What are the Building Blocks of Angular?**

## **What is Angular?**

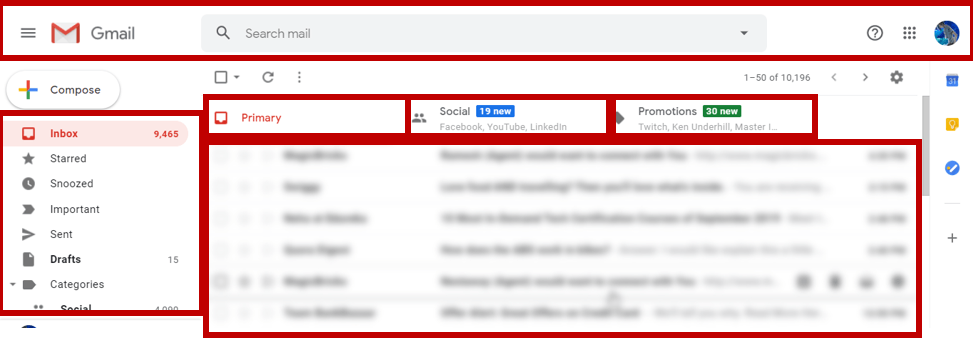


Angular is a front-end development framework. It was first released as AngularJS but was soon renamed to just ‘Angular’. Angular is a [TypeScript](https://www.edureka.co/blog/typescript-vs-javascript/#whatistypescript) based framework. It is used for creating single page web applications. If you don’t know what’s a single page web application, you could have a look at our, ‘[What is Angular?](https://www.edureka.co/blog/what-is-angular-getting-started-with-angular/#webapplicationvssinglepageapplication)‘ blog. Angular was developed by a group of developers at Google. After its public release, Google has been responsible for pushing out regular updates too. Angular is currently on its 8th version, which majorly issued changes regarding the HTTP module and how BAZEL and IVY are going to be integrated into Angular.

That’s enough about [Angular](https://www.edureka.co/blog/angular-tutorial/), let’s address the elephant in the room i.e. angular components.

## **What Is An Angular Component?**

Simply put, a component is the building block of any angular application.

  
  
Gmail is made using Angular, so it’s a great example to showcase components. Think about each section that I have marked out as a separate component. Each component is just its own bundle of [HTML](https://www.edureka.co/blog/what-is-html/), [*CSS*](https://www.edureka.co/blog/what-is-css/) and TypeScript. With the use of components, the entire page can be broken down into simpler pieces that can be coded separately and integrated later.

With the use of components, developers also get closer to the DRY protocol. A component once created, can be used as many times needed, hence reducing coding time, and also not writing repetitive code. Now that you have a general idea of angular components, let’s learn how to create them.

## **How to Create a Component?**

There are two ways to create a component –

* [**Using the CLI**](https://www.edureka.co/blog/angular-components/#CLI)
* [**Manually**](https://www.edureka.co/blog/angular-components/#manually)

### **Creating a Component Using The CLI**

Creating a component using the CLI is extremely easy and straightforward. This is the method you will be using most of the time. To create a component called server, you can use the command:

### [Angular Certification Training](https://www.edureka.co/angular-training" \t "_blank)

* *[Instructor-led Sessions](https://www.edureka.co/angular-training" \t "_blank)*
* *[Assignments](https://www.edureka.co/angular-training" \t "_blank)*
* *[Assessments](https://www.edureka.co/angular-training" \t "_blank)*
* *[Lifetime Access](https://www.edureka.co/angular-training" \t "_blank)*

[Explore Curriculum](https://www.edureka.co/angular-training" \t "_blank)

|  |  |
| --- | --- |
| 1 | ng generate component server |

The above command can be also written as:

|  |  |
| --- | --- |
| 1 | ng g c server |

You can also specify a path for your component to be created in:

|  |  |
| --- | --- |
| 1 | ng g c server-folder/server |

Any of the above commands will create a component called server. Since it has been created through the angular CLI, imports and declarations are all done automatically. The case is a little different if you wish to create your components manually for some reason. Let’s see how we can create these files manually.

### **Manually Creating a Component**

In this section, we are going to create the server component manually.

**Step 1:** Create a folder called server in your project under app/src.

**Step 2:** Create a new file in this folder called server.component.html. We are naming it in this fashion so that we are very clear about the file that we are creating.

**Step 3:** Fill in your [HTML file](https://www.edureka.co/blog/what-is-html/) with the appropriate code. For this example, let’s just paste the following:

### Front End Web Development Training

### [REACT JS CERTIFICATION TRAINING COURSE](https://www.edureka.co/reactjs-redux-certification-training" \o "React JS Certification Training Course" \t "_blank)

### [React JS Certification Training Course](https://www.edureka.co/reactjs-redux-certification-training" \o "React JS Certification Training Course" \t "_blank)

*[Reviews](https://www.edureka.co/reactjs-redux-certification-training" \o "React JS Certification Training Course" \t "_blank)*

**[5](https://www.edureka.co/reactjs-redux-certification-training" \o "React JS Certification Training Course" \t "_blank)**[(5690)](https://www.edureka.co/reactjs-redux-certification-training" \o "React JS Certification Training Course" \t "_blank)

### [WEB DEVELOPMENT CERTIFICATION TRAINING](https://www.edureka.co/complete-web-developer" \o "Web Development Certification Training " \t "_blank)

### [Web Development Certification Training](https://www.edureka.co/complete-web-developer" \o "Web Development Certification Training " \t "_blank)

*[Reviews](https://www.edureka.co/complete-web-developer" \o "Web Development Certification Training " \t "_blank)*

**[5](https://www.edureka.co/complete-web-developer" \o "Web Development Certification Training " \t "_blank)**[(7002)](https://www.edureka.co/complete-web-developer" \o "Web Development Certification Training " \t "_blank)

### [ANGULAR CERTIFICATION TRAINING](https://www.edureka.co/angular-training" \o "Angular Certification Training" \t "_blank)

### [Angular Certification Training](https://www.edureka.co/angular-training" \o "Angular Certification Training" \t "_blank)

*[Reviews](https://www.edureka.co/angular-training" \o "Angular Certification Training" \t "_blank)*

**[5](https://www.edureka.co/angular-training" \o "Angular Certification Training" \t "_blank)**[(16447)](https://www.edureka.co/angular-training" \o "Angular Certification Training" \t "_blank)

### [JAVASCRIPT AND JQUERY ESSENTIALS TRAINING AND CERTIFICATION](https://www.edureka.co/javascript-jquery-training" \o "JavaScript and JQuery Essentials Training and Certification" \t "_blank)

### [JavaScript and JQuery Essentials Training and Certification](https://www.edureka.co/javascript-jquery-training" \o "JavaScript and JQuery Essentials Training and Certification" \t "_blank)

*[Reviews](https://www.edureka.co/javascript-jquery-training" \o "JavaScript and JQuery Essentials Training and Certification" \t "_blank)*

**[5](https://www.edureka.co/javascript-jquery-training" \o "JavaScript and JQuery Essentials Training and Certification" \t "_blank)**[(2737)](https://www.edureka.co/javascript-jquery-training" \o "JavaScript and JQuery Essentials Training and Certification" \t "_blank)

### [JQUERY UI DEVELOPMENT](https://www.edureka.co/jquery-ui-development" \o "jQuery UI Development" \t "_blank)

### [jQuery UI Development](https://www.edureka.co/jquery-ui-development" \o "jQuery UI Development" \t "_blank)

*[Reviews](https://www.edureka.co/jquery-ui-development" \o "jQuery UI Development" \t "_blank)*

**[5](https://www.edureka.co/jquery-ui-development" \o "jQuery UI Development" \t "_blank)**[(1024)](https://www.edureka.co/jquery-ui-development" \o "jQuery UI Development" \t "_blank)

Next

|  |  |
| --- | --- |
| 1 | <h3>You are currently viewing the server component</h3> |

**Step 4:** Now create a [TypeScript file](https://www.edureka.co/blog/typescript-vs-javascript/#howtousetypescript). Right-click on the folder and create a new file. Name it – server.component.ts.

**Step 5:** Create the CSS file for your component. Name it – server.component.css

**Step 6:** In your TypeScript file you will be defining the business logic of your component. Other than that, the component’s metadata is also defined here.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | import { Component } from '@angular/core';    @Component ({    selector: 'app-server',    templateUrl: './server.component.html',    styleUrl: [ './server.component.css' ]  })    export class ServerComponent {} |

In the first line of code, we import components from the core angular library. Then we define the metadata of the component using the component decorator. In the metadata, we first define the selector, which out here is app-server. To use this component we have to use the <app-server></app-server> tag in our HTML file. The templateUrl tells Angular where to look for the content of app-server and similarly, the stylesUrl tells us about the styling of that particular component.

Lastly, we are exporting the class ServerComponent, so that we can use the component in other parts of our project. In the class, you can define the logic that you would deem appropriate for your component.  
  
**Step 7:** We are done creating the component, but we still need to inform TypeScript and Angular about this new component that we have created. Head over to the app module file. The name of the file should be app.module.ts. In the file you will find a declarations array. Add ServerComponent to the declaration. After that import the component.

|  |  |
| --- | --- |
| 1 | import { ServerComponent } from './server.component' |

## **Constituents of a Component**

A component has 3 main files.

* **HTML** – In your HTML file, you define the view structure of your component.
* **CSS** – In your CSS file, you define the styling pertaining to your component.
* **TypeScript** – In your TypeScript file, you define the metadata and business logic for your component.

These three files when working hand in hand, make up an angular component.