ANGULAR-17

A.NAGARAJU

What is Angular ?

- Angular is a popular open-source web application framework developed by Google and maintained by a community of developers and organizations.
- It is used for building dynamic single-page web applications (SPAs) and is based on TypeScript, a superset of JavaScript.
- Angular provides a comprehensive set of tools and features for developing robust and scalable front-end applications

Features of Angular

- Component-based architecture.
- Two-way data binding.
- Directives.
- Services.
- Routing.
- Forms
- HTTP client.

Component-based architecture

- Angular applications are built using components, which are reusable and encapsulated building blocks. Each component represents a part of the user interface and its behavior
- components are typically associated with user interface elements such as buttons, forms, navigation bars, or entire sections of a webpage.

Property and Event Binding

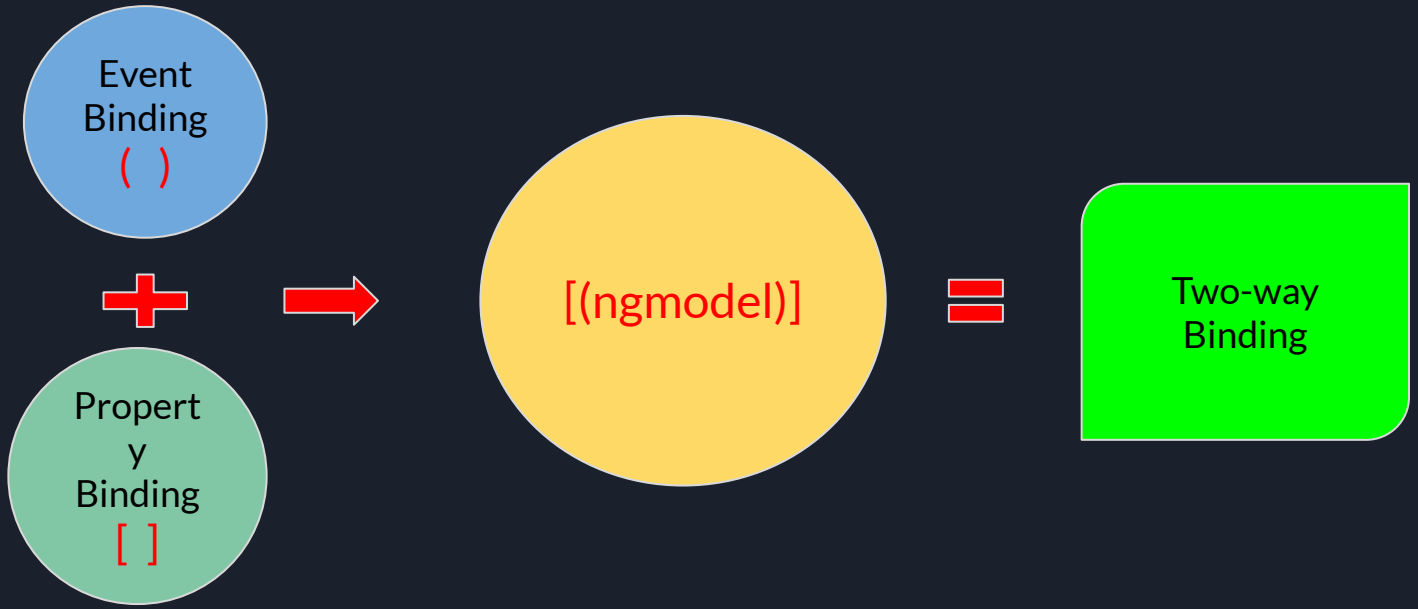
- To bind to an element's property [ ], which identifies the property as a target property.

  Example:- <img alt="item" [src]="itemImageUrl">

- To bind to an event you use the Angular event binding syntax ( ). This syntax consists of a target event name within parentheses to the left of an equal sign, and a quoted template statement to the right.

  Example:- <button (click)="onSave()">Save</button>

Two-way data binding - [( )]



Syntax:- [(ng model)]=property name(var).

Directives

Attribute directives,Structural directives,Component based directives

- Directives are classes that add additional behavior to elements in your Angular applications.
- Used to add new attributes for the existing HTML elements to change its look and behaviour.

  Example:- <HTMLTag [attr Directive]='value' />

- Used to add or remove DOM elements in the current HTML document.

  Example:-<HTMLTag [structural Directive]='value' />

- Component can be used as directives. Every component has **Input** and **Output** option to pass between component and its parent HTML elements.

  Example:-<component-selector-name [input-reference]="input-value"> ... </component-selector-name>

Attribute Directives

Attribute directives in Angular, like NgStyle, NgClass, and NgModel, modify the appearance or behavior of DOM elements or components. NgModel facilitates two-way data binding, keeping data and input values synchronized.

Systax:-`[directiveName]="value";`

Ngclass

ngClass is used to add or remove CSS classes in HTML elements.

Example:-

```
<some-element [ngClass]="'first second'">...</some-element>
```

NgStyle

ngStyle directive is used to add dynamic styles. Below example is used to apply blue color to the paragraph.

Example:-

```
<p [ngStyle]="{'color': 'blue', 'font-size': '14px'}">
paragraph style is applied using ngStyle
</p>
```

NgModel

NgModel It allows you to establish a connection between a data model in your component class and an input element in your template

Example:-

```
<input [(ngModel)]="name" #ctrl="ngModel"
```

Structural directives

- Structural directives change the structure of **DOM** by adding or removing elements. It is denoted by @ sign with three predefined directives **@If, @For** and **@Switch**.

Example:- (@for)

```
@if (condition)
    {
    Content to
render when the
  condition is
    true.
  } @else {
    Content to
render when the
  condition is
    false.
    }
```

Ex:-(@for)

```
@for (item of
items; track
  item.id) {
{{ item.name }}
    }
```

Ex:-(@switch)

```
@switch (page) {
  @case (1) {
 <p>Viewing content
of first page</p>
    }
  @case (2) {
 <p>Viewing content
of second page</p>
    }
  @default {
   <p>No page
selected</p>
    }
  }
```

Pipes

- Pipes are referred as filters. It helps to transform data and manage data within interpolation, denoted by {{ | }}.
- It accepts data, arrays, integers and strings as inputs which are separated by `|' symbol.
- In some additional pipe `DatePipe,DecimalPipe,CurrencyPipe,PercentPipe,UpperCasePipe,LowerCasePipe..etc.`

Example:- Date pipe

```
<div>

  Today's date :- {{presentDate | date }}

</div>
```

Reactive Form

- Reactive forms provide a model-driven approach to handling form inputs.
- Template-driven forms, where the form structure is defined in the HTML template, reactive forms are created programmatically in the component class using TypeScript.
- Import the Required Modules
  - import { ReactiveFormsModule } from '@angular/forms';
- Create the Form in the Component in the T.s file
- Bind Form Controls to HTML Elements

Reactive Form

Form Group

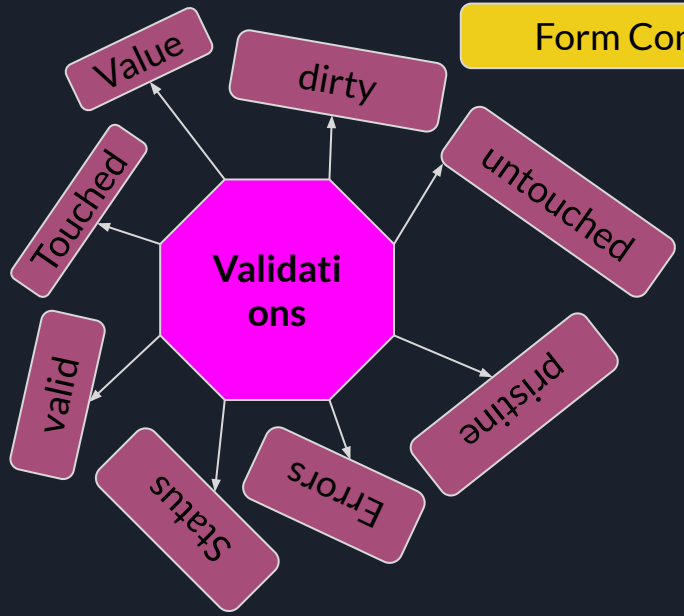
Form Control

T.S

```
login=new FormGroup( {
Name:new FormControl( ),
Password:new FormControl(
)})
show(){
console.log(this,login)
```

HTML

```
<form [FormGroup]=”login”>
username:<input type=”text”
FormControlName=”Name”>
<button(click)=”show(
)”>click</button>
```



Validations

```
New
FormControl(‘default
value;[validations])
```

Routing

- Routing in Angular refers to the mechanism of navigating between different views or pages in a single-page application (SPA).
- Angular provides its own routing module, called RouterModule, which allows developers to define navigation paths and associate them with specific components.

app.routes.ts

```
Export const routes:
     Routes = [
      { path: '',
     component:
  HomeComponent },
   { path: 'about',
     component:
  AboutComponent }
         ];
```

HTML

```
<a
roughtlink=”About”>about
</a>
<router-outlet></router-o
utlet>
```

HTTP
(Hypertext Transfer Protocol)

- HTTP (Hypertext Transfer Protocol) is a built-in module that allows you to make HTTP requests to remote servers. This module provides a way to fetch data from a server and to send data to a server over HTTP or HTTPS protocols.
- To use HTTP in Angular, you typically import the **`HttpClient Module`** from
- Angular also provides support for handling other HTTP methods like POST, PUT, DELETE, etc., through methods like `post`, `put`, and `delete` in the `HttpClient` service.

T.S

```
constructor(private http HTTP client)
{
Get data( ) {
this.http.get("link")
.subscribe(bata)=> {
console.log(data) }}
```

HTML

```
<button (click)="getData
( )">get</button>
```

Material

Materials refer to the components, styles, and design patterns provided by Angular Material, a UI component library developed by the Angular team.

Angular Material offers a set of reusable and customizable UI components following the Material Design guidelines set by Google.

Materials provided by Angular Material:

- Components
- Theming
- Layout
- Accessibility
- Animations
- Icons

You can import the materials

```
<mat-card>
<mat-card-header>
<mat-card-title>Welcome to My App</mat-card-title>
</mat-card-header>
<mat-card-content>
<mat-form-field>
<input matInput placeholder="Enter your name">
</mat-form-field>
<button mat-raised-button color="primary">Submit</button>
</mat-card-content>
</mat-card>
```

Services.

- services are a way to encapsulate reusable functionality that can be shared across components, directives, or other services within an application.
- Services are typically used for tasks such as fetching data from a server, performing data transformations, sharing state between components, or implementing application logic
- ng generate service data