1. Nodejs Proje Oluşturma;

npm init -y && npm install express && npm install -D typescript ts-node @types/node @types/express jest ts-jest @types/jest && npx tsc --init && npx ts-jest config:init

npm install express-session  
npm i --save-dev @types/express-session

npm i dotenv

// Angular



https://angular.io/guide/setup-local



// yeni proje başlatma

ng new my-app



// Projeyi çalıştırma

ng serve --open



// Port değiştirerek uygulamayı çalıştırma

ng serve --port 4401



// portu kalıcı olarak değiştirme

"development": {

"browserTarget": "twodays:build:development",

"port": 4200

}



// yaşayan projeyi durdurma

^C



// çalıştırma url'li

http://localhost:4200/



// Npm paketlerini (Bağımlılıkları) yükleme

npm install





// Notasyon

@Component



// export

//bu anahtar kelime bir özelliğin (Class, function, variable) gibi unsurların dışarıdan görünmesini sağlar.



// yeni component kurulumu

ng generate component xyz



// oluşturulan componentin çağrılması

selector: 'app-header',

<app-header></app-header>



// constructor

// sınıfın nesne halinde üretilmesi ile birlikte çalışan ilk methodtur.



// OnInit

// constructor çalıştıkrtan sonra çalışan ve view katmanındaki hazırlıklar bittikten sonra çalışır.



 // değişken kurulu

 // const -> sabit -> sınıfa ait olduğunda tanımlanamaz, method gövdesinde kullanılır.

 // let -> normal değişken ama fonksiyon içinde yazıldığıda globalde kullanım

 // var -> normal değişişken

 header\_title = '';

 // header\_title = ''; // değişken değeri sınıf gövdesinde değişime tabi tutalamaz.



// this -> this.call()

// sınıfa ait değişken yada fonksiyonları işaret ederek bunlara erişimi sağlar.



 // array

// angularda diziler mutable(daha sonradan değiştirilebilir) türdedir

users = ['İstanbul', 'Ankara', 'İzmir', 'Bursa'];

// indexOf -> dizi içerisindeki index değerini bulur

this.users.indexOf('İstanbul');

// splice -> dizi elemanını siler

this.users.splice( indexOf, 1 );



// throw new Error('Hata Mesajı');

// hata olduğunda kendisinden sonra çalışmasını istemediğini ve kullanıcıyı uyarmak istediğiniz durumlarda kullanılır.



// dizi elemanlarının tüketilmesi

// for - Html

<ul>

<li \*ngFor="let item of users"> {{ item }} </li>

</ul>





// html tarafta if kullanımı

<div \*ngIf="item != 'İstanbul'">

{{ item }}

</div>





// component - sınıf tarafında for döngüsü

this.users.forEach( item => {

console.log( item )

})



// Object -> propery class

// Key - val şeklinde bir kullanımı vardır.

 obj = {

title: 'Angular',

age: 30,

status: true

}

<div> {{ obj.title }} </div>





// form işlemleri

// 1. Yöntem

[(ngModel)]="name" ile model data alınır



// 2. Yöntem -> Form Submit ile data gönderimi

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

imports: [

BrowserModule,

AppRoutingModule,

FormsModule,

ReactiveFormsModule

],



app.components.ts içerisine

import { FormBuilder } from '@angular/forms';



 constructor( private formBuilder:FormBuilder ) {}

// formGroup

userForm = this.formBuilder.group({

name: '',

email: ''

})



app.component.html

<form [formGroup]="userForm" (ngSubmit)="fncSend()" method="post">

<input id="name" formControlName="name" placeholder="Name" required />

<input id="email" formControlName="email" placeholder="E-Mail" required />

<input type="submit" value="Send">

</form>







// form validations - start

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



export class AppComponent implements OnInit {



 constructor( private formBuilder:FormBuilder ) {}

ngOnInit(): void {

this.userForm = new FormGroup({

name: new FormControl( this.user.name, [

Validators.required,

]),

email: new FormControl(this.user.email, [

Validators.required,

Validators.email

])

})





 <form [formGroup]="userForm" (ngSubmit)="fncSend()" method="post">

<input class="form-control" id="name" formControlName="name" placeholder="Name" required />

<div \*ngIf="userForm.controls['name'].invalid" >

Lütfen Name Giriniz!

</div>

<input class="form-control" id="email" formControlName="email" placeholder="E-Mail" required />

<div \*ngIf="userForm.controls['email'].invalid" >

Lütfen E-Mail Giriniz!

</div>

<input class="btn btn-danger" type="submit" value="Send">

</form>





 <div \*ngIf="name?.invalid">

Lütfen Name? Giriniz!

</div>





// form validations - end





// setTimeOut -> bir işlemin belirli bir süre sonra tetiklenmesini sağlar, tekrarlayıcı değildir.

setTimeout(() => {

this.error = ''

}, 4000);





// sayfalar arasında geçiş ve veri aktarımı

ng g c settings -> component oluşturma



// sayfa yönlendirme için tanımlar yapılıyor

{path: 'product/:id', component: ProductComponent},

{path: 'settings', component: SettingsComponent}



// Gönderilen dataların yakalanması

import { ActivatedRoute } from '@angular/router';

constructor( private route:ActivatedRoute ) { }



 ngOnInit(): void {

this.route.paramMap.subscribe( params => {

const id = params.get('id')

this.getid = id!

})

}

// Events

// kullanıcı harektelerini dinleyen yapıları kullanmak

<input (keyup)="fncUserValid($event)" class="form-control" placeholder="User E-Mail" />

fncUserValid( evt: KeyboardEvent ): void {

 const data = (<HTMLInputElement> evt.target).value

 this.fncControl(data)

 }

// ElementRef

<input #email (keyup)="fncUserValid($event)" class="form-control" placeholder="User E-Mail" />

@ViewChild('email') emailRef: ElementRef | undefined;

ngAfterViewInit() {

console.log( this.emailRef )

this.emailRef?.nativeElement.focus()

}

// interface kullanımı

// uygulamaya gelen yada uygulama içinde üretilen dataların belli bir kıstasa göre gelmesini sağlayan ve objelere bu yönüyle filtre katan bir araçtır.