**if condition** : if the condition is true the output will be shown, if the condition is false nothing will be display .

**if else** : if the if condition was true the if condition will be run,if condition is false the else condition will be run .

**else if** : if the else was true there will be a inside else there will be a if statement

**for** : it will loop the operation till the condition will result of the given operation.

**while** : when the condition is true the Operation will be run .

**do while** : (first do statement will be on checking , if its true while condition will be run and its execute the result .

**ARRAY** : square braces [] array is a collection of elements.

**string**: " " inside the double codes(sequences of characters)

**bolean**: true or false

LOOP: there are three types of loop

For loop (the condition was true the function will be executed)

While loop ( if the condition was true it will be executed and the if its false it will not executed )

Do while (loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

**What is Data Binding?**

Data binding is one of the finest and useful features of the Angular framework. Due to this feature, the developer needs to write less code compared to any other client-side library or framework. Data binding in an Angular application is the automatic synchronization of data between the model and view components.

**Different Types of Data Binding**

In Angular 8, there are four different types of Data binding processes available. They are:

**1. Interpolation**

Interpolation data binding is the most popular and easiest way of data binding in Angular 8. This feature is also available in previous Angular framework versions. Actually, the context between the braces is the template expression that Angular first evaluates and then convert into strings. Interpolation uses the braces expression i.e. {{ }} to render the bound value to the component template. It can be a static string, numeric value, or an object of your data model. In Angular, we use it like this: {{firstName}}.

The below example shows how we can use the interpolation in the component to display data in the front end.

<div>

<span>User Name : {{userName}}</span>

</div>

**2. Property Binding**

In Angular 8, another binding mechanism exists, which is called Property Binding. In nature, it is just the same as interpolation. Some people also called this process as one-way binding like the previous AngularJS concept. Property binding used [] to send the data from the component to the HTML template. The most common way to use property binding is to assign any property of the HTML element tag into the [] with the component property value, i.e:

<input type=”text” [value]=”data.name”/>

To implement the property binding, we will just make the below changes in the previous HTML file from the interpolation sample i.e. interpolation.component.html

<div>

<input [value]="value1" />

<br /><br />

</div>

**3. Event Binding**

Event binding is another of the data binding techniques available in Angular. This data binding technique does not work with the value of the UI elements—it works with the event activities of the UI elements like click-event, blur-event, etc. In the previous version of AngularJS, we always used different types of directives like ng-click, ng-blur to bind any particular event action of an HTML control. But in the current Angular version, we need to use the same property of the HTML element (like click, change, etc.) and use it within parentheses. In Angular 8, for properties, we use square brackets, and in events, we use parentheses.

<div>

<input type="submit" value="Submit" (click)="fnSubmit()">

</div>

**4. Two-Way Binding**

In Angular Framework, the most used and important data binding techniques are known as Two-Way Data Binding. Two-way binding is mainly used in the input type field or any form element where the user can provide input values from the browser or provides any value or changes any control value through the browser. On the other side, the same is automatically updated into the component variables and vice versa. Similarly, in Angular 8 we have a directive called ngModel, and it needs to be used as below:

<input type=”text” [(ngModel)] =”firstName”/>

We use [] since it is actually a property binding, and parentheses are used for the event binding concept i.e. the notation of two-way data binding is [()].

INPUT

**@Input() Decorator**

Input decorator marks the property as the input property. I.e it can receive data from the parent component. The parent component uses the property binding to bind it to a component property. Whenever the value in the parent component changes angular updates the value in the child component.

OUTPUT

**@Output() Decorator**

EventEmitter is responsible for raising the event. The @output property normally is of type EventEmitter. The child component will use the emit() method to emit an event along with the data.

Output decorates the property as the output property. We initialize it as an EventEmitter. The child component raises the event and passes the data as the argument to the event. The parent component listens to events using event binding and reads the data.

**What is the Directive?**

A directive modifies the DOM by changing the appearance, behavior, or layout of DOM elements. Directives just like Component are one of the core building blocks in the Angular framework to build applications. In fact, Angular 8 components are, in large part, directives with templates.

**Types of Directives**

There are three main types of directives in Angular 8:

**Attribute Directives** – Directives that change the behavior of a component or element but don’t affect the template.

**Structural Directives** – Directives that change the behavior of a component or element by affecting the template or the DOM decoration of the UI.

**Types of Pipe**

In Angular 8, we can categories the pipes in two types i.e. Pure Pipes and Impure Pipes.

**Pure Pipes**:- Pure pipes in angular are those pipes which always accepts some arguments as input value and return some value as the output according to the input values. Some examples of the pure pipes are – decimal pipes, date pipes, etc. When we use these types of pipes in Angular, we provide input value with related configuration value to the pipes and pipes return us the formatted value as an output.

**Impure Pipes**:- Impure pipes in angular are those pipes which also accepts the input values, but return the different types of the value set according to the state of the input value. An example of the impure pipes is async pipes. These pipes always store the internal state and return different types of value as the output according to the internal state and logic.

**Types of Angular Forms**

1. Template Form
2. Reactive Form

|  |  |
| --- | --- |
| Template-Driven Form | Reactive Form |
| Template-Driven Form is less explicit, and it is mainly created by Directives. | Reactive Form is more explicit and normally created within the Component class. |
| It supports the unstructured data model | It always supports the structured data model. |
| It uses directives for implementing Form validations | It uses the function for implementing Form Validations |
| When form control value changes, it provides an asynchronous mechanism to update form controls. | When form control value changes, it provides synchronous mechanism to update form controls. |

FORMS VALIDATION:

* Required name,
* red box,
* patten for date,
* min length,
* max length

LOCAL STORAGE

SESSION STORAGE

QUERY PARAMS

SERVICE

* In service typescript page we can create anything and we can call in any component of the folder.

There are there types of servise

* *Injector*
* *Provider*
* *Dependency*
* Injector  
  The main purpose of the using Injector section is the expose an object or APIs which basically helps us to create instances of the dependent class or services.

* Provider  
  A Provider is basically acting as an Instructor or Commander. It basically provides instruction to the injector about the process of creating instances of the dependent objects. The provider always taken the token value as input and then map that token value with the newly created instances of the class objects.

* Dependency  
  Dependency is the processor type that basically identifies the nature of the created objects.

NGIF = if the condition is true it will be executed .

Ngclass = if the condition is true the class will be activated(color, style)

Ngstyle = if the condition is true I will executed the style we hae mentioned .

Ngmodule = if we edit in display page it will be also be edited in ts page .

Interview

Html tags :

* Div, P,h,video,img,style,a,span,
* Inline css : html style
* Internel : head
* External : class css