## Adding Bootstrap

npm install bootstrap –save

Note: --save will add an entry in package.json. That add as a dependency.

In style.css

@import "~bootstrap/dist/css/bootstrap.css";

# Directives

## ngIf

|  |
| --- |
| <div \*[ngIf](https://angular.io/api/common/NgIf)="condition; then thenBlock else elseBlock"></div>  <ng-template #thenBlock>Content to render when condition is true.</ng-template> <ng-template #elseBlock>Content to render when condition is false.</ng-template> |

<button (click)="condition=!condition">Click</button>

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

<ng-template #thenBlock>condition is true.</ng-template>

<ng-template #elseBlock>condition is false.</ng-template>

***Only selected div will exists in div***

## Hidden Property

<div [hidden]="condition">condition</div>

<div [hidden]="!condition">not condition</div>

***Both div will exists in DOM***

## ngSwitchCase

<ul [ngSwitch]="person">

<li \*ngSwitchCase="'Mohan'">Hello Mohan</li>

<li \*ngSwitchCase="'Sohan'">Hello Sohan</li>

<li \*ngSwitchCase="'Vijay'">Hello Vijay</li>

<li \*ngSwitchDefault>Bye Bye</li>

</ul>

## ngFor

<ul>

<li \*ngFor="let course of courses; index as i;even as isEven">

{{isEven}}.{{i}}. {{course.name}}

</li>

</ul>

<https://angular.io/api/common/NgForOf>

The following exported values can be aliased to local variables:

* $implicit: T: The value of the individual items in the iterable ([ngForOf](https://angular.io/api/common/NgForOf)).
* [ngForOf](https://angular.io/api/common/NgForOf): [NgIterable](https://angular.io/api/core/NgIterable)<T>: The value of the iterable expression. Useful when the expression is more complex then a property access, for example when using the async pipe (userStreams | async).
* index: number: The index of the current item in the iterable.
* first: boolean: True when the item is the first item in the iterable.
* last: boolean: True when the item is the last item in the iterable.
* [even](https://angular.io/api/common/NgForOfContext#even): boolean: True when the item has an even index in the iterable.
* [odd](https://angular.io/api/common/NgForOfContext#odd): boolean: True when the item has an odd index in the iterable.

ngClass

[ngClass]="{ 'btn-success': condition, 'btn-danger' : !condition }

## Safe Traversal Directive

<span>{{task?.item.id}}</span>

Angular will not throw error of item dose not exits.

# Custom Directive

[https://www.udemy.com/the-complete-angular-master-class/learn/lecture/7349054](https://www.udemy.com/the-complete-angular-master-class/learn/lecture/7349054#announcements)

# Display Data and Events

## String interpolation

{{imgURL}}

<img width="300" alt="Angular Logo" src="{{imgURL}}">

## Property Binding

[src]="imgURL"

<img width="300" alt="Angular Logo" [src]="imgURL" >

## Attribute Binding

Some attribute will not have one to one binding with DOM. In that case angular will throw error.

<td [attr.colspan]="colSpan"></td>

## Class Binding

[class.active]="isActive"

<button class="btn btn-primary " [class.active]="isActive" >Save</button>

[ngClass]="isFavourite ? 'btn-success':'btn-danger'"

[ngClass]="{ 'btn-success': isFavourite, 'btn-danger' : !isFavourite }

## Style Binding

HTML DOM Style Object: <https://www.w3schools.com/jsref/dom_obj_style.asp>

[style.backgroundColor]=" isActive ? 'blue' : 'red' "

<button [style.backgroundColor]=" isActive ? 'blue' : 'red' " >Save</button>

## Event Binding

(click)="onSave()"

(click)="onSave($event)"

onSave($event)

## Event bubbling:

$event.stopPropagation();

<div (click)="onDivClicked()">

<button (click)="onSave($event)" >Save</button>

</div>

onSave($event){

$event.stopPropagation();

this.isActive = !this.isActive;

console.log($event);

}

onDivClicked(){

console.log("Div clicked");

}

## Event Filtering

<input (keyup.enter)="onKeyUp()">

<input (keyup.enter)="onKeyUp($event)">

onKeyUp($event){

console.log("enter clicked: " + $event.target.value);

}

## Template Variable

<input #email (keyup.enter)="onKeyUp1(email.value)">

## Two-way binding

In app.module

import { FormsModule } from'@angular/forms';

imports: [ FormsModule,

<input [(ngModel)]="Title" (keyup.enter)="onKeyUp4()">

onKeyUp4(){

console.log("email: " + this.Title);

}

## Pipes

To format data

{{Title | uppercase | lowercase}}

{{Salary | number}}

Adds coma like 2,000

{{Rating | number:'1.2-2'}}

No of digits before dot(adds 0 before) . min number of digits after dot – max number of digits after dot

{{1.1234 | number:'3.2-2'}} => 001.12

1.234

{{price | currency:'INR'}}

{{price | currency:'INR':false:'3.2-2'}}

False > show current symbol or not

{{DOB | date:'shortDate'}}

<https://angular.io/api/common/DatePipe>

* 'short': equivalent to 'M/d/yy, h:mm [a](https://angular.io/api/router/RouterLinkWithHref)' (6/15/15, 9:03 AM).
* 'medium': equivalent to 'MMM d, y, h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref)' (Jun 15, 2015, 9:03:01 AM).
* 'long': equivalent to 'MMMM d, y, h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref) z' (June 15, 2015 at 9:03:01 AM GMT+1).
* '[full](https://angular.io/api/core/Version#full)': equivalent to 'EEEE, MMMM d, y, h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref) zzzz' ([Monday](https://angular.io/api/common/WeekDay#Monday), June 15, 2015 at 9:03:01 AM GMT+01:00).
* 'shortDate': equivalent to 'M/d/yy' (6/15/15).
* 'mediumDate': equivalent to 'MMM d, y' (Jun 15, 2015).
* 'longDate': equivalent to 'MMMM d, y' (June 15, 2015).
* 'fullDate': equivalent to 'EEEE, MMMM d, y' ([Monday](https://angular.io/api/common/WeekDay#Monday), June 15, 2015).
* 'shortTime': equivalent to 'h:mm [a](https://angular.io/api/router/RouterLinkWithHref)' (9:03 AM).
* 'mediumTime': equivalent to 'h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref)' (9:03:01 AM).
* 'longTime': equivalent to 'h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref) z' (9:03:01 AM GMT+1).
* 'fullTime': equivalent to 'h:mm:ss [a](https://angular.io/api/router/RouterLinkWithHref) zzzz' (9:03:01 AM GMT+01:00).

## Custom Pipe

* Add ts file => app\summary.pipes.ts
* <https://angular.io/api/core/PipeTransform>

import { Pipe, PipeTransform } from '@angular/core';

@Pipe({name: 'summary'})

export class SummaryPipe implements PipeTransform {

transform(value: string, limit?:number, endsWith?:string): string {

if(!value){

return null;

}

if(!limit){

limit=50;

}

if(!endsWith){

endsWith = "...";

}

return value.substring(0, 50) + " " + endsWith;

}

}

* In app.module.ts

import { SummaryPipe } from './summary.pipes';

declarations: [ SummaryPipe,

In HTML template

{{Summary | summary:10:'\*\*\*'}}

## Reusable Components

## Input Properties

In component

import { Input } from '@angular/core';

@Input() isFavourite:boolean = false;

Or

@Input('is-favourite') isFavourite:boolean;

//Where 'is-favourite'is alias

Pass data as property:

<app-favourate [isFavourite]="post.isFavourite” ></app-favourate>

Or

<app-favourate [is-favourite]="post.isFavourite” ></app-favourate>

## Output Properties

In component

import { EventEmitter } from '@angular/core';

@Output('fav-change') change = new EventEmitter();

onClick(){

this.isFavourite = !this.isFavourite;

this.change.emit( { newValue: this.isFavourite } );

}

<app-favourate (fav-change)="onFavouriteChanged($event)" ></app-favourate>

onFavouriteChanged(eventArgs:FavouriteChangedEventArgs ){

alert("onFavouriteChanged - " + eventArgs.newValue);

}

interface FavouriteChangedEventArgs{

newValue:boolean

}

# Template Driven Forms

## ZEN Coding

|  |  |  |  |
| --- | --- | --- | --- |
| . | class | div.name | <div class="name"></div> |
| # | id | div#name | <div id="name"></div> |
| > | Child element | table>tr>td |  |
| [] | attribute | p[title] | <p title=""></p> |
| + | New element | p.one+p.two |  |
| $ | Dynamic id | p.name-$\*3 |  |
|  |  |  |  |
|  |  |  |  |

label+input[type='text'].form-control

---------------------------------------------------------------------------------

<label for=""></label>

<input type="text" class="form-control">

div#name

---------------------------------------------------------------------------------

<div id="name"></div>

div.name

---------------------------------------------------------------------------------

<div class="name"></div>

table>tr>td

---------------------------------------------------------------------------------

<table>

<tr>

<td></td>

</tr>

</table>

**Use > for child element.**

p.one+p.two

---------------------------------------------------------------------------------

<p class="one"></p>

<p class="two"></p>

**Use +, more than one element is expanded.**

p[title]

---------------------------------------------------------------------------------

<p title=""></p>

p\*3

---------------------------------------------------------------------------------

<p></p>

<p></p>

<p></p>

p.name-$\*3

---------------------------------------------------------------------------------

<p class="name-1"></p>

<p class="name-2"></p>

<p class="name-3"></p>

**By using $, you can include sequential numbers within the ID names of class names**

## Bootstrap form