**Angular: (1.34 edureka timestamp)**

**What & Why:**

* **Framework to build client side apps**
* **It helps developers to build SPA (single page application) by managing various components & interactions within the application.**
* **Modular approach**
* **Re-usable code**
* **Unit-testable**
* **It follows MVC architecture.**
  + **Model - Represented by services & components that manages application data & business logic.**
  + **View - Represented by templates, which define the ui & how data is displayed.**
  + **Controller - represented by components that handle user interaction, data binding & communication between various parts of the application.**

**Features:**

* **Component-based architecture**
* **Dependency Injection**
* **Data binding**
* **Routing**
* **Forms (validation)**
* **Http Client**
* **Testing**

**Commands:**

* **ng new proj1 - creates new project with name ‘proj1’**
* **ng serve - runs angular project**
* **ng generate component <name> - generate component**
* **npm install (in project) - installs all dependencies which are in package.json**
* **tsc –init - creates a config file for typescript to add configuration**

**Directives:**

* **ngClass -** It allows you to dynamically set the classes of an element based on the truthiness of certain expressions or properties.
* **ngStyle -** allows you to dynamically set inline CSS styles for an HTML element
* **ngIf** 
  + **true - displays element**
  + **false - hides element**

  <h2 \*ngIf="val">hello world</h2>

  val=true

//example2 with else block

  <h2 \*ngIf="name; else elseBlock" >random text</h2>

  <ng-template #elseBlock>

    <h2>hidden text</h2>

  </ng-template>

  public name:any=true

//example3 with then-else block

  <div \*ngIf="name; then thenBlock; else elseBlock"></div>

  <ng-template #thenBlock>

    <h2>then block</h2>

  </ng-template>

  <ng-template #elseBlock>

    <h2>else block</h2>

  </ng-template>

  public name:any=true

* **ngSwitch:**

<div [ngSwitch]>

    <div \*ngSwitchCase="'red'">red color</div>

    <div \*ngSwitchCase="'blue'">blue color</div>

    <div \*ngSwitchCase="'green'">green color</div>

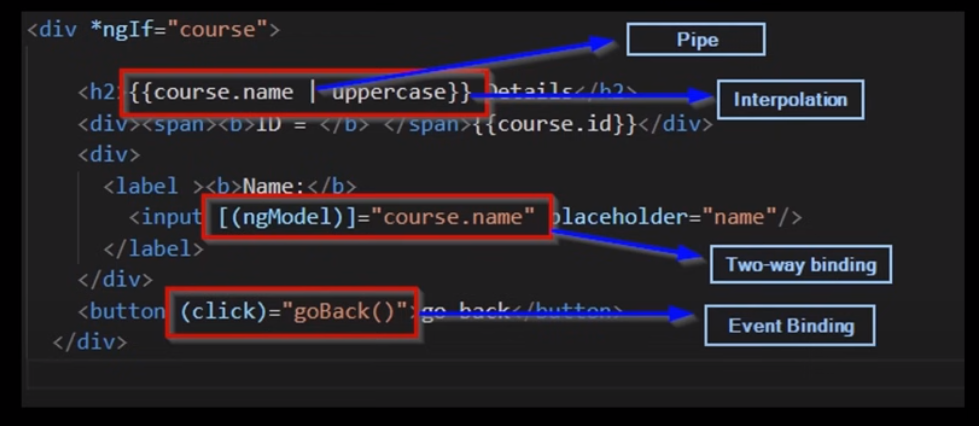
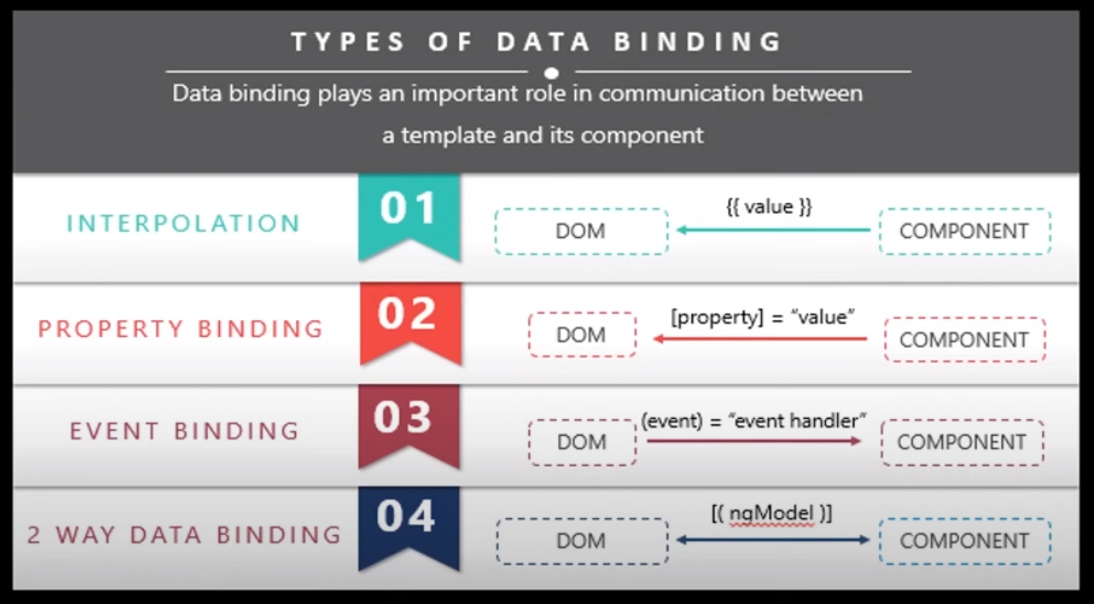
    <div \*ngSwitchDefault>pick again</div>

</div>

public color="blue"

**Building Blocks of Angular:**

* **Module:**
  + **It is a class with @NgModule metadata**
  + **Every angular app has atleast one root module**
  + **Encapsulation of similar functionalities**
* **Component:**
  + **Encapsulate UI & behaviour of specific part of application**
  + **Each component has its own style, template & logic**
* **Metadata:**
  + **Metadata describes how to process the class**
  + **Decorator used to attach metadata**
* **Template:**
  + **Defines view of a component**
  + **Looks like HTML but with few differences**
* **Data Binding:**
  + **Plays imp role in communication b/w a template & its component**

****

**Interpolation:**

* + - * **{{name}}**
      * **{{“welcome” + name}}**
      * **{{name.length}}**
      * **{{name.toUpperCase()}}**
      * **{{greetUser()}}**
      * **Cant’ bind js global variables**
      * **Can’t do assignments (a = 2 + 2)**

**Property Binding:**

* + - * **Property vs Attribute** 
        + **Both are not same**
        + **Attributes - HTML**
        + **Property - DOM**
        + **Attribute initialise DOM properties & then they are done. Attribute values can’t change once they are initialized**
        + **Property values can change**
      * **<input [id] = “myId” type = “text” value = “val”/>**
        + **[id] = bind - id**
      * **<input [disabled] = “isDisabled” type = “text” value = “val”/>**
        + **[disabled] = bind - disabled**
        + **myId, isDisabled are vars in TS**

**Class Binding:**

<input [class] = “successClass” type = “text” value = “val”/>

.text-success{

color:green;

}

Public successClass=”test-success”

<h2 [class.unsafe]="status"  >hello world</h2>

  .unsafe{

    color:red;

  }

status:boolean=true

<h2 [ngClass]="status">hello world</h2>

<h2 [ngClass]="validate()">{{text | uppercase}}</h2> //code can be written in validate() also

.unsafe{

      color:red;

    }

    .safe{

      color:green;

    }

flag: boolean = true

  status = {

    "unsafe": !this.flag,

    "safe": this.flag

  }

**Style-Binding:**

  <h2 [style.color] = "flag? 'red':'green' ">{{text | uppercase}}</h2>

  public flag = false

<h2 [ngStyle] = "styling">{{text | uppercase}}</h2>

  public text = 'random text'

  styling={

    color:"red",

    fontStyle:"italic"

  }

**Event-Binding:**

    <button (click)="onclick()">click me</button>

    <button (click)="onclick($event)">click me</button> //info about event

  onclick(event:any){

    alert("clicked on a button")

  console.log(event)

  }

**TwoWay Binding:**

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

  {{name}} //import FormsModule in app.module.ts

  public name:any=""

* + **Pipes - {{course.name | uppercase}}**
* **Services:**
  + **Services are used to provide functionality that can be shared across multiple components.**
  + **They encapsulate business logic, data manipulation, and other operations that don't belong directly in a component**
  + **Services are typically singletons.**
  + **It is a broad category of encompassing any value, function, or feature that our application needs**
    - **Ex: logging service, data service, tax calculator, message bus . . .**
* **Directive Types:**
  + **Component - <app-root> in index.html (used with template)**
  + **Attribute - (click) (change apperance/behaviour of an element/component/other directive)**
  + **Structural - \*ngFor (changes DOM layout by adding or removing elements)**

**Template Reference Variables:**

  <input type="text" #inp/>

  <button (click)=onclick(inp)>click me</button> //return entire tag

<button (click)=onclick(inp.value)>click me</button> //returns value

  onclick(inp:any){

    console.log(inp);

  }

**Notes:**

* **package.json - dependencies of project**
* **tsconfig.json - typescript configuration**
* **angular.json - angular configuration**
* **tsconfig.spec.json - related to unit testing**
* **module - logically grouping components**
* **main.ts - which module to run first is defined**
* **app.module.ts - components are listed & which component to run first is defined**
* **by using reference vars we can manipulate dom**
  + **to access It use #**