ANGULAR NOTLARI

APO:Application Programming Interfaces

Npm: node package manager managin 3rd party librarires

Angular CLI:

Npm install –g @angular/cli

ng --version

ng new hello-world

ng serve

**Folders**

e2e: end to end test, automated test simulate real user

node\_modules: 3rd libraries

src: source code

app: module component

assets: static images vs

envirements: configurations

main:bootstaring mainmodule

polyfills: fill the gap js and angular

karma: test runner js code

package.json: dependecies

prtractor.conf.js: running e2e tst

tsconfig.js: typscript compiler configs

tslint.json: static anlaysit tool for typescrit code

**Webpack**

A tool that creates bundles for JavaScript code and stylesheets and injects those bundles into index.html at run-time.

Compiles

Hot module Replacement(hmr)

İnjects bundles to index.html

**Versions**

AngularJS

Angular2 Angular4

Typescript

Core

Compiler

http

router --different from angular4

**Typescript**

**Brief Info**

Superset of JS

Typescript =>transpile=> JS

Strong typeing

OOP

Compile-time errors

Great tooling(intellisense)

**Install**

**npm install –g typescript**

**tsc –version**

tsc app.ts = compiles to JS

node main.js= runs js codes out of browser

compile: tsc main.ts

run: main.js

**Declaring variables**

**Var** :

available outside of scope of function

**let**

ec2016 included in Js

not available outside of scope of function

use let to diagnose compiler errors!!!

**Types**:

**let**:

let a: number;

let a; ==> any

let e: number[];

let e: number[]=[1,2,3];

let g: any[]=[1,"sdfs",'as'];

**const**:

const ColorRed=0;

const ColorGreen=1;

**enum**:

enum Color{Red =0,Green=1,Blue=2};

let backGroundColor=Color.Red;

**Type assertion:**

let endsWithC= (<string>message).endsWith('c');

let alternativeWay= (message as string ).endsWith('c');

**Arrow Functions:**

let log=function(message){

console.log(message);

}

let doLog=(message) =>{

console.log(message);

}

let doLog2=(message) => console.log(message);

let doLog3=() => console.log();

Interfaces:

interface Point{

x:number,

y:number

}

let drawPoint=(point:Point)=>

{

//...

}

drawPoint({

x:1,

y:2

}

)

interface Point{

x:number,

y:number,

draw:()=>void //dont need paramter because it access x and y

}

**Class:**

class Point{

x: number;

y: number;

draw:() {

//..

}

getDistance(another:Point)

{

//..

}

}

Objects

class Points{

x: number;

y: number;

draw() {

console.log('X:'+ this.x +',Y:' +this.y);

}

getDistance(another: Points)

{

}

}

let points = new Points();

points.x=3;

points.y=4;

points.draw();

**Constructors:**

There must be just one constructor

If you make one parameter optinal all other right side paramters also have to be optional

class Points{

x: number;

y: number;

constructor (x?:number,y?: number){

this.x=x;

this.y=y;

}

draw() {

console.log('X:'+ this.x +',Y:' +this.y);

}

}

let points = new Points(3,4);

points.draw();

**Access modifiers**

Public

Protected

Private

**Access modifiers in constructors**

class Points {

constructor(private x?: number, public y?: number) {

}

draw() {

console.log('X:' + this.x + ',Y:' + this.y);

}

}

let points = new Points(3, 4);

points.draw();

**Property**

class Points {

constructor(private x?: number, private y?: number) {

}

draw() {

console.log('X:' + this.x + ',Y:' + this.y);

}

get X(){

return this.x;

}

get Y(){

return this.y;

}

set X(value)

{

if(value<0)

throw new Error("value can not be less than 0");

this.x=value;

}

}

let points = new Points(3, 4);

let x=points.X;

points.X=10;

points.draw();

**Modules**

export class Point {

import {Point,a,b,c}

import {Point} from './Point'

**Compile Error**

Error: targeting ECMAScript 5 and higher

Solution: tsc \*.ts --target ES5 && node main.js

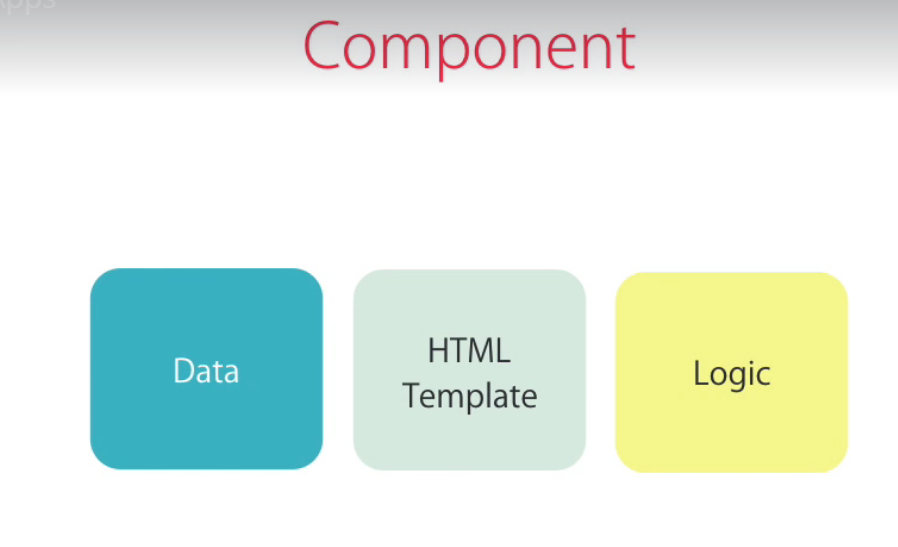
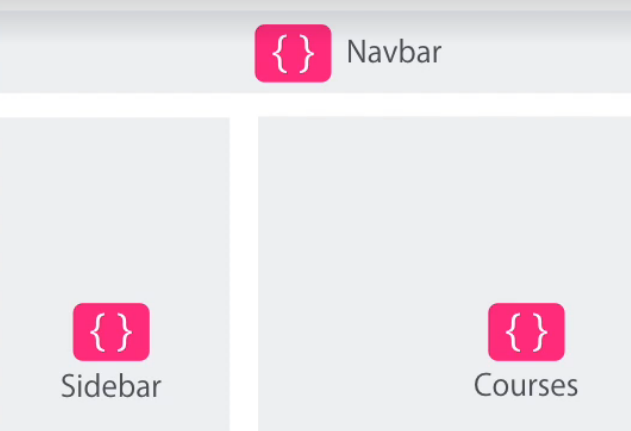
**ANGULAR FUNDAMENTALS**

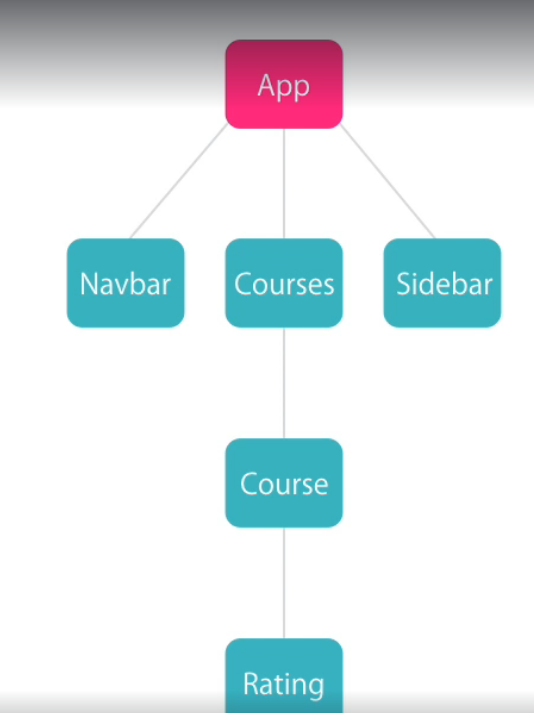
**Component**

Reuse

Work smaller

App component

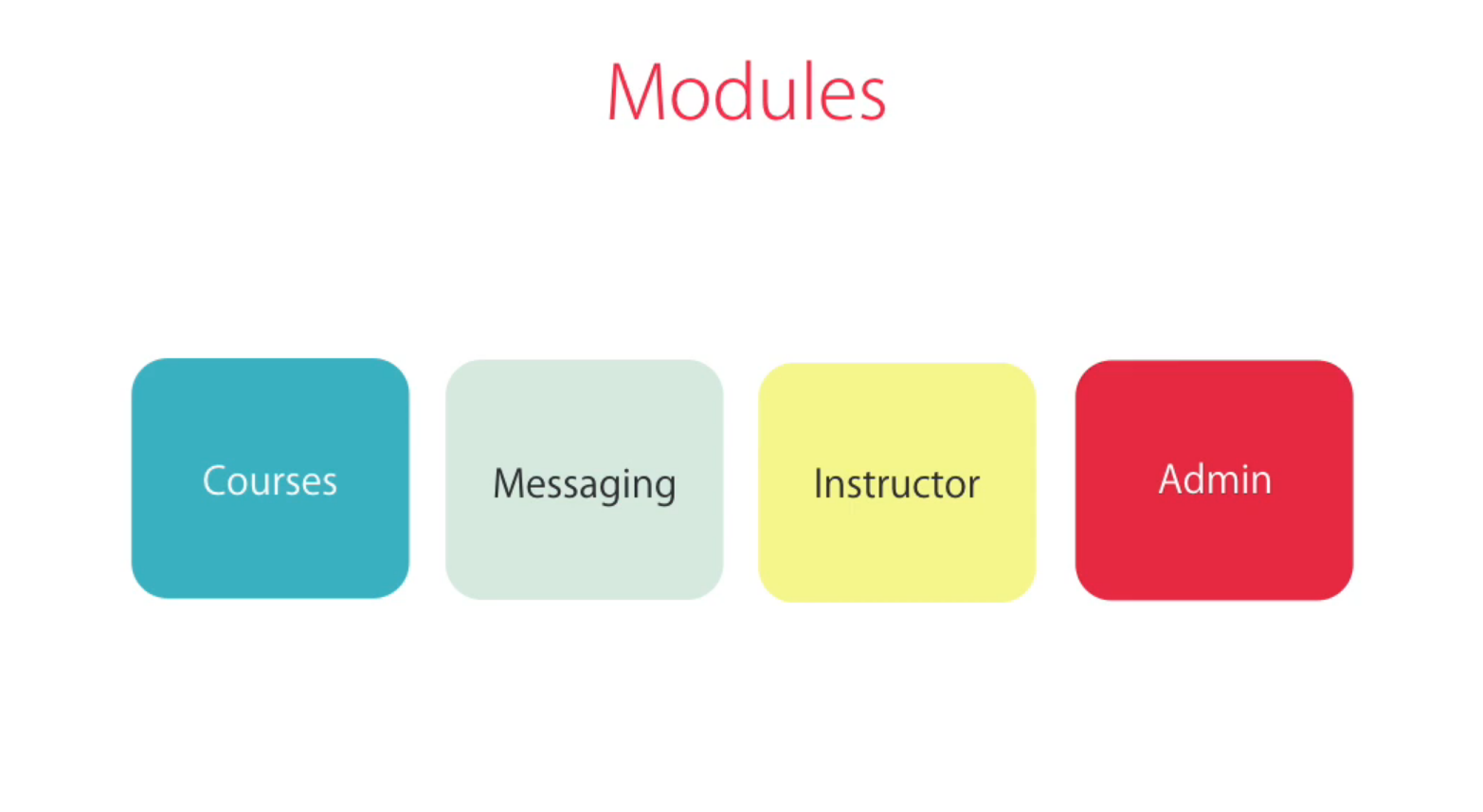




**Modules**

Consist of components

App module

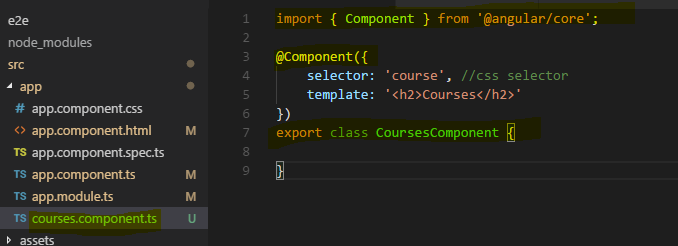


**How to Create Component**

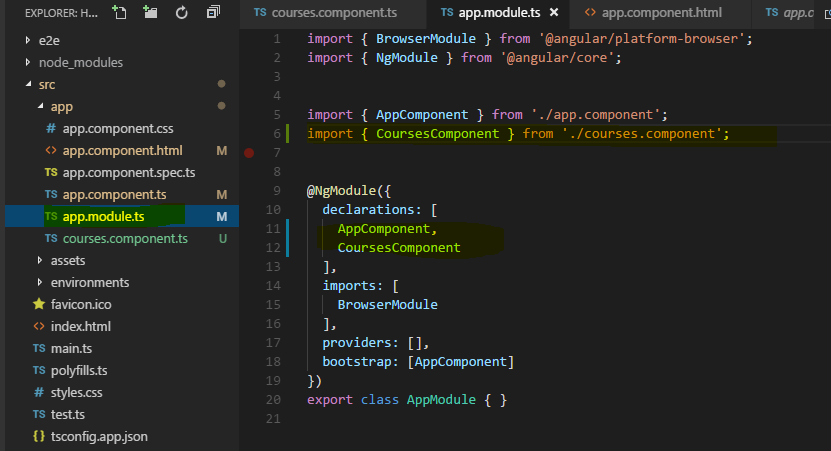
1**.**Create component

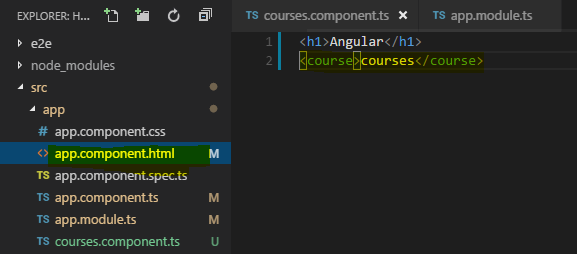
2.Register it in module

3.Add an element in an html markup

1. 

2.



3. 

**Create Componen From Angular CLI**

Ctrl+” terminali açar

Ctrl + B project explorer

Path’e %AppData%\npm ekle

ng g c “componentname”

css

html =template

spec=unit test

ts

updates app.module and register to module

**Templates**

String interpolation

{{ }} içinde property method kullanılabiliyor.

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

@Component({

selector: 'course', //css selector

template: '<h2>{{ title}}</h2>'

})

export class CoursesComponent {

title="List of Courses";

getTitle(){

return this.title;

}

}

**Directives**

**Used for manipulate DOM**

**Start with \***

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

@Component({

selector: 'course', //css selector

template: `

<h2>{{title}}</h2>

<ul>

<li \*ngFor="let course of courses">

{{course}}

</li>

</ul>

`

})

export class CoursesComponent {

title = "List of Courses";

courses = ["Course 1", "Course 2", "Course 3"];

}

**Services**

export class CoursesService{

getCourses(){

return ["Course 1", "Course 2", "Course 3"];

}

}

**Dependecy Injection**

Component constructor

constructor(service: CoursesService){

this.courses=service.getCourses();

}

Module @NgModule

providers: [

CoursesService

],

Singleton

**Generate server in Angular CLI**

ng g s email

spec. =unit tset

service.ts

@Injectabel decorative funciton sevice nject other classes

shortcuts

Ctrl +P => open file

Ctrl + Tab => navagate in open files

ul>li Tab => ul li

**SECTION 4: DISPLAYING DATA and HANDLING EVENTS**

**Property Binding**

<img [src]="imageUrl"/>

Interpolation => property binding

Interpolation use for text

Property binding is only one way

Binds to DOM not HTML

1. **Attribute Binding**

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

Yanlış çünkü DOM’da colpan yok Binds to DOM not HTML

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

1. **Adding Bootstrap**

getbootstrap.com bak

npm install bootstrap –save

npm install package.jsaondaki tüm dependencyleri install eder

*styles.css içine*

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

1. **Style Binding**

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

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

**Event Binding**

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

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

@Component({

selector: 'course', //css selector

template: `

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

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

</div>

`

})

export class CoursesComponent {

onSave($event) {

$event.stopPropagation(); //stops event bubbling

console.log("Button clicked", $event);

}

onDivClick() {

console.log("On div clicked");

}

}

**Event Filtering**

**Eski**

@Component({

selector: 'course', //css selector

template: `

<input (keyup)="onKeyUp($event)"/>

`

})

export class CoursesComponent {

onKeyUp($event){

if($event.keyCode === 13) console.log("Enter pressed");

}

}

**Angular vesiyonu**

@Component({

selector: 'course', //css selector

template: `

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

`

})

export class CoursesComponent {

onKeyUp(){

console.log("Enter pressed");

}

}

**Template Variables**

**Eski**

@Component({

selector: 'course', //css selector

template: `

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

`

})

export class CoursesComponent {

onKeyUp($event){

console.log($event.target.value);

}

}

**Angular**

@Component({

selector: 'course', //css selector

template: `

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

`

})

export class CoursesComponent {

onKeyUp(email){

console.log(email);

}

}

**Two Way Binding**

**[(ngModel)]**

**App-module de forms mudle import et**

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

`<span [(ngModel)]="isFavourite" class='glyphicon glyphicon-star' (click)="onClick(isFavourite)"></span> `

imports: [

BrowserModule,

FormsModule

],

**Pipes**

**Used to format data**



@Component({

selector: 'course', //css selector

template: `

{{ course.title | uppercase | lowercase }} <br/>

{{ course.students | number }} <br/>

{{ course.rating | number:'2.1-1' }} <br/>

{{ course.students }} <br/>

{{ course.price | currency:'TRY':true:'3.2-2'}} <br/>

{{ course.releaseDate|date:'dd/MM/yyyy' }} <br/>

`

})

**Custome Pipes**

**1.pipe classını oluştur**

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

@Pipe({

name: 'summary'

})

export class SummaryPipe implements PipeTransform {

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

if (!value)

return null;

let actuallimit=(limit) ? limit :50;

return value.substr(0, actuallimit) + '...';

}

}

**2.App module register et**

@NgModule({

declarations: [

AppComponent,

CoursesComponent,

TestComponent,

SummaryPipe

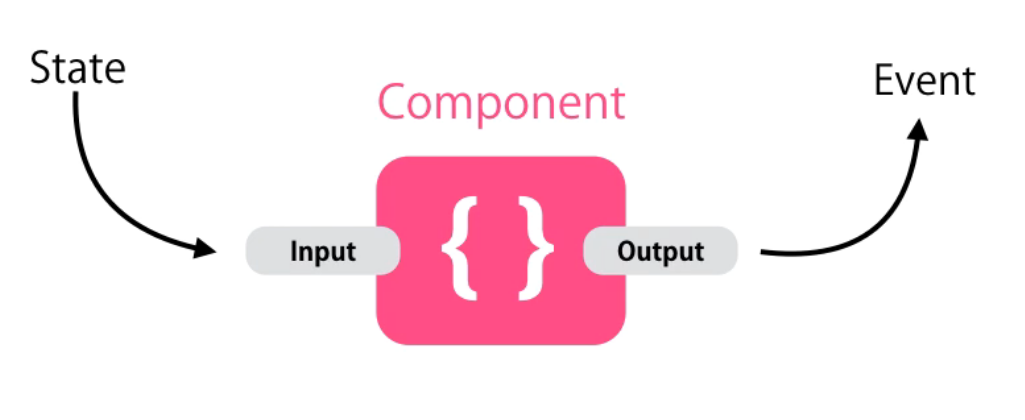
],

1. **Kullan ☺**

{{ text |summary:10 }}

**COMPONENTS**

**Component API**



**Input**

import { Component, OnInit,Input } from '@angular/core';

export class FavouriteComponent implements OnInit {

@Input('is-favorite') isFavorite=false;

}

export class AppComponent {

title = 'Angular app';

post={

title:"Title",

isFavorite:true

}

App.component.html

<favourite [is-favorite]="post.isFavorite"></favourite>

II .YOL => KULLANMA !!!

@Component({

selector: 'favourite',

templateUrl: './favourite.component.html',

styleUrls: ['./favourite.component.css'],

inputs:['isFavorite']

})

**OUTPUT**

import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Component({

selector: 'favourite',

templateUrl: './favourite.component.html',

styleUrls: ['./favourite.component.css'],

})

export class FavouriteComponent implements OnInit {

@Input('is-favorite') isFavorite = false;

@Output() change=new EventEmitter();

onClick() {

this.isFavorite = !this.isFavorite;

this.change.emit();

}

constructor() { }

ngOnInit() {

}

}

App.component.ts

onFavoriteChanged() {

console.log("Favorite changed");

}

App.component.html

<favourite [is-favorite]="post.isFavorite" (change)="onFavoriteChanged()"></favourite>

**Passing Event Data**

import { Component, OnInit, Input, Output, EventEmitter } from '@angular/core';

@Component({

selector: 'favourite',

templateUrl: './favourite.component.html',

styleUrls: ['./favourite.component.css'],

})

export class FavouriteComponent implements OnInit {

@Input('is-favorite') isFavorite = false;

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

onClick() {

this.isFavorite = !this.isFavorite;

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

}

constructor() { }

ngOnInit() {

}

}

export interface FavouriteChangedEventArgs {

newValue: boolean

}

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

import { FavouriteChangedEventArgs } from './favourite/favourite.component';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular app';

post = {

title: "Title",

isFavorite: true

}

onFavoriteChanged(eventArgs: FavouriteChangedEventArgs) {

console.log("Favorite changed", eventArgs);

<favourite [is-favorite]="post.isFavorite" (change)="onFavoriteChanged($event)"></favourite>

**TEMPLATES**

**Sadece biri kullanılabilir**

TemplateUrl: EXTERNAL USE IN BIG MAİNBUNDLE.JS İÇİNDE

Template: INTERNAL USE IN SMALL

**STYLES**

Sırasıyla en sadece sondakini alır

styleUrls: ['./favourite.component.css'],

styles: [

`

`

]

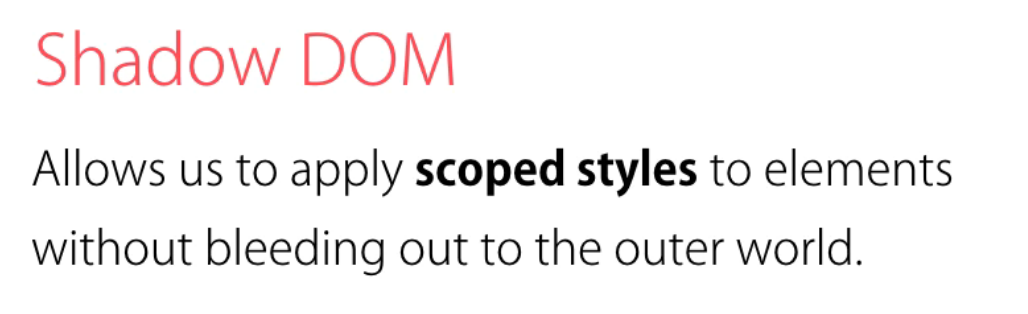
<style>

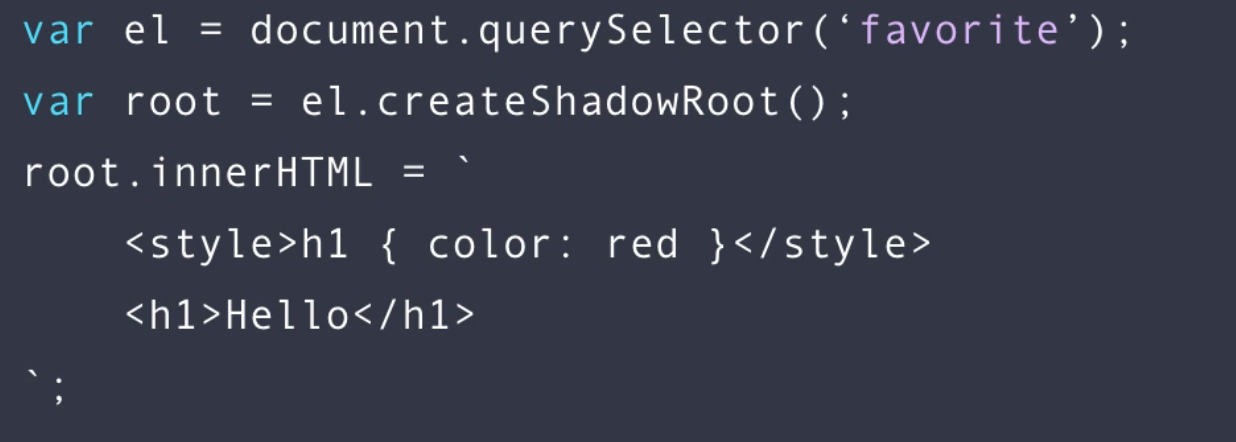
</style>

**View Encapsulation**

Styles template dışında kullanılmıyor.

Shadow DOM





@Component({

selector: 'favourite',

templateUrl: './favourite.component.html',

styleUrls: ['./favourite.component.css'],

encapsulation: ViewEncapsulation.Emulated // emulates shadowDOM

encapsulation: ViewEncapsulation.Native // native shadow root import //bootstrap performans problemi yaratır

encapsulation: ViewEncapsulation.None //leak outside template

})

**NgContent**

div.panel.panle-default>div.panel-heading+div.panel-body (TAB’A BAS)

when create reusable component like bootstrap component end consumer add custom content use ngContent

you dont need a selector whene you have one ngContent

panel.content.html

<div class="panel panel-default">

<div class="panel-heading">

<ng-content select=".heading"></ng-content>

</div>

<div class="panel-body">

<ng-content select=".body"></ng-content>

</div>

</div>

App.component.html

<bootstrap-panel>

<div class="heading">Heading</div>

<div class="body">

<h2>Body</h2>

<p>Some Content Here...</p>

</div>

</bootstrap-panel>

**ngContanier**

**render somethin without html markup only inside**

App.component.html

<bootstrap-panel>

<ng-container class="heading">Heading</container>

<div class="body">

<h2>Body</h2>

<p>Some Content Here...</p>

</div>

</bootstrap-panel>

**Like Component**

Grey:#ccc deeppig

Cursor:pointer

import { Component, OnInit, Input } from '@angular/core';

@Component({

selector: 'like',

templateUrl: './like.component.html',

styleUrls: ['./like.component.css']

})

export class LikeComponent {

@Input('likesCount') likesCount: number;

@Input('isActive') isActive: boolean;

onClick() {

this.likesCount +=(this.isActive)?-1:+1;

this.isActive=!this.isActive;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

.glyphicon{

color:#ccc;

cursor:pointer;

}

.highlighted{

color: deeppink;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<span

class="glyphicon glyphicon-heart"

[class.highlighted]="isActive"

(click)="onClick()"

>

</span>

<span>{{likesCount}}</span>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

import { FavouriteChangedEventArgs } from './favourite/favourite.component';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'Angular app';

tweet = {

bofy: '...',

likesCount:10,

isLiked: true

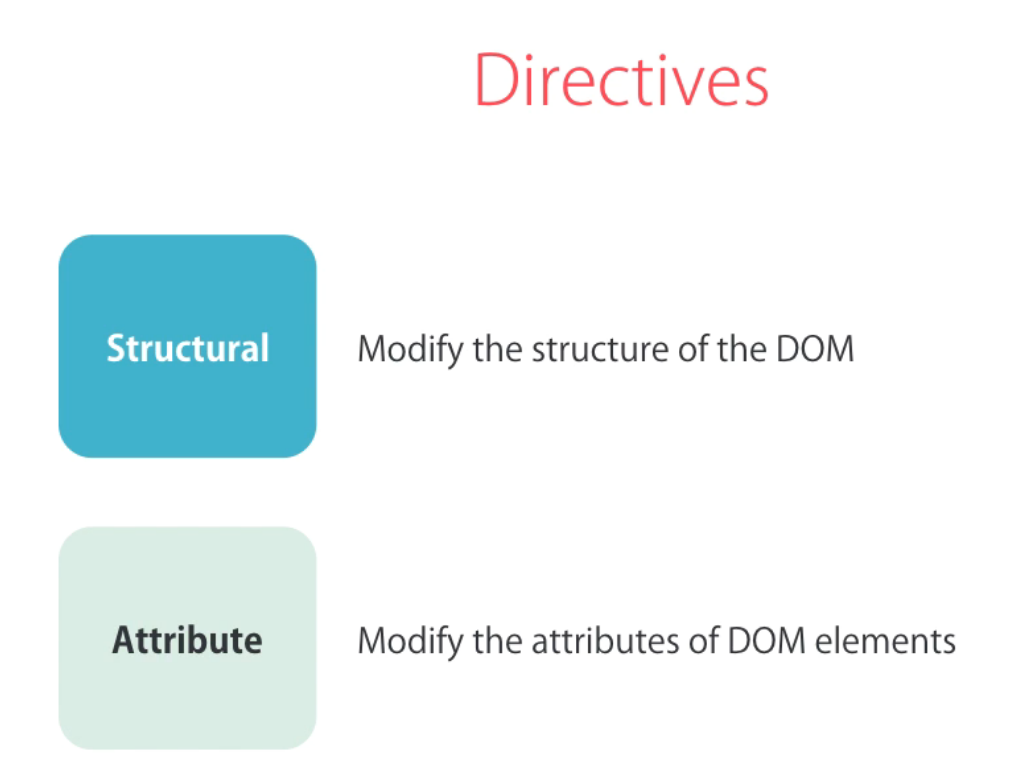
}

}

<like [likesCount]="tweet.likesCount" [isActive]="tweet.isLiked"

></like>

**DIRECTIVES**



**ngIf**

Structural

Use \*

If olmazsa domdan kaldırır

İf else

<div \*ngIf="courses.length>0; else noCourses">

List of Courses

</div>

<ng-template #noCourses>

No Courses yet

</ng-template>

2.YOL

<div \*ngIf="courses.length>0;then coursesList else noCourses">

</div>

<ng-template #coursesList></ng-template>

<ng-template #noCourses>

No Courses yet

</ng-template>

**HIDDEN PROPERTY**

**Element is in the dom**

<div [hidden]="courses.length == 0"> List of Courses</div>

<div [hidden]="courses.length > 0">No Courses Yet</div>



**ngSwitchCase**

**Zencode**

**ul.nav.nav-pills 🡪 TAB -->** <ul class="nav nav-pills">

**(li>a)\* 🡪 TAB 🡪**  <li class="active"><a href="">Map View</a></li>

<li><a href="">List View</a></li>

<ul class="nav nav-pills">

<li [class.active]="viewMode == 'map'">

<a (click)="viewMode ='map'">Map View</a>

</li>

<li [class.active]="viewMode == 'list'">

<a (click)="viewMode = 'list' ">List View</a>

</li>

</ul>

<div [ngSwitch]="viewMode">

<div \*ngSwitchCase="'map'">Map View Content</div>

<div \*ngSwitchCase="'list'">List View Content</div>

<div \*ngSwitchDefault>Otherwise</div>

</div>

**ngFor**

<ul>

<li \*ngFor="let course of courses">

{{course.name}}

</li>

</ul>

<ul>

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

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

</li>

</ul>

<ul>

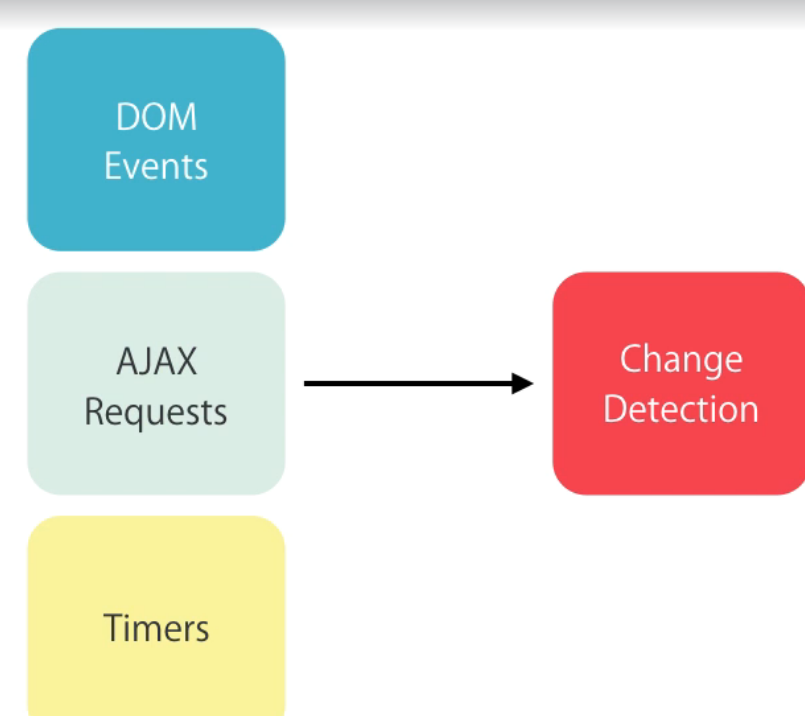
<li \*ngFor="let course of courses;even as isEven">

{{ course.name }} <span \*ngIf="isEven">(EVEN)</span>

</li>

</ul>

**ngFor and change Detection**



<button (click)="onAdd()">Add</button>

<ul>

<li \*ngFor="let course of courses">

{{ course.name }}

<span>

<button (click)="onRemove(course)">Remove</button>

<button (click)="onChange(course)">Change</button>

</span>

</li>

</ul>

courses = [

{ id: 1, name: 'course-1' },

{ id: 2, name: 'course-2' },

{ id: 3, name: 'course-3' }

]

onAdd() {

this.courses.push({ id: 4, name: 'course-4' });

}

onRemove(course) {

let index = this.courses.indexOf(course);

this.courses.splice(index, 1);

}

onChange(course) {

course.name = 'UPDATED';

}

**ngFor and TrackBy**

Angular her detectionda dom objecti tekrardan yüklüyor(sadec o listeyi) ama liste büyürse performans sorunu olur.

Defaultta memorylocationa bakıyor farklıysa reconstruct ediyor content aynı olsa bile

Bunu trackBy ile değiştiriyoruz.

Simple list ise hiç trackBy’a girme.

Use it only when you need to

<button (click)="loadCourses()">Load Courses</button>

<ul>

<li \*ngFor="let course of courses; trackBy: trackCourse">

{{ course.name }}

</li>

</ul>

trackCourse(index, course) {

return course ? course.id : undefined;

}

**The leading asterix\***

**Aşağıdaki iki kod aynı işi yapıyor**

<div \*ngIf="courses.length > 0; else noCourses">

List of courses

</div>

<!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

<ng-template [ngIf]="courses.length > 0">

<div>

List of Courses

</div>

</ng-template>

<ng-template [ngIf]="!(courses.length > 0)">

No Courses

</ng-template>

**ngClass**

**Eski**

<span

class="glyphicon"

[class.glyphicon-star]="isFavorite"

[class.glyphicon-star-empty]="!isFavorite"

(click)="onClick()"

></span>

**Yeni**

<span

class="glyphicon"

[ngClass]="{

'glyphicon-star': isFavorite,

'glyphicon-star-empty': !isFavorite

}"

(click)="onClick()"

></span>

**ngStyle**

**Eski**

<button

[style.backGroundColor]="canSave ? 'blue': 'gray'"

[style.color]="canSave ? 'white' :' black'"

[style.fontWeight]="canSave ? 'bold': 'normal'"

>

Save

</button>

**Yeni**

<button

[ngStyle]="{

'backgroundColor': canSave ? 'blue':'gray',

'color': canSave ? 'white': 'black',

fontWeight: canSave ? 'white':'black'

}"

>

Save

</button>

**Safe Traversal Operator(Null Pointer)**

<span \*ngIf="task.assignee">{{task.assignee.name}}</span>

<span >{{task.assignee?.name}}</span>

**Creating Custom Directives**

**Ng g d directive-name**

**İmport** HostListener

@HostListener('focus') onFocus()

{

console.log("On focus");

}

<input type="text" appInputFormat [format]="'uppercase'">

import { Directive,HostListener, ElementRef, Input } from '@angular/core';

@Directive({

selector: '[appInputFormat]'

})

export class InputFormatDirective {

constructor(private el:ElementRef) { } //gives accesss to DOM objecct

@Input('appInputFormat') format;

@HostListener('focus') onFocus()

{

console.log("On focus");

}

@HostListener('blur') onBlur()

{

let value:string =this.el.nativeElement.value; //current dom object

if(this.format =='lowercase')

this.el.nativeElement.value=value.toLowerCase();

else

this.el.nativeElement.value=value.toUpperCase();

}

}

**Ng-content -> dışarıdan verilen content**

**Template Driven Forms**

Label+input[type=’text’].form-control ->> TAB

**Create Bootstrap Form**

<form>

<div class="form-group">

<label for="firstName">First Name</label>

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

</div>

<div class="form-group">

<label for="comment">Comment</label>

<textarea id="comment" cols="30" rows="10" class="form-control">

</textarea>

</div>

<button class="btn btn-primary">Submit</button>

</form>

**Types of Form**

**FormControl**

For each input in form we need to instance of control class

**FormControl**

Value

Touched

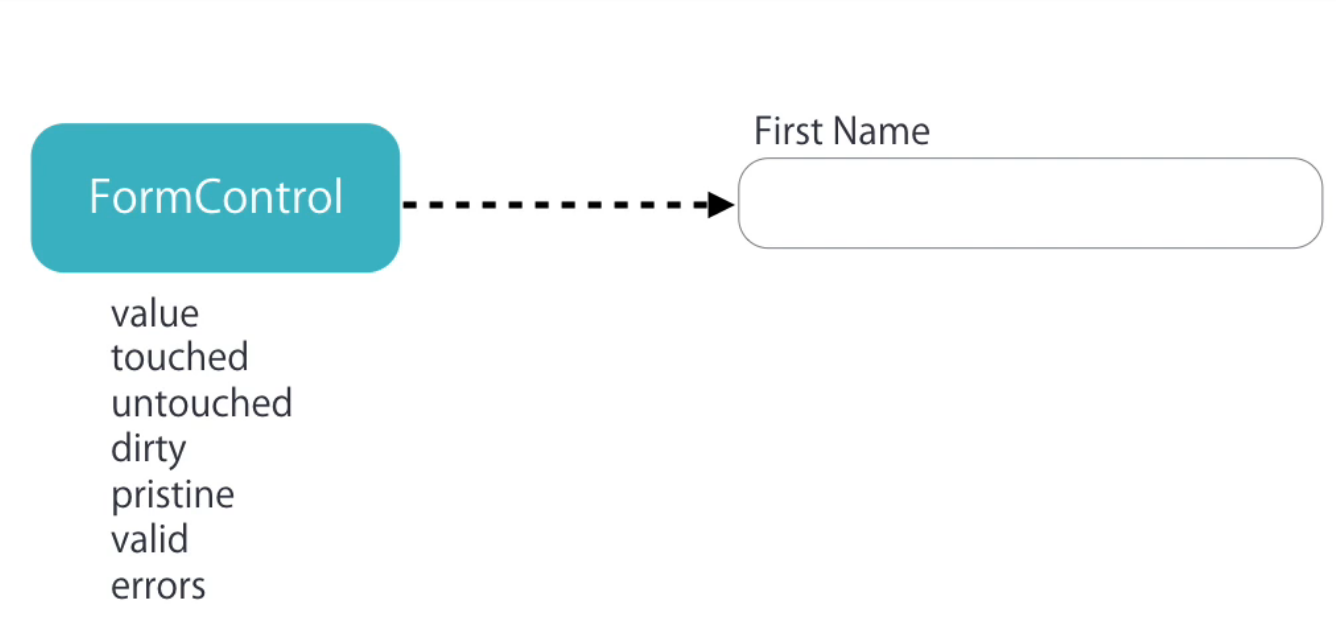
Untouched

Dirty : value’s changed

Pristine: values is not changed

Valid

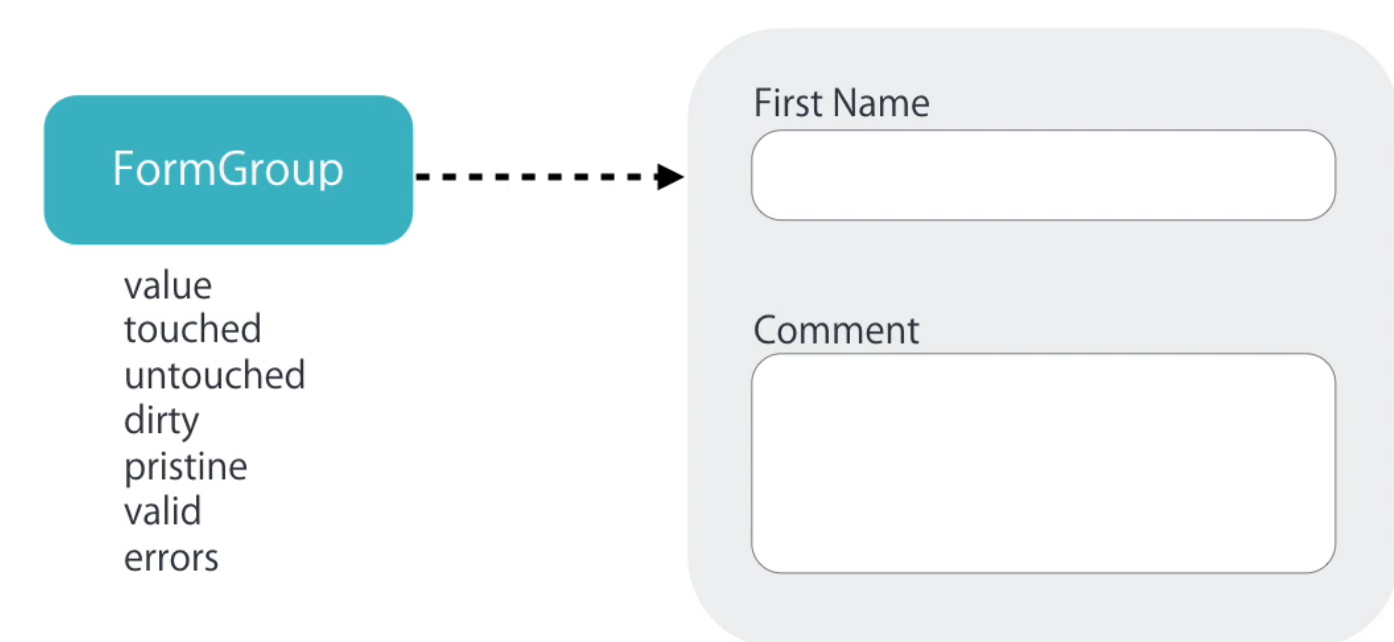
Errors: validaiton errors



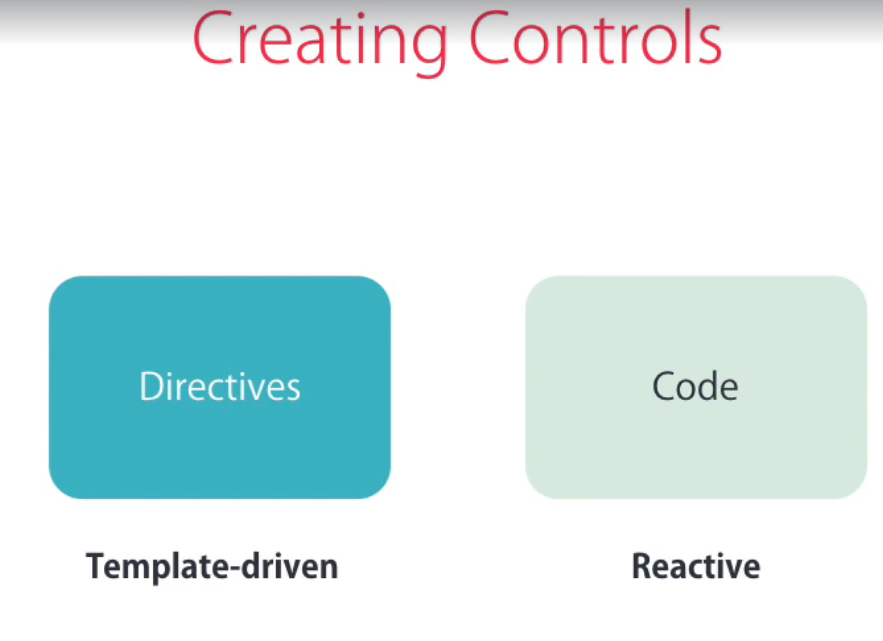
**FormGroup**

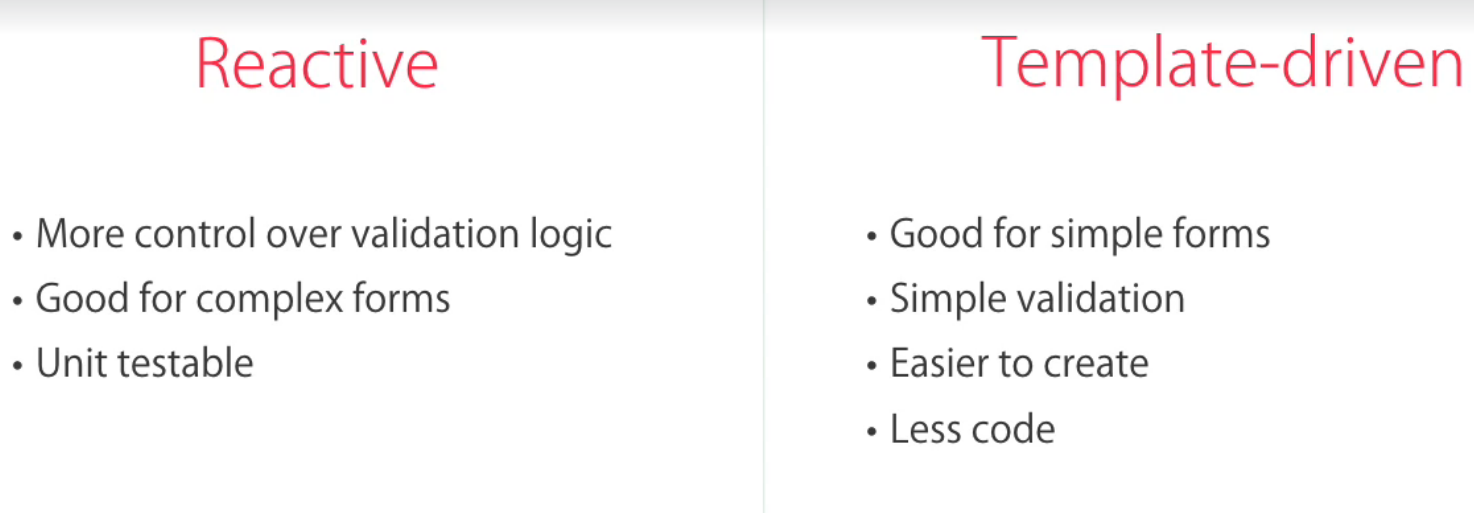
**Group of controls**

**Checks all controls**









**ngModel**

1. Two way binding
2. Angular creates control object under the hood

<form>

<div class="form-group">

<label for="firstName">First Name</label>

<input ngModel name="firstName" #firstName="ngModel" (change)="log(firstName)" id="firstName" type="text" class="form-control">

</div>

<div class="form-group">

<label for="comment">Comment</label>

<textarea ngModel name="comment" id="comment" cols="30" rows="10" class="form-control">

</textarea>

</div>

<button class="btn btn-primary">Submit</button>

</form>

**Adding Validation**

<form>

<div class="form-group">

<label for="firstName">First Name</label>

<input required ngModel name="firstName" #firstName="ngModel" (change)="log(firstName)" id="firstName" type="text" class="form-control">

<div class="alert alert-danger" \*ngIf="firstName.touched && !firstName.valid"> Firs Name is required </div>

</div>

<div class="form-group">

<label for="comment">Comment</label>

<textarea ngModel name="comment" id="comment" cols="30" rows="10" class="form-control">

</textarea>

</div>

<button class="btn btn-primary">Submit</button>

</form>

**Specific Validation Errors**

<form>

<div class="form-group">

<label for="firstName">First Name</label>

<input required minlength="3" maxlength="10" pattern="banana" ngModel name="firstName" #firstName="ngModel" (change)="log(firstName)"

id="firstName" type="text" class="form-control">

<div class="alert alert-danger" \*ngIf="firstName.touched && !firstName.valid">

<div \*ngIf="firstName.errors.required">First Name is required</div>

<div \*ngIf="firstName.errors.minlength">First Name should be minimum {{firstName.errors.minlength.requiredLength}} characters</div>

<div \*ngIf="firstName.errors.maxlength">First Name should be maximum {{firstName.errors.maxlength.requiredLength}} characters</div>

<div \*ngIf="firstName.errors.pattern">First Name doesn't match the pattern</div>

</div>

</div>

<div class="form-group">

<label for="comment">Comment</label>

<textarea ngModel name="comment" id="comment" cols="30" rows="10" class="form-control">

</textarea>

</div>

<button class="btn btn-primary">Submit</button>

</form>

**Styling Invalid Input Fields**

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

body{

padding: 20px;

}

.form-control.ng-touched.ng-invalid{

border: 2px solid red;

}

**Cleaner Templates**

Her satırda bir attribute

**ngForm**

<form #f="ngForm" (ngSubmit)="submit(f)">

submit(f){

//f.value json object

}

**ngModelGroup**



Form sub groups

<form #f="ngForm" (ngSubmit)="submit(f)">

<div ngModelGroup="contact" #contact="ngModelGroup">

<div \*ngIf="!contact.valid"></div>

<div class="form-group">

<label for="firstName">First Name</label>

<input required minlength="3" maxlength="10" pattern="banana" ngModel name="firstName" #firstName="ngModel" (change)="log(firstName)"

id="firstName" type="text" class="form-control">

<div class="alert alert-danger" \*ngIf="firstName.touched && !firstName.valid">

<div \*ngIf="firstName.errors.required">

First Name is required

</div>

<div \*ngIf="firstName.errors.minlength">

First Name should be minimum {{firstName.errors.minlength.requiredLength}} characters

</div>

<div \*ngIf="firstName.errors.maxlength">

First Name should be maximum {{firstName.errors.maxlength.requiredLength}} characters

</div>

<div \*ngIf="firstName.errors.pattern">

First Name doesn't match the pattern

</div>

</div>

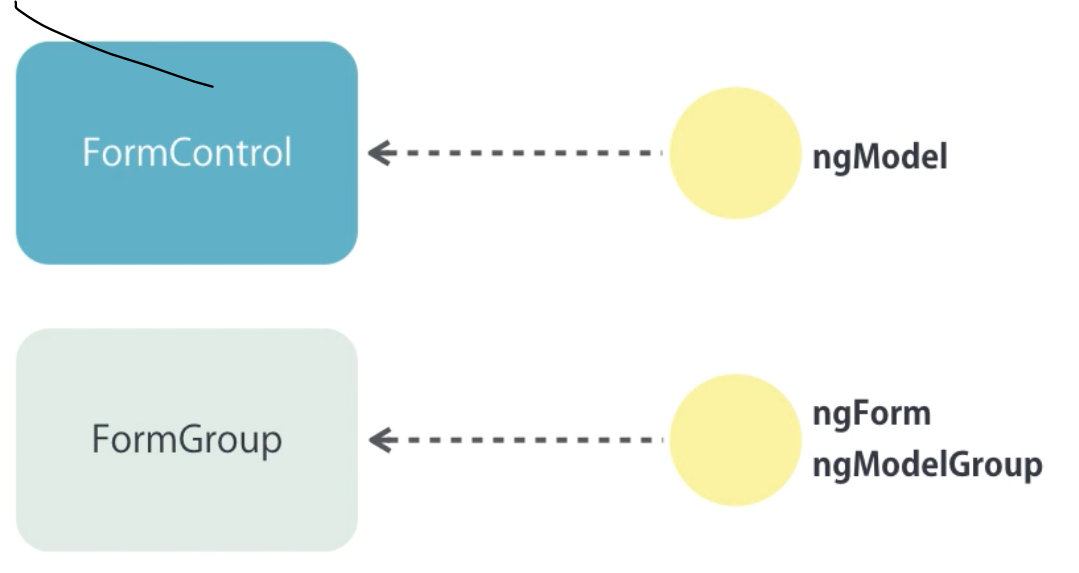
</div>

</div>

**Control Classes and Directives**

**ngForm vs ngModelGroup**

**have submit doesn’t have submit**



**CheckBox**

div.checkbox>labe>input[type='checkbox']

<label>

<input type="checkbox" ngModel name="isSubcribed"> Subscribe to Mailing List

</label>

**DropDownList**

**div.form-group>label+selec.form-control**

**Multiple attribute**

**ngValue obje göndermek için**

<select ngModel name="contactMethod" id="contactMethod" class="form-control">

<!-- <option value=""></option> -->

<option value=""></option>

<!-- <option \*ngFor="let method of contactMethods" [ngValue]="method">{{method.name}}</option> -->

<option \*ngFor="let method of contactMethods" [value]="method.id">{{method.name}}</option>

</select>

**Radio Buttons**

**Only one selection**

div.radio>label>input[type='radio'][nam2= contactMethodra]

<div \*ngFor="let method of contactMethods" class="radio">

<label>

<input ngModel type="radio" name="contactMethodra" [value]="method.id"> {{method.name}}

</label>

</div>

<!-- <div class="radio">

<label>

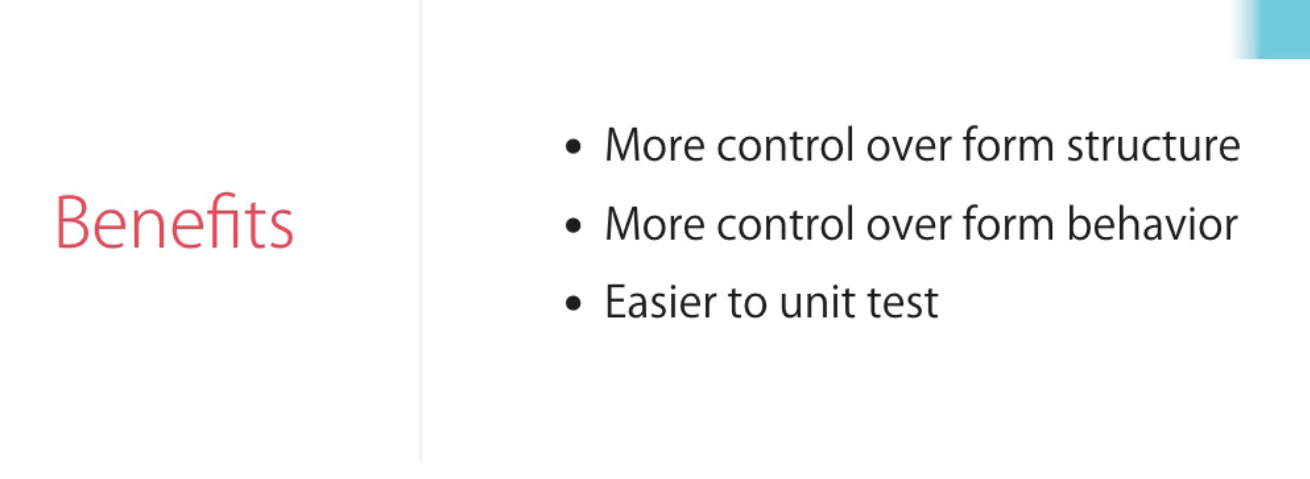
<input ngModel type="radio" name="contactMethodra" value="2"> Phone

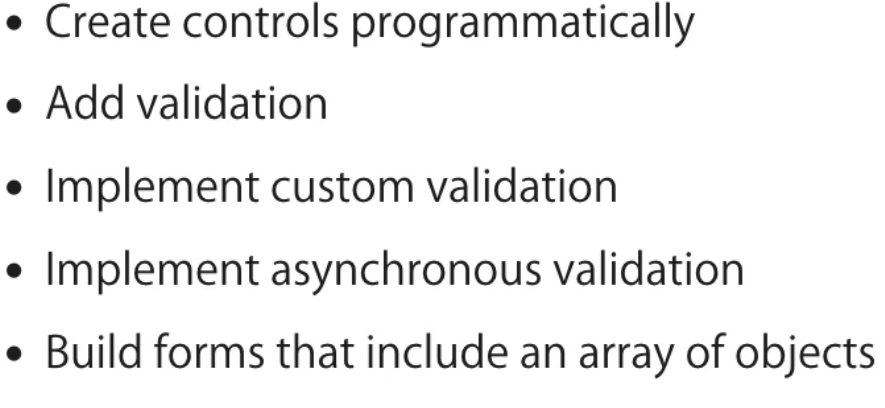
</label>

</div> -->

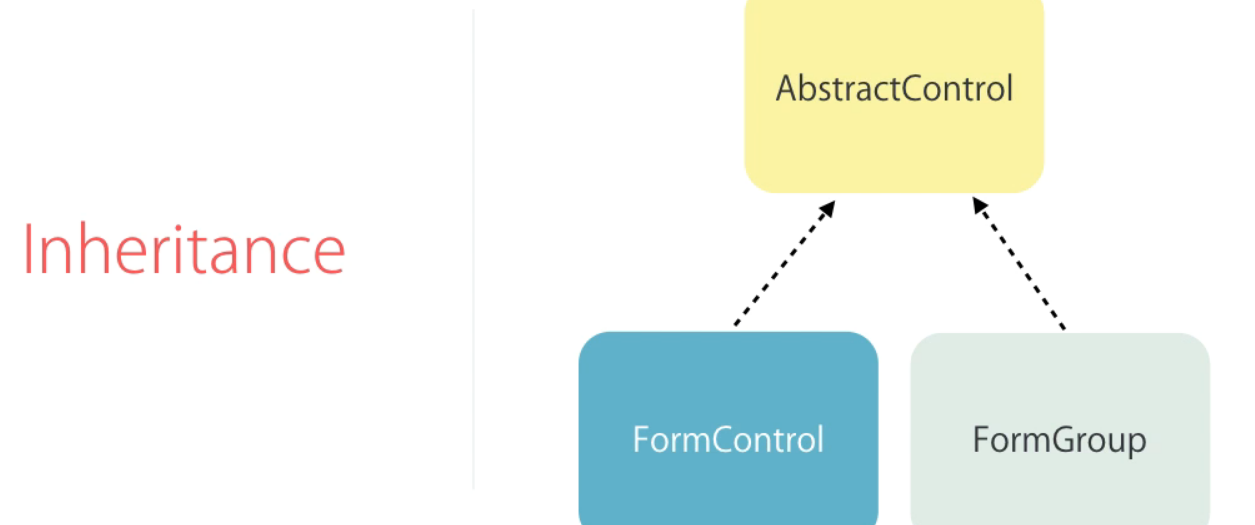
**Reactive Forms**

**Explicitly create control object with code**





**Creating Controls Programatically**



**.ts**

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

import { FormGroup, FormControl } from '@angular/forms';

@Component({

selector: 'signup-form',

templateUrl: './signup-form.component.html',

styleUrls: ['./signup-form.component.css']

})

export class SignupFormComponent {

form=new FormGroup(

{ username:new FormControl(),

password: new FormControl()

}) ;

}

**.html**

<form [formGroup]="form">

<div class="form-group">

<label for="username">Username</label>

<input

formControlName="username"

id="username"

type="text"

class="form-control">

</div>

<div class="form-group">

<label for="password">Password</label>

<input

formControlName="passwords"

id="password"

type="text"

class="form-control">

</div>

<button class="btn btn-primary" type="submit">Sign Up</button>

</form>

**Adding Validation**

Validators class in

form = new FormGroup(

{

username: new FormControl('', Validators.required),

password: new FormControl('', Validators.required)

});

get username() {

return this.form.get('username');

}

<form [formGroup]="form">

<div class="form-group">

<label for="username">Username</label>

<input

formControlName="username"

id="username"

type="text"

class="form-control">

<div \*ngIf="username.touched && username.invalid" class="alert alert-danger">Username is required</div>

</div>

<div class="form-group">

<label for="password">Password</label>

<input

formControlName="passwords"

id="password"

type="text"

class="form-control">

</div>

<button class="btn btn-primary" type="submit">Sign Up</button>

</form>

Custom Valiadation

ValidatorFn => interface

Validaitonerrors

import { AbstractControl, ValidationErrors } from "@angular/forms";

export class UsernameValidators {

static cannotContainSpace(control: AbstractControl): ValidationErrors | null {

if ((control.value as string).indexOf(' ') >= 0) {

return { cannotContainSpace: true };

return null;

/\*

return {minlength: {

reuiredLength:10,

actualLength:control.value.length

}}\*/

}

}

}

Component.ts

export class SignupFormComponent {

form = new FormGroup(

{

username: new FormControl('',

[Validators.required,

Validators.minLength(3),

UsernameValidators.cannotContainSpace

]),

password: new FormControl('', Validators.required)

});

get username() {

return this.form.get('username');

}

}

.html

<div \*ngIf="username.errors.cannotContainSpace">Username cannot contain space</div>

Asynchronous Operations

Asynchronous Validatiors

Validators.ts

static shouldBeUnique(contorl: AbstractControl): Promise<ValidationErrors | null> {

return new Promise((resolve, reject) => {

setTimeout(() => {

{

if (contorl.value === "talip")

resolve({ shoulBeUnique: true });

else resolve(null);

}

}, 2000);

});

}

Component.ts

form = new FormGroup(

{

username: new FormControl('',

[Validators.required,

Validators.minLength(3),

UsernameValidators.cannotContainSpace

],

UsernameValidators.shouldBeUnique

),

password: new FormControl('', Validators.required)

});

<div \*ngIf="username.errors.shoulBeUnique">Username is already taken</div>

Showing a loader Image

<div \*ngIf="username.pending">Checking for Uniqueness</div>

Validaitng the Form upon Submit