**Overview:** AngularJS is an open source framework built over JavaScript. It was built by the developers at Google. This framework was used to overcome obstacles encountered while working with Single Page applications. Also, testing was considered as a key aspect while building the framework. It was ensured that the framework could be easily tested.

Compiler: Traceur Compiler

**AngularJs And Angular Are Different:**

<https://www.sitepoint.com/angularjs-vs-angular/>

**Features of Angular:**

<https://dzone.com/articles/angular-2-vs-angular-4-features-performance>

<https://angular.io/features>

>>Can we have multiple Constructor in angular component?

**Advantages of angular:**

<https://medium.com/@angularminds/comparison-between-angular-1-vs-angular-2-vs-angular-4-62fe79c379e3>

<https://www.quora.com/What-are-the-advantages-of-angular2-over-angular1>

# **Prerequisites:**

The user should be familiar with the basics of web development and JavaScript. Since the Angular framework is built on the JavaScript framework, it becomes easier for the user to understand Angular if they know JavaScript.

**Angular Environment setup:**

To start working with Angular 2, you need to get the following key components installed.

* **Npm** − This is known as the node package manager that is used to work with the open source repositories. Angular JS as a framework has dependencies on other components. And **npm** can be used to download these dependencies and attach them to your project.
* **Git** − This is the source code software that can be used to get the sample application from the **github** angular site.
* **Editor** − There are many editors that can be used for Angular JS development such as Visual Studio code and WebStorm. In our tutorial, we will use Visual Studio code which comes free of cost from Microsoft.

## npm Installation

Let’s now look at the steps to get npm installed. The official site for npm is <https://www.npmjs.com/>

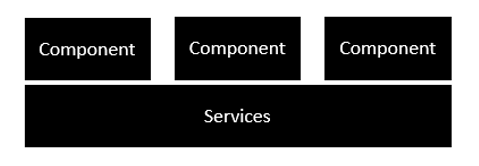
<https://www.tutorialspoint.com/angular2/angular2_environment.htm>

**Hello world first programme in angular:**

Create a hello world programme by creating a new project from angular.cli and run this programe.

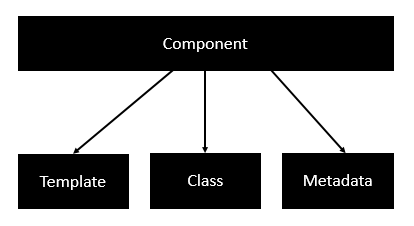
**About Angular Architecture:**

he following screenshot shows the anatomy of an Angular 2 application. Each application consists of Components. Each component is a logical boundary of functionality for the application. You need to have layered services, which are used to share the functionality across components.



Following is the anatomy of a Component. A component consists of −

* **Class** − This is like a C++ or Java class which consists of properties and methods.
* **Metadata** − This is used to decorate the class and extend the functionality of the class.
* **Template** − This is used to define the HTML view which is displayed in the application.



Following is an example of a component.

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

@Component ({

selector: 'my-app',

templateUrl: 'app/app.component.html'

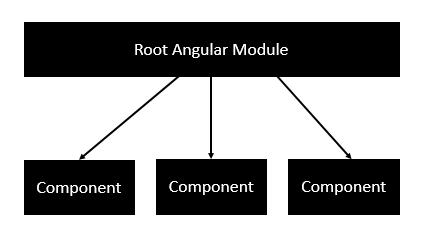
})

export class AppComponent {

appTitle: string = 'Welcome';

}

Each application is made up of modules. Each Angular 2 application needs to have one Angular Root Module. Each Angular Root module can then have multiple components to separate the functionality.



Following is an example of a root module.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule ({

imports: [ BrowserModule ],

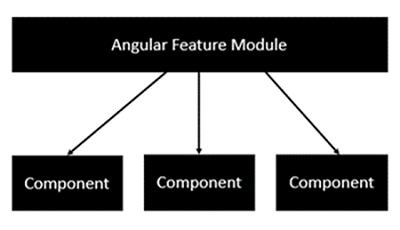
declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

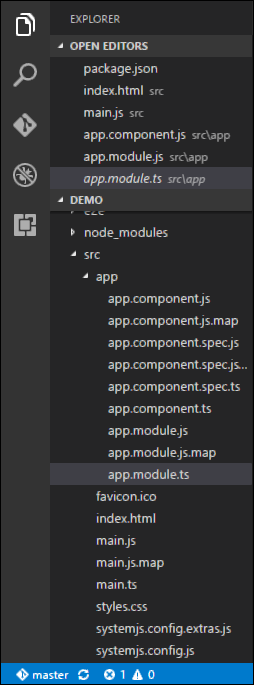
Each application is made up of feature modules where each module has a separate feature of the application. Each Angular feature module can then have multiple components to separate the functionality.



**What is modules in angular:**

Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application. Let’s inspect the code which gets added to the demo application.

In Visual Studio code, go to the app.module.ts folder in your app folder. This is known as the root module class.



The following code will be present in the **app.module.ts** file.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule ({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

Let’s go through each line of the code in detail.

* The import statement is used to import functionality from the existing modules. Thus, the first 3 statements are used to import the NgModule, BrowserModule and AppComponent modules into this module.
* The NgModule decorator is used to later on define the imports, declarations, and bootstrapping options.
* The BrowserModule is required by default for any web based angular application.
* The bootstrap option tells Angular which Component to bootstrap in the application.

A module is made up of the following parts −

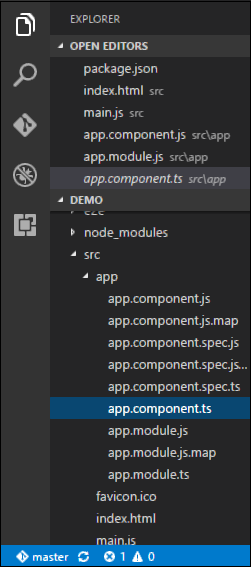
* **Bootstrap array** − This is used to tell Angular JS which components need to be loaded so that its functionality can be accessed in the application. Once you include the component in the bootstrap array, you need to declare them so that they can be used across other components in the Angular JS application.
* **Export array** − This is used to export components, directives, and pipes which can then be used in other modules.
* **Import array** − Just like the export array, the import array can be used to import the functionality from other Angular JS modules.

# **Components:**

Components are a logical piece of code for Angular JS application. A Component consists of the following −

* **Template** − This is used to render the view for the application. This contains the HTML that needs to be rendered in the application. This part also includes the binding and directives.
* **Class** − This is like a class defined in any language such as C. This contains properties and methods. This has the code which is used to support the view. It is defined in TypeScript.
* **Metadata** − This has the extra data defined for the Angular class. It is defined with a decorator.

Let’s now go to the app.component.ts file and create our first Angular component.



Let’s add the following code to the file and look at each aspect in detail.

## Class

The class decorator. The class is defined in TypeScript. The class normally has the following syntax in TypeScript.

### **Syntax**

class classname {

Propertyname: PropertyType = Value

}

### **Parameters**

* **Classname** − This is the name to be given to the class.
* **Propertyname** − This is the name to be given to the property.
* **PropertyType** − Since TypeScript is strongly typed, you need to give a type to the property.
* **Value** − This is the value to be given to the property.

### **Example**

export class AppComponent {

appTitle: string = 'Welcome';

}

In the example, the following things need to be noted −

* We are defining a class called AppComponent.
* The export keyword is used so that the component can be used in other modules in the Angular JS application.
* appTitle is the name of the property.
* The property is given the type of string.
* The property is given a value of ‘Welcome’.

## Template

This is the view which needs to be rendered in the application.

### **Syntax**

Template: '

<HTML code>

class properties

'

### **Parameters**

* **HTML Code** − This is the HTML code which needs to be rendered in the application.
* **Class properties** − These are the properties of the class which can be referenced in the template.

### **Example**

template: '

<div>

<h1>{{appTitle}}</h1>

<div>To Tutorials Point</div>

</div>

'

In the example, the following things need to be noted −

* We are defining the HTML code which will be rendered in our application
* We are also referencing the appTitle property from our class.

## Metadata

This is used to decorate Angular JS class with additional information.

Let’s take a look at the completed code with our class, template, and metadata.

### **Example**

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

@Component ({

selector: 'my-app',

template: ` <div>

<h1>{{appTitle}}</h1>

<div>To Tutorials Point</div>

</div> `,

})

export class AppComponent {

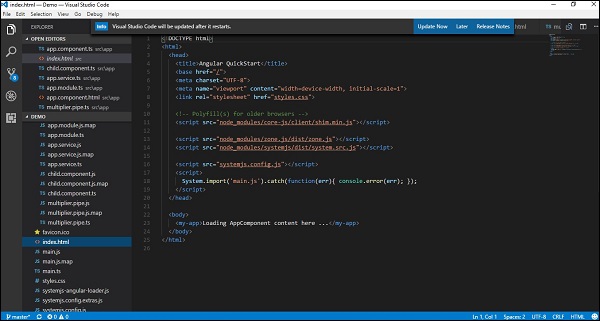
appTitle: string = 'Welcome';

}

In the above example, the following things need to be noted −

* We are using the import keyword to import the ‘Component’ decorator from the angular/core module.
* We are then using the decorator to define a component.
* The component has a selector called ‘my-app’. This is nothing but our custom html tag which can be used in our main html page.

Now, let’s go to our index.html file in our code.



Let’s make sure that the body tag now contains a reference to our custom tag in the component. Thus in the above case, we need to make sure that the body tag contains the following code −

<body>

<my-app></my-app>

</body>

### **Output**

Now if we go to the browser and see the output, we will see that the output is rendered as it is in the component.

