Angular

How Angular project is executed?

When we run an Angular project, the main.ts file gets executed. From main.ts the Anguler comes to
know about the AppModule. From here, it goes to app.module.ts file. There, it comes to know about
AppComponent. SO it goes to app.component.ts file. From here, it comes to know about the app-root
which is in the selector property of the component and also know about the view from the templateUrl
property. After this the index.html file will be loaded in the browser where the
in index.html will be replaced by the content in the template provided by app component.

When we start the application by hitting the ng-serve command, it rebuilds the project then it will
create javascript bundles automatically to the index.html file. So, the script files are injected into the
index.html by the angular cli.

Component

Component is a TypeScript class decorated with @Component decorator and it contains methods &
properties which can use in html. The AppComponent (Root Component)is the main component in any
Angular application. The root component can have several nested components and at the end we will
have a complete application consisting of different components.

We can have components for header, navbar, main body, sidebar, footer,etc. We can also have
components for home, about, contact present in the navbar.
@Component({

selector: 'app-component-overview',

templateUrl: './component-overview.component.html',

styleUrls: ['./component-overview.component.css'] })

```
export class AppComponet{
```

```
title='AngularApp';
```

```
message = 'Angular is the app of apps';
```

```
}
```

So a component has :

A Class, that contains the code required for the view template i.e. contains the UI logic

A View template, defines the user interface. It contains the Html, directives and data binding.

A decorator, it adds meta data to the class, making it a component.

In order to use a component, we need to export it, and register it in app module.

We can use the value in selector property as an html tag.

Template property of Component:

With template property, we can write html in the component file. It won't need the html file.
```
@Component({
```

```
selector: 'app-component-overview',
```

```
template: '<div><p>This web takes cookies</p></div>',
```

```
styleUrls: ['./component-overview.component.css'] })
```

```
export class AppComponet{
```

```
title='AngularApp';
```

```
message = 'Angular is the app of apps';
```

```
}
```

In templateUrl property, we specify the path to a html file to use as a view file. While for template
property we specify the html inside a string. For multiline html code using template property,
user``(tactiks). Its good practice to write the html code in separate file, if we have 3 or more lines
of html

code.

Disadvantage: We know about errors in html at runtime only. We are mixing typescript code with html
code.

Styles property of Component:

With style property, we can add css style in the component file. No need for css file.
@Component({

selector: 'app-component-overview',

template: '<div><p>This web takes cookies</p></div>',

styles: [“div{margin: 10px 0px; padding: 10px 20px;}”,

”p{font-size:40px}”]})

export class AppComponet{

title=’AngularApp’;

message = ‘Angular is the app of apps’;

}

It is similar to template property and has similar disadvantages.

Selectors in Angular:

There are different selector in Angular:

1. We can use a selector like an HTML tag.
   We do so by using this syntax.
   Selector : ‘app-nav’,
   Example:
2. We can use a selector like a HTML attribute.
   We do so by using this syntax: Write the value between []
   Selector : ‘[app-nav]’,
   Example: <div app-nav></div>
3. We can use a selector like a CSS class.
   We can do so by using this syntax: Add a dot(.) before the value.
   Selector: ‘.app-nav’,

Example : <div class="app-nav"></div>

Data binding:

It allows us to communicate between a component class and its corresponding view template or vice
versa.

One-way data binding:

It is when we can access a component class property in its corresponding view template Or when we can
access a value form view template in corresponding component class property. I.E only in one direction.

Two-way data binding:

Binds data from component to view template and view template to component class. This is a combination of property binding and event binding i.e. change in component will be seen in view and
change in view will be seen in component. In Both Directions.

Property binding is done by [].

Event binding is done by ().

So for two way binding, [()]. This is banana box syntax. Then we use a directive called ngModel and
assign this property. To use ngModel you need to import FormModules.

<input type="text" [(ngModel)] = "searchValue">

searchValue is a property in the component/

String Interpolation:

Used to bind data from component class to view template. I.E. data flows from component to view. It is
used to achieve one way data binding. Using string interpolation we can use property or methods of
component class.

We use {{}} to perform string interpolation in html file. Inside {{}}, we can write any typescript code. We

can also call typescript method using string interpolation. Example: <h1> {{ getName() }} </h2>

We use Angular to dynamically render data in the webpage. When we write static content in html,
angular isn't need. Also, if the same static data is used in multiple parts, then it is prone to have issue
like not rendering consistent data. To solve this, we can create a property in th component where we
can define the data we want to render.

Inside component:

export class HeaderComponent{

    slogan: string = "Hello world."

}

In Html:

<h1>{{slogan}}</h1>

We using slogan property in the component to render the value in the view dynamically using data
binding.

Property binding:

With property binding, we are binding the properties of html elements with the property or method ofc
component class.

In component class,

export class HeaderComponent{

    source : string = "/assets/shopping.jpg';

}

In Html,