**Angular 2**

**What is typescript**

* **Typescript is the super set of JavaScript.**

**What is SPA**

* **Usually, when we click button or link in website to refresh small part of page entire web page is reloaded**
* **But in Single Page Application only the portion of web page refresh and reloaded**
* **So this will improve the speed and performance of the application.**

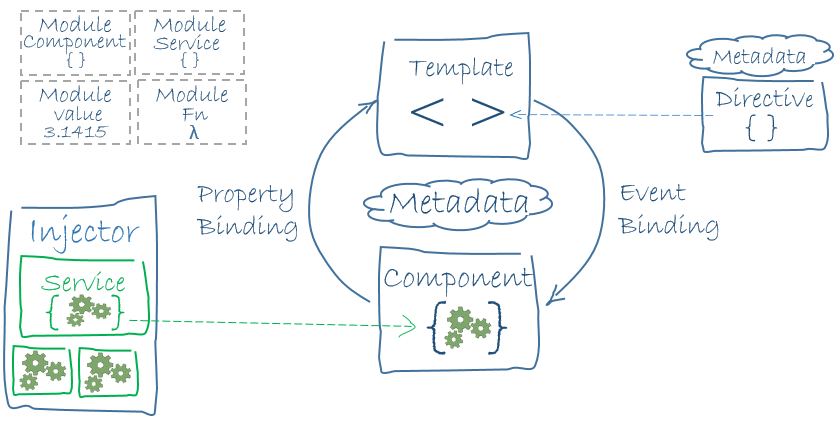
**What is Angular:**

* **The Angular is a framework to development platform for building a Single Page Client-Side Application for mobile and desktop.**
* **It uses Typescript & HTML to build Apps.**
* **It comes with features like modules,** [**component**](https://www.tektutorialshub.com/angular/angular-component/)**,**[**Directives**](https://www.tektutorialshub.com/angular/angular-directives/)**,**[**Forms**](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/)**,**[**Pipes**](https://www.tektutorialshub.com/angular/angular-pipes/)**,**[**HTTP Services**](https://www.tektutorialshub.com/angular/angular-httpclient/)**,**[**Dependency Injection**](https://www.tektutorialshub.com/angular/angular-dependency-injection/)**, etc**

**Why Angular**

* **Angular used modular approach, hence, application will be in clear structure**
* **Can able to write Re usable code, maintainable, testable code**

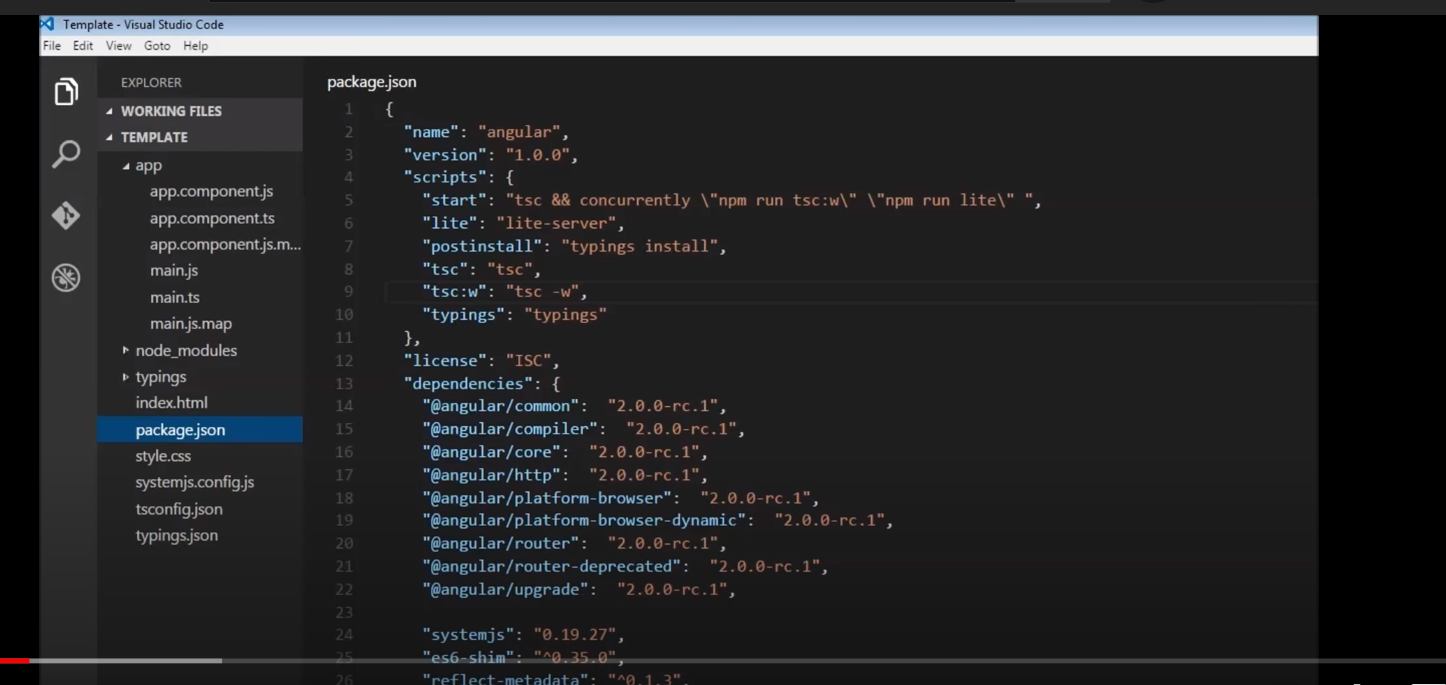
**Angular Architecture:**

****

**The building blocks of an Angular application:**

1. [**Modules**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#modules)
2. [**Components**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#components)
3. [**Templates**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#templates)
4. [**Metadata**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#metadata)
5. [**Data binding**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#data-binding)
6. [**Services**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#services)
7. **Routers**
8. [**Directives**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#directives)
9. [**Dependency injection**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#dependency-injection)

**Angular Project Structure:**

****

**Package.json**

* **package.json file locates in project root and contains information about your web application.**
* **The main purpose of the file comes from its name package, so it'll contain the information about npm packages installed for the project.**
* **It has Project Metadata which contains information about your application.**
* **Also we can add the fields like project description, homepage,repository, author and scripts.**
* **Dependencies - The list of packages installed as dependencies for this project are required at runtime.**
* **Development Dependencies-The list of packages that are required only for development. This packages are installed only on developer's machine and will not be run for production build.**
* **The symbol in front of version says to npm install which package version to use**

**Eg:  
1.0.0 means strictly 1.0.0 version of the package  
~1.0.0 means, 1.0.0 version or it's later patch version (third number may vary): 1.0, 1.0.x  
^1.0.0 means, 1.0.0 version or it's later minor version (second number may vary): 1, 1.x**

**Tsconfig.js –**

**Index.js-**

**Main.ts/main.js – it is bootstrap our app module which is root module**

**Default port for the application is 3000**

1. **Modules:**

* **It is main building block of Angular application**
* **The Angular Modules help us to organize our code into manageable parts or block.**
* **Each block implements a specific feature.**
* **Every Angular app has a root module, conventionally named AppModule, which provides the bootstrap mechanism that launches the application.**
* **An app typically contains many functional modules.**
* **If we want to use another custom Angular module, then we need to register that module inside the  app.module.ts file import section .**

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

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

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

* **import { HomeModule } from './component/home/home.module';**

**@NgModule({**

**declarations: [**

**AppComponent**

**],**

**imports: [**

**BrowserModule,**

**HomeModule**

**],**

**Providers: [],**

**bootstrap: [AppComponent]**

**})**

**export class AppModule { }**

1. **Component**

* **The Component is the main building block of an Angular Application.**
* **Every Angular project has at least one component, the *root component and* root component connects the component hierarchy with a page document object model (DOM).**
* **A Component contains the definition of the View(HTML template) and the data that defines how the View looks and behaves.**
* **A *component* controls a patch of screen called a *view*.**
* **The @Component decorator identifies the class immediately below it as the component and provides the template and related component-specific metadata.**

**3.** [**Templates**](https://v2.angular.io/docs/ts/latest/guide/architecture.html#templates)

* **The angular template combines the HTML with Angular markup that can modify HTML elements before they are displayed.**
* **Template directives provide program logic, and binding markup connects your application data and the DOM. There are two types of data binding.**
* **Event binding lets your app respond to user input in the target environment by updating your application data.**
* **Property binding lets you interpolate values that are computed from your application data into the HTML.**

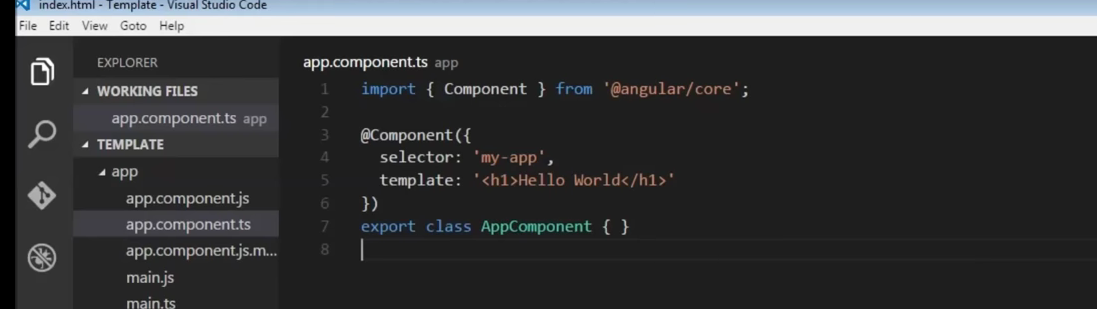
**<div style="text-align:center">  
 <h1>  
 {{2 | power: 5}}  
 </h1>  
</div>**

**4.Metadata**

* **Metadata tells Angular how to process a class.**
* **It is used to decorate the class so that it can configure the expected behaviour of a class.**
* **Metadata can be attached to the TypeScript using the decorator.**

**Here are a few of the most useful @Component configuration options:**

* **selector: CSS selector that tells Angular to create and insert an instance of this component where it finds a <app-root>tag which is Parent Component.**
* **templateUrl: module-relative address of this component's HTML template.**
* **providers: array of dependency injection providers for services that the component requires.**
* **@Injectable, @Input, and @Output are a few of the more popular decorators.**
* **The template, metadata, and component together describe a view.**

****

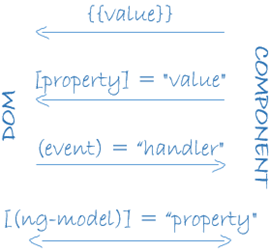
1. **Data Binding**

**Reg:** [**https://www.tektutorialshub.com/angular/angular-data-binding/**](https://www.tektutorialshub.com/angular/angular-data-binding/)

* **Data binding is a technique, where the data stays in sync between the component and the view. Whenever the user updates the data in the view, Angular updates the component. When the component gets new data, the Angular updates the view.**
* **The data binding in Angular can be broadly classified into two groups. One way binding or two-way binding**

1. **One way binding**

* **In one way binding data flows from one direction. Either from view to component or from component to view.**

****

1. **From the Component to the view**

* **To bind data from component to view, we make use of Interpolation & Property Binding.**
* **Interpolation:** [**Interpolation**](https://www.tektutorialshub.com/angular/interpolation-in-angular/)**allows us to include expressions as part of any string literal, which we use in our HTML. The angular evaluates the expressions into a string and replaces it in the original string and updates the view. You can use interpolation wherever you use a string literal in the view**
* **The Angular uses the {{ }} (double curly braces) in the template to denote the interpolation. The syntax is {{ templateExpression }}**
* **Property binding: [property]=”value”- The**[**Property binding**](https://www.tektutorialshub.com/angular/property-binding-in-angular/)**allows us to bind HTML element property to a property in the component. Whenever the value of the component changes, the Angular updates the element property in the View. <input type="text" [value]="student.name" />  
  <input type="text" [value]="student.college" />**

**Difference between Interpolation and property binding**

1. **From the view to the Component**

* **Event binding: (Event)=”myFunction($event)”- Event binding allows us to bind events such as keystrokes, clicks, hover, touch, etc to a method in component. It is one way from view to component. By tracking the user events in the view and responding to it, we can keep our component in sync with the view.**
* **For Example, when the user changes input in a text box, we can update the model in the component, run some validations, etc. When the user submits the button, we can then save the model to the backend server.**
* **Angular uses the following syntax for event binding**

**<button (click)="onSave()">Save</button>**

**2. Two-Way binding: [(ngModel)]=”property”**

* **Two-way binding means that changes made to our model in the component are propagated to the view and that any changes made in the view are immediately updated in the underlying component**
* **Two-way binding is useful in data entry forms. Whenever a user makes changes to a form field, we would like to update our model. Similarly, when we update the model with new data, we would like to update the view as well**
* **The two-way binding uses the special syntax known as  [()]**
* **<someElement [(someProperty)]="value"></someElement>.**
* **The above syntax sets up both property binding & event binding. But to make use of it, the property must have the change event with the name <propertyName>Change**
* **But, angular has a special directive ngModel, which sets up the two-way binding**

**NgModel**

* **The Angular uses the ngModel directive to achieve the two-way binding on HTML Form elements. It binds to a form element like input, select, selectarea. etc.**
* **The ngModel directive is not part of the Angular Core library. It is part of the @angular/forms. You need to import the FormsModule package into your Angular module.**

|  |  |
| --- | --- |
| **1**  **2**  **3** | **import { FormsModule } from '@angular/forms';** |

* **<input type="text" name="value" [(ngModel)]="value">**
* **When you bind to a ngModel directive, behind the scene it sets up property binding & event binding. It binds to the value property of the element using property binding. It then uses the ngModelChange event to sets up the event binding to listen to the changes to the value.**

1. **Directives**

* **The**[**Angular directive**](https://www.tektutorialshub.com/angular/angular-directives/)**helps us to manipulate the DOM.**
* **You can change the appearance, behaviour, or layout of a DOM element using the directives. They help you to extend HTML.**
* **The**[**Angular directives**](https://www.tektutorialshub.com/angular/angular-directives/)**are classified into three categories based on how they behave.  They are Component, Structural and Attribute Directives**

1. **Component Directive**

* **Components are special directives in Angular. They are the directive with a template (view).**
* **It is used to change the layout of DOM.**

1. **Structural Directive**

* **Structural directives can change the DOM layout by adding and removing DOM elements.**
* **All structural Directives are preceded by Asterix(\*) symbol**
* **Below are commonly used directives.**
* **The**[**ngFor**](https://www.tektutorialshub.com/angular/angular-ngfor-directive/)**is an Angular structural directive, which repeats a portion of the HTML template once per each item from an iterable list (Collection).**

**<tr \*ngFor="let customer of customers;">**

**<td>{{customer.customerNo}}</td>**

**<td>{{customer.name}}</td>**

**<td>{{customer.address}}</td>**

**<td>{{customer.city}}</td>**

**<td>{{customer.state}}</td>**

**</tr>**

**﻿**

* **The**[**ngSwitch**](https://www.tektutorialshub.com/angular/angular-ngswitch-directive/)**directive lets you add/remove HTML elements depending on a match expression.**
* [**ngSwitch**](https://www.tektutorialshub.com/angular/angular-ngswitch-directive/)**directive used along with**[**ngSwitchCase**](https://www.tektutorialshub.com/angular/angular-ngswitch-directive/#ngswitchcase)**and**[**ngSwitchDefault**](https://www.tektutorialshub.com/angular/angular-ngswitch-directive/#ngswitchdefault)

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9** | **<div [ngSwitch]="Switch\_Expression">**  **<div \*ngSwitchCase="MatchExpression1”> First Template</div>**  **<div \*ngSwitchCase="MatchExpression2">Second template</div>**  **<div \*ngSwitchCase="MatchExpression3">Third Template</div>**  **<div \*ngSwitchCase="MatchExpression4">Third Template</div>**  **<div \*ngSwitchDefault?>Default Template</div>**  **</div>** |

* **The**[**ngIf**](https://www.tektutorialshub.com/angular/angular-ngif-directive/)**Directives is used to add or remove HTML elements based on an expression. The expression must return a boolean value. If the expression is false then the element is removed, else the element is inserted**

**<div \*ngIf="condition">**

**This is shown if condition is true**

**</div>**

1. **Attribute Directives**

* **An Attribute or style directive can change the appearance or behavior of an element.**
* **The ngModel directive is used the achieve the**[**two-way data binding**](https://www.tektutorialshub.com/angular/angular-data-binding/)**.**
* **The**[**ngClass**](https://www.tektutorialshub.com/angular/angular-ngclass-directive/)**Directive is an Angular Attribute Directive, which allows us to add or remove CSS classes to an HTML element.**
* **The**[**ngStyle**](https://www.tektutorialshub.com/angular/angular-ngstyle-directive/)**directive allows you to modify the style of an HTML element using the expression.  Using the**[**ngStyle**](https://www.tektutorialshub.com/angular/angular-ngstyle-directive/)**you can dynamically change the style of your HTML element.**

**Difference b/w ngClass and ngStyle**

1. **Jjjjkj**
2. **Njjj**

**1. angular objecrvables /subcribe**

**2. how to send data between components / communicate between components**

**3. what is main components in angular / building block of angular**

**4.lifecycle in angular**

**5. what is dependency in angular**

**6. how to catch exception in angular**

**7. types of compilation in Angular or JIT vs AOT**

**program:**

**{​​​8,0,6,3,4,0,5,0,8,0,1}​​​**

**1. move all 0 to end**

**2. find duplicate integer in list using stram API –**

Set<Integer> set= Stream.*of*(5,5, 13, 4,

21, 13, 27,

2, 59, 59, 34).collect(Collectors.*toSet*());