**HTML 5**

-- HyperText Markup Language bir web sitesinin nasıl görüneceği ile ilgili kodları barındıran yapıdır. Basit bir tag dilidir. Açılış ve kapanış taglerinin içerisinde kodları yazarız.

-- h tagi heading anlamına gelir. h1, h2, h6 gibi ifadeler puntoyu ifade eder.

-- p tagi paragraf anlamına gelir.

-- br tagi breakten gelir. Bir satır atlamaya yarar.

-- ul tagi unordered list anlamına gelir.

-- ol tagi ordered list anlamına gelir.

-- li tagi list item anlamına gelir. ul veya ol içerisine li tagi açarak içerisine itemlar atabiliriz. ul’de sırasız, ol’de sıralı olarak listeler (nokta ve 1234). Li içerisine başka ol veya ul’ler açabiliriz.

-- a tagi anchore anlamına gelir. tıklayınca başka linklere gönderme işlemi yapar.içerisine hyperlink referance alır. <a href=”index.html” target=”\_blank”> Anasayfa </a> şeklinde kullanılır. index.html yerine web adresi de yazılabilir. targetin içerisindeki \_blank yeni bir sekme aç, var olan sekmeyi kapatma demek.

-- img tagi ile Anasayfa metni yerine bir resim yerleştirilebilir. image içerisine source’u <img width=”75” height=”75” alt=”resim bulunamadı” src=”resimlinki”></img> ile yerleştiririz. alt alternatif anlamına gelmekte olup, resim bulunamazsa kullanıcıya ne gösterileceğini belirler. width ve height ise resim boyutunu belirler.

-- img tagi link dışında da kullanılabilir.

-- table tagi ile tablo oluştururuz. tr table row, içerisine aldığı td ise table dimension ise hücreyi kast eder.

-- caption tagi ile tabloya başlık ekleyebiliriz.

-- b tagi bold anlamına gelir. içerisine yazılan şeyleri kalın olarak gösterir.

-- strong tagi de bold ile aynı işlemi yapar. genelde bu tag kullanılır.

-- em tagi emphasis vurgulamak demektir. içerisine yazılan şeyleri italik olarak gösterir. i tagi de kullanılabilir.

-- small tagi içerisine yazılanlar biraz daha küçük formatta gözükür.

-- sub tagi içerisine yazılanları biraz daha aşağıda gösterir.

-- sup tagi içerisine yazılanları biraz daha yukarıda gösterir.

-- ins tagi içerisine yazılanları altı çizili olarak gösterir.

-- del tagi içerisine yazılanları üzeri çizili olarak gösterir.

-- mark tagi içerisine yazılanları fosforlu kalemle üzeri çizilmiş gibi gösterir.

Layout

-- header tagi sayfanın başlığını alır.

-- nav tagi navigation anlamına gelir. içerisine genelde linkler eklenir.

-- article tagi ana kısmın başlığıdır.

-- footer sayfanın en alt kısmının tagidir.

Form tagleri

<form action="gideceğiyerburaya.html">

    <label> Username: </label>

    <input name:"username" type="text" required/>

    <br>

    <label> Password: </label>

    <input name:"password" type="password" required/>

    <br>

    <label> Cinsiyet: </label>

    <select name="gender" size=10 multiple>

      <option> Erkek </option>

      <option> Kadın </option>

    </select>

    <br>

    <label> Hakkınızda: </label>

    <textarea name="biography" rows="5" cols="50"></textarea>

    <br>

    <input type="submit" value="Defaul değeri buradan değiştirebiliriz">

    <br>

    <button onclick=alert('Kaydedildi')> Kaydet </button>

  </form>

**CSS**

-- Stillerimizi style.css dosyasına eklemek en kullanışlı yöntemdir.

h1{

  color: red;

}

body{

  background-color: gold;

}

-- Kodlar indexte style tagi içerisine de yazılabilir veya h1 içerisine direkt yazılabilir ama önerilmez.

  <style>

    h1{

  color: red;

}

body{

  background-color: gold;

}

    </style>

<h1 style="color:red" >Hello Plunker!</h1>

-- Style için birçok özellik bulunur. Ortak durumlarda div,table{} gibi kullanımlar yapılabilir. Kendisine has olanı da ayrı bir yerde yazabiliriz.

  <style>

    div{

      /\* Border çerçeve ekler. px sayısı kalınlığı belirler.

         solid çerçeve türüdür. green ise çerçeve rengi.

         margin div'in üstündeki elementten uzaklığını ayarlar.

         padding ise çerçeveye olan uzaklığını ayarlar. \*/

      border:2px solid green;

      margin-top:100px;

      margin-left:100px;

      margin-right:100px;

      padding-top:100px;

      padding-bottom:100px;

      padding-right:100px;

      padding-left:100px;

    }

    </style>

-- border-collapse:collapse; ile borderların çizgileri birleştirilebilir.

-- width, height gibi özelliklerde px haricinde %100 gibi değerler kullanılabilir.

-- text-align:right; metni sağa dayar.

-- vertical-align:top; metni yukarıya yaklaştırır.

-- table:hover{ background-color:yellow;} mouse ile tablo üzerine gelindiğinde arka planı sarı olur.

-- table:nth-child(odd){background-color:red;) tablodaki tek sayı sırasına denk gelenlerin arkası kırmızı olur.

-- input[type=text], input[type=password]{} şeklinde sadece text ve password tipindeki inputlara özel style yazılabilir.

-- border-radius:3px; border köşelerini yuvarlaklaştırır.

-- border-bottom:3px çerçevenin alt kısmının px ayarını yapar.

-- input[type=text]:focus , input[type=password]{ background-color:red;} text tipindeki input alanlarının arka planı tıklayınca kırmızı olur. password tipine focus eventi eklemediğimiz için o aynı kalır.

-- .ClassAdı{backgroun vs özellikleri} ile class oluşturulur. <h1 class=”ClassAdı”> İçerik</h1>

şeklinde class özellikleri verilebilir.

-- h.ClassAdı olarak oluşturulsaydı bu class sadece h taglerinde kullanılabilirdi.

-- #idİsmi{} şeklinde id oluşturulur. Classlarla aynı sayılır. Genelde classlar kullanılır.

**BOOTSTRAP**

 <head>

        <!-- bootsrapi indirerek projeye ekleyebilir veya documentation sayfasından ekleme kodunu kopyalayarak projeye ekleyebiliriz. genelde indirilerek eklenir. -->

        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z" crossorigin="anonymous">

    </head>

<!-- Kullanıcıyla iletişime geçme işini javascript yapar. Sayfanın sağındaki üç çizginin çalışması için javascript kodlarını ekledik. -->

    <!-- Bu kodların sıralı olması js için önemlidir. Kodları documentation sayfasından çekip, ekledim.-->

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>

    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>

    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>

-- Body kısmında not olmadığı için eklenmedi.

Conteiner

<html>

  <head>

    <link

      rel="stylesheet"

      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"

      integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"

      crossorigin="anonymous"

    />

  </head>

  <body>

    <div class="jumbotron text-center">

      <h1>Web Sayfama Hoş Geldiniz!</h1>

      <p>Burada jumbotron altına yazı yazdık.</p>

    </div>

    <!-- conteiner-fluid kolonları sola yaslı şekilde oluşturur.-->

    <!-- col-md-4 varsayılan değerdir. yanına eklenen değere göre column şekil alır.-->

    <div class="conteiner">

      <div class="row">

        <div class="col-md-2">Kolon 1</div>

        <div class="col-md-4">Kolon 2</div>

        <div class="col">Kolon 3</div>

      </div>

    </div>

  </body>

</html>

Tables

<html>

  <head>

    <link

      rel="stylesheet"

      href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"

      integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"

      crossorigin="anonymous"

    />

  </head>

  <body>

      <!-- Bootstrap içerisindeki table class'ını tablomuza ekleyince adama benzedi. -->

      <!-- table-striped class'ı ile de bir satır gri bir satır beyaz yaptık. -->

      <!-- table-bordered class'ı ile tablomuza çerçeve ekledik. -->

      <!-- table-hover class'ı ile tablomuz üzerinde mouse'umuzu nereye koyarsak orası farklı görünecek. -->

      <!--<table class="table table-striped table-bordered table-hover">-->

    <!-- table-dark ile arka planı karanlık bir tablo oluşturduk. -->

    <!-- table-borderless ile ürünler arasındaki alt çizgiyi kaldırdık. -->

    <table class="table table-dark table-borderless">

      <thead>

        <tr>

          <td>Id</td>

          <td>Ürün Adı</td>

          <td>Kategori</td>

          <td>Fiyat</td>

          <td>Stok Adedi</td>

        </tr>

      </thead>

      <tbody>

          <!-- table-danger table-warning table-success gibi classlarla satırlarımızı renklendirebiliriz. -->

          <!-- bootstrap sayfasından başka özellikleri görüntüleyebilirsin. -->

        <tr class="table-danger">

          <td>1</td>

          <td>Monster Laptop</td>

          <td>Elektronik</td>

          <td>5000</td>

          <td>14</td>

        </tr>

        <tr>

          <td>2</td>

          <td>Hp Laptop</td>

          <td>Elektronik</td>

          <td>3000</td>

          <td>10</td>

        </tr>

        <tr>

          <td>3</td>

          <td>Asus Laptop</td>

          <td>Elektronik</td>

          <td>4500</td>

          <td>8</td>

        </tr>

      </tbody>

    </table>

  </body>

</html>

Dropdown

body>

    <!-- dropright classı dropdown'ın butonun sağından aşağıya doğru açılmasını sağlar. -->

    <!-- dropup classı da bulunuyor. -->

    <div class="dropdown dropright">

      <!-- dropdown-toggle class'ı ile menümüzü açılır, kapanır hale getirdik. -->

      <!-- dropdown kullanıcı ile etkileşime geçtiği için js'ye ihtiyaç duyar. -->

      <!-- data-toogle'ı dropdown olarak belirledik ki menuye bastığımızda dropdown olarak açılsın. -->

      <button

        type="button"

        class="btn btn-success dropdown-toggle"

        data-toggle="dropdown"

      >

        Menu

      </button>

      <!-- yeni bir dropdown-menu divi oluşturduk. -->

      <div class="dropdown-menu">

        <!-- Menu başına hafif gri yazı ekler. -->

        <div class="dropdown-header">Sepet Elemanları</div>

        <!-- menu içerisindeki itemlari genelde a tagi ile yazarız ama farketmez. -->

        <!-- item olmaları için dropdown-item class'ını almaları gerekir. Yoksa tıklanmaz. -->

        <!-- itemlara active passive classlarını ekleyerek kontrol edebiliriz. -->

        <a class="dropdown-item"> Item 1 </a>

        <a class="dropdown-item"> Item 2 </a>

        <a class="dropdown-item"> Item 3 </a>

        <!-- itemlar arasına çizgi çeker. -->

        <div class="dropdown-divider"></div>

        <a class="dropdown-item"> Satın Al </a>

      </div>

    </div>

  </body>

  <div class="btn-group">

<!-- 2 buton oluşturup, birini sepete birini menüye yönlendirebiliriz. -->

  <div class="btn-group">

      <button class="btn btn-success"> Sepete Git </button>

      <button

        type="button"

        class="btn btn-success dropdown-toggle dropdown-split"

        data-toggle="dropdown"

Forms and Inputs

  <body>

    <!-- class container ile sağdan soldan boşluklar bırakarak hoş bir görüntü oluştururuz.-->

    <!-- form-control classı inputları güzel bir görüntüye kavuşturur. -m -s -lg ile boyutu ayarlanabilir. -->

    <div class="container">

      <div class="col-md-6">

        <form>

          <div class="form-group">

            <label for="firstName">Ad:</label>

            <!-- placeholder inputun içerisine geçici yazı koyar. -->

            <input

              type="text"

              class="form-control"

              id="firstName"

              placeholder="Ad"

            />

          </div>

          <div class="form-group">

            <label for="lastName">Email:</label>

            <!-- placeholder inputun içerisine geçici yazı koyar. -->

            <input

              type="text"

              class="form-control"

              id="lastName"

              placeholder="Soyad"

            />

          </div>

          <div class="form-group">

            <label for="email">Email:</label>

            <!-- placeholder inputun içerisine geçici yazı koyar. -->

            <input

              type="email"

              class="form-control"

              id="email"

              placeholder="Email"

            />

          </div>

          <div class="form-group">

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

            <!-- placeholder inputun içerisine geçici yazı koyar. -->

            <input

              type="email"

              class="form-control"

              id="password"

              placeholder="Password"

            />

          </div>

          <div class="form-group">

            <label for="cinsiyet">Cinsiyet:</label>

            <select class="form-control" id="cinsiyet">

              <option>Erkek</option>

              <option>Kadın</option>

            </select>

          </div>

          <div class="form-group">

            <label for="bio">Hakkınızda:</label>

            <textarea rows="3" id="bio" class="form-control"></textarea>

          </div>

          <!-- checkboxları form-check classı ile ekliyoruz. inputlarını ise form-check-input ile eklenir.-->

          <div class="form-check">

            <label class="form-check-label" for="terms"

              ><input type="checkbox" class="form-check-input" id="terms" />I

              Accept The Terms And Conditions

            </label>

          </div>

          <div class="form-check">

            <label class="form-check-label" name="radio" for="radio"

              ><input

                name="options"

                type="radio"

                class="form-check-input"

                id="radio"

              />Option 1

            </label>

          </div>

          <div class="form-check">

            <label class="form-check-label" name="radio" for="radio"

              ><input

                name="options"

                type="radio"

                class="form-check-input"

                id="radio"

              />Option 2

            </label>

          </div>

          <div>

            <button type="submit" class="btn btn-success">Kaydet</button>

          </div>

        </form>

      </div>

    </div>

  </body>

Lists

<html>

  <link

    rel="stylesheet"

    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"

    integrity="sha384-JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"

    crossorigin="anonymous"

  />

  <body>

    <div class="container">

      <div class="col-md-3">

        <ul class="list-group">

          <li class="list-group-item">Furkan Keskin</li>

          <li class="list-group-item">Ayşe Keskin</li>

          <li class="list-group-item">Mehmet Keskin</li>

        </ul>

      </div>

    </div>

    <div class="container">

        <div class="col-md-3">

          <ul class="list-group">

              <!-- a tagi kullandık ve içerisine link yerleştirdik. -->

              <!-- linkler mavi gözüktüğü için list-group-item-action eklersek daha güzel bir görüntü elde ederiz. -->

              <!-- stilleri burada da kullanabiliriz. -->

            <a href="#" class="list-group-item list-group-item-action list-group-item-warning">Furkan Keskin</a>

            <a href="#" class="list-group-item list-group-item-action list-group-item-primary">Ayşe Keskin</a>

            <a href="#" class="list-group-item list-group-item-action list-group-item-dark">Mehmet Keskin</a>

          </ul>

        </div>

      </div>

  </body>

</html>

JAVASCRIPT ES6+

-- prompt(“Adınız:”) ile input alınır.

-- Number(stringAdı) ile int’e çevirme işlemi yapılır.

-- console.log ile print işlemi yapılır.

-- function fonksiyonAdı(){} ile fonksiyon oluşturulur.

-- js’de fonksiyona parametre gönderilmezse parametreleri undefined olarak alır.

-- 2+”b” sonucu “2b” olarak verir.

-- arrayAdı.pop() dizinin son elemanını verir ve diziden onu siler.

-- arrayAdı.shift() dizinin ilk elemanını verir ve onu diziden siler.

-- arrayAdı.push(“Eleman”) ile diziye eleman eklenir.

-- arrayAdı.concat([“istanbul”,”Ankara”]) arrayle başka elemanlar eklenip, sonuç verilir.

Kalıcı değildir.

-- arrayAdı.sort() ile dizi sıralanır.

-- arrayAdı.lenght ile uzunluk alınır.

-- var Product={Name:”Ürün1”, Price:100} şeklinde newlemeden nesne tanımlayabiliriz.

-- nesne içerisine içerisinde nesne bulunan diziler tutabiliriz. nesneler içerisinde fonksiyon da tutabiliriz.

-- arrayAdı.forEach(function(x)) şeklinde foreach kullanımı yapılabilir.

Dom Manipulasyonu

dom.html

<html>

  <head>

  </head>

  <body>

    <p id="metin"></p>

  </body>

  <!-- Scripti en alta yazdık. Sayfanın başında olsaydı metin isimli elementi bulamayıp, hata verecekti. -->

  <script src="dom.js"></script>

</html>

dom.js

// getElementById ile id'sine göre elementi aldık.

// Daha sonra onu manipüle ederek içerisine metin yerleştirdik.

document.getElementById("metin").innerHTML="Furkan Keskin"

// üstteki kodu herhangi bir değişkene de atayabiliriz.

// innerHTML koduyla elementin içerisine ulaşırız.

// getElementByTagName, getElementByClassName, getElementByName gibi fonksiyonlar da bulunur.

-- querySelectorAll(“p.intro1”) şeklinde tagAdı.ClassAdı formatında sadece o tag’in içerisindeki belirli classları alabiliriz.

-- getsElementsByName(“ElementAdı”) şeklinde name değeriyle elementleri çekebiliriz.

-- document.getElementById(“elementAdı”).addEventListener(“eventAdı”, gerçekleşecek işlem veya fonksiyon) şeklinde eventlere göre işlem belirlenebilir. event listesini webden bulabilirsin.

-- elementi değişkene atayıp, değişkenAdı.childNodes[0].nodeValue ile içerisindeki yazıya ulaşabiliriz.

Node.html

<html>

  <head> </head>

  <body id="body1">

    <div id="div1">

      <p id="p1">Furkan Keskin</p>

      <p id="p2">BURADA</p>

    </div>

  </body>

  <script src="node.js"></script>

</html>

Node.js

// createElement ile bir element yaratabiliriz.

var baslik=document.createElement("h1");

// createTextNode ile element içerisine ekleyeceğimiz texti oluşturduk.

var node=document.createTextNode("Merhaba Javascript");

// appendChild komutu ile node'umuzu oluşturduğumuz elemente ekledik.

baslik.appendChild(node);

// Div1 ve p2 elementlerini çektik ve ayrı ayrı değişkenlere atadık.

var div1=document.getElementById("div1");

var p2=document.getElementById("p2");

// div1'e p2'den önce baslik'i ekle dedik.

div1.insertBefore(baslik,p2);

// div1'in altından p2 elementini sildik.

alert("P2 Elementi Siliniyor!");

div1.removeChild(p2);

// replaceChild ile p1 yerine basliki koyduk.

// Ekranda sadece Merhaba Javascript gözükecek.

var p1=document.getElementById("p1");

alert("P1 Degistiriliyor!")

div1.replaceChild(baslik,p1)

ECMASCRIPT

-- var ile tanımlanan değişkenler daha sonra tekrar var ile tanımlanabilir.

-- let ile tanımlanan değişkenler daha sonra tekrar let ile tanımlanamaz.

-- for içerisindeki i’yi var ile tanımlarsan döngünün son halindeki i’ye dışardan ulaşabilirsin. Let ile tanımlarsan ulaşamazsın.

-- const ile değişken tanımlarsak asla değişmez.

-- const ile referans tip üzerinde ekleme yapılabilir ama yeniden tanımlama yapılamaz.

// İleri Fonksiyonlar

// Fonksiyonları değişkenlere atabiliriz.

// Bunu yaparsak fonksiyonu fonksiyonAdı ile değil değişken adıyla çağırmak zorundayız.

var selamFonksiyonu1=function selam(){

    console.log("Selam 1")

}

const selamFonksiyonu2=()=>{

    console.log("Selam 2");

}

const selamFonksiyonu3=(parametre)=>{

    console.log("Selam 3");

}

const selamFonksiyonu4=()=>console.log("Selam 4");

// Dizi tanımı

const dizi=[1,2,3,4];

// Map fonksiyonu

// Dizinin her elemanını tek tek gezer.

const mapDizi = dizi.map(sayi => sayi\*3);

console.log(mapDizi);

// Filter fonksiyonu

const filtreliDizi = dizi.filter(sayi => sayi>2);

console.log(filtreliDizi);

// Reduce fonksiyonu

// Toplama işlemi yapar

// acc toplam değerini tutar.

// 5 ise acc'nin default değeri. Boş kalırsa 0 olur.

const toplamDizi = dizi.reduce((acc, sayi)=>{

return acc+sayi;

},5);

console.log(toplamDizi)

-- Classlar class classAdı ile tanımlanır.

-- Constructorlar ise constructor(parametreler){} şeklinde tanımlanır.

**TYPESCRIPT**

-- Veri tipleri number, string, boolean, array, tuple, enum’dır.

-- Değişkenler let değişkenAdı:değişkenTürü = verilecekse Değer şeklinde tanımlanır.

-- Arrayler let arrayAdı:arrayTipi[] = [atanacaksaDeğerler] şeklinde tanımlanır.

-- Tipe özel arrayler let arrayAdı:Array<Customer> = <atanacaksaDeğerler] şeklinde atanır.

-- Tuple’lar let arrayAdı:[arrayTipi1,arrayTipi2] = [2,”Ankara”] şeklinde tanımlanabilir.

-- enum Renk{Kirmizi=1, Siyah, Mavi}

let renk : Renk = Renk.Kirmizi şeklinde kullanımlar yapılabilir. Böylece kirmizinin değeri 3543 olsa bile tek bir noktadan değiştirerek sorun çözülebilir.

-- Bunların dışında any, void, null ve undefined vardır.

-- let değer : any = “İçerisine herhangi bir şeyi alabilir”

-- fonksiyon içerisindeki parametreye y:number=4 şeklinde default değer atayabiliriz.

-- parametreyi y?:number şeklinde yazarsak y değeri verilmeden de fonksiyon çalışır. y’yi undefined olarak alınır. Ona göre çalışılmalı.

-- …davetliler:string[] şeklinde sınırsız parametre alabiliriz. return ederken return davetliler.join(“ “) şeklinde return ediyoruz.

-- inheritance işlemi extends keywordu ile yapılır.

-- classları newleyerek kullanırız.

-- private, public, protected typescripte de geçerli.

-- get isim():string{

return “Sayın” + this.\_isim;

}

set isim(ad:string){

this.\_isim = ad;

}

get ve set olmadan da atama yapılabilir. Fakat get ve set ile ekstra işler yapılabilir.

-- for(let i in sehirler) 0,1,2 gibi index değerlerini verirken for(let i of sehirler) Ankara, İstanbul, Kastamonu gibi data değerlerini verir.

**ANGULAR**

-- ng serve --open ile projemizi yayına alırız.

-- src altındaki styles.css global bir css dosyasıdır. bu kullanılabilir veya her bileşene özel css dosyası kullanılabilir.

-- cmd’den istenilen klasörü seçtikten sonra ng new projeAdı şeklinde proje oluşturulur.

Components

-- Component html ve js dosyalarından oluşur.

-- ng g component product ile projeye başka componentler eklenebilir.

app.component.ts

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

// selector tag adıdır. index html'de bu kısmı <app-root></app-root> tagi ile çalıştırırız.

// templateUrl bu sayfadaki bileşenleri kullanacağımız sayfadır. aynı klasörde oldukları için ./ ile yerini gösterdik.

// styleUrls css ayarlarının bulunduğu dosyadır.

// @Component dekaratörü ile component tanımlarız.

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

// App component adında export(public) bir class oluşturduk.

// İçerisine de bu classın propertylerini yazdık.

// Bu propertyler {{componentAdı}} şeklinde kullanılabiliyor.

export class AppComponent {

  title = 'intro';

  name = "Furkan";

}

app.component.html

<div style="text-align:center">

  <!-- {{bu şekilde yazıma binding denir.}}-->

  <!-- Şu anki kullanım one way binding'tir.-->

  <!-- Ancak isim kutusuna base değer verip, kullanıcının değiştirdiği değeri de alabiliyorsak bu two way binding'tir.-->

  <h1>Welcome to {{title}} {{name}}!</h1>

</div>

<app-product></app-product>

  <router-outlet></router-outlet>

product.component.ts

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

@Component({

  selector: 'app-product',

  // templateUrl'e html dosyası vermek yerine `` işaretleri arasında html içerisinde yer alan kodları yazabiliriz.

  // buna inline template denir. performans olarak artısı olsa da bu gözle görülür değildir. html dosyasına yazmak daha iyi olur.

  template: `<p>{{ product }} works!</p> `,

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

})

export class ProductComponent implements OnInit {

  constructor() {}

  product = 'Laptop';

  ngOnInit(): void {}

}

app.module.ts

import { BrowserModule } from '@angular/platform-browser';

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

import { AppRoutingModule } from './app-routing.module';

import { AppComponent } from './app.component';

import { ProductComponent } from './product/product.component';

// Modül tanımı

@NgModule({

  // Modül içerisine eklenen componentler.

  declarations: [

    AppComponent,

    ProductComponent

  ],

  // Modüle eklenen diğer modüller.

  imports: [

    BrowserModule,

    AppRoutingModule

  ],

  providers: [],

  // Modül'ün başlangıç componenti.

  // Yani diğer tüm componentler burada.

  bootstrap: [AppComponent]

})

// Export(Public) şekilde modül class'ı.

export class AppModule { }

main.ts

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

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

  enableProdMode();

}

// Ana modülü belirttiğimiz yer.

platformBrowserDynamic().bootstrapModule(AppModule)

  .catch(err => console.error(err));

styles.css

/\* Projenin genelinin stili burada belirlenir. \*/

/\* @import işlemiyle yüklediğimiz paketleri import edebiliriz. \*/

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

@import "~font-awesome/css/font-awesome.min.css";

Paket Yönetimi NPM ile çalışmak

-- paketler package.json içerisinde bulunur.

-- içerisindeki dependencies projenin çalışması için gerekli olan paketleri barındırırken, devdependencies geliştiriciler için gerekli olan paketleri barındırır.

-- örneğin projeye bootstrap eklemek için; bootstrap projenin görüntüsünü etkilediği için dependencies altına "bootstrap":"4.5.2" yazısını ekleyip, kaydettikten sonra konsola npm install yazmalıyız.

-- nmp install package.json’daki paketleri node\_modules klasöründen kontrol eder, eksiklik veya farklılık varsa yükler.

-- npm install font-awesome şeklinde konsoldan da direkt paket yüklemesi yapabiliriz.

-- bootstrap’i kullanabilmek için styles.css’e import etmeliyiz.

ngIf-Else

<!-- \*ngIf ile products nesnesi yoksa bu başlığı gösterme diyebiliyoruz.-->

<!-- product.component.ts'de products nesneleri oluşturduğumuz için şu an başlık gözükür. -->

<!-- ; ile ayırıp else koyabiliriz. noProduct else durumunda çağırılacak template'in adı-->

<h3 \*ngIf="products; else noProduct">{{ title }}</h3>

<ng-template #noProduct> <div class="alert alert-primary" role="alert">

  Bu Kategoride Ürün Bulunmamaktadır!

</div></ng-template>

ngFor

<!-- \*ngFor ile döngü şeklinde products içerisindeki her bir ürün için ürün çerçevesi oluşturduk. -->

  <div \*ngFor="let product of products" class="card" style="width: 18rem">

    <img [src]="product.imageUrl" class="card-img-top" alt="..." />

    <div class="card-body">

      <h5 class="card-title">{{product.name}} {{product.price}} ₺</h5>

      <p class="card-text">

        {{product.description}}

      </p>

      <a href="#" class="btn btn-primary">Ürünü Görüntüle</a>

    </div>

  </div>

PIPE MİMARİSİ

-- ng g pipe pipeAdı şeklinde custom pipe oluşturulabilir.

 <!-- | ile para birimini verdik, sembol gösterme dedik ve 1.2 formatında virgülden sonra 2 basamak olacak şekilde price'ı yazdırdık. -->

      <h5 class="card-title">{{product.name | uppercase}} {{product.price | currency:'₺':false:"1.2-2"}} </h5>

Event Binding

      <!-- Butona click eventi gerçekleşince product.component.ts'den çağıracağı fonksiyonu verdik. -->

      <a (click)="addToCart(product)" class="btn btn-primary">Sepete Ekle</a>

  // Sepete ekleme fonksiyonu

  addToCart(product){

    alert("Sepete " + product.name + " Eklendi!")

  }

AlertifyJS

-- Yeni bir js eklenince sistem yeniden build edilmeli.

-- Konsola npm install alertifyjs yazarak paketi yükledik.

-- angular.json scripts altına "./node\_modules/alertifyjs/build/alertify.min.js" yolunu tanıtarak ekledik.

-- kendisini ve kendisine özel bootstrap dosyalarını kullanabilmek için

@import "~alertifyjs/build/css/alertify.min.css";

@import "~alertifyjs/build/css/themes/bootstrap.min.css"

satırlarını styles.css’e ekledik.

-- product.component.ts’ye declare let alertify:any; koduyla import ettik.

-- alertify.success("Sepete " + product.name + " Eklendi!") şeklinde fonksiyon altında kullandık.

Services And Dependency Injection Mimarisiyle Çalışmak

-- ng g service alertify şeklinde servis oluşturulur.

-- Global servis tanımı bu şekildedir. Fakat componente özel local servis tanımlanacaksa provider’ı silip, kullanılacağı componente providers:[ServisAdı, ServisAdı2] şeklinde instance oluştururuz. Dependency injection’ı otomatik olarak angular yapar.

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

declare let alertify:any;

// providedIn satırı yerine app.module.ts'de providers içerisine AlertifyService de ekleyebiliriz. Aynı şey.

@Injectable({

  providedIn: 'root'

})

export class AlertifyService {

  constructor() { }

  success(message:string){

    alertify.success(message);

  }

  error(message:string){

    alertify.error(message);

  }

  warning(message:string){

    alertify.warning(message);

  }

}

HTTP, JSON-SERVER VE OBSERVABLE MİMARİSİYLE ÇALIŞMAK

-- npm install -g json-server diyerek json server’ı kurduk.

-- json dosyamızın konumuna gittikten sonra json-server --watch dosyaAdı.json diyerek server’da json dosyamızı yayınlıyoruz.

-- [localhost:3000/products?name=Laptop](http://localhost:3000/products?name=Laptop) şeklinde filtreli sorgu yapılabilir.

-- // Json dosyalarımızı çekebilmek için http paketini kullandık.

export class ProductComponent implements OnInit {

  constructor(private alertifyService: AlertifyService, private http:HttpClient) {}

  title = 'Ürün Listesi';

  filterText = '';

  products: Product[];

  path="http://localhost:3000/products"

  // Linkteki json'u product listesi türünde alıp, subscribe komutuyla linke abone olduk.

  // Abone olduğumuz datayı ise products listemize eşitledik.

  // Burayı servis haline getirip, kullanmak daha mantıklı.

  ngOnInit(): void {

    this.http.get<Product[]>(this.path).subscribe(data=> {

      this.products=data;

    });

}

-- Servis haline getirme

@Injectable()

export class ProductService {

  constructor(private http:HttpClient) { }

  path="http://localhost:3000/products"

  // Observable sayesinde fonksiyon çağırılınca işlem bitmiyor, devam ediyor.

  // Bu sayede subscribe komutunu kullanabileceğiz.

  getProducts():Observable<Product[]>{

    return this.http.get<Product[]>(this.path);

  }

}

export class ProductComponent implements OnInit {

  constructor(private alertifyService: AlertifyService, private productService:ProductService) {}

  title = 'Ürün Listesi';

  filterText = '';

  products: Product[];

  // Linkteki json'u product listesi türünde alıp, subscribe komutuyla linke abone olduk.

  // Abone olduğumuz datayı ise products listemize eşitledik.

  ngOnInit(): void {

      this.productService.getProducts().subscribe(data=>{

        this.products=data;

      });

    };

Loglama ve Hata Yakalama

@Injectable()

export class ProductService {

  constructor(private http:HttpClient) { }

  path="http://localhost:3000/products"

  // Observable sayesinde fonksiyon çağırılınca işlem bitmiyor, devam ediyor.

  // Bu sayede subscribe komutunu kullanabileceğiz.

  getProducts():Observable<Product[]>{

    return this.http.get<Product[]>(this.path).pipe(

      tap(data=>console.log(JSON.stringify(data))),

      catchError(this.handleError)

    );

  }

  handleError(err: HttpErrorResponse)  {

    let errorMessage=''

    if(err.error instanceof ErrorEvent){

      errorMessage='Bir Hata Oluştu:'+err.error.message;

    }else{

      errorMessage='Sistemde Bir Hata Oluştu!'

    }

    return throwError(errorMessage);

  }

}

Routing Mimarisiyle Çalışmak

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

import { Routes, RouterModule } from '@angular/router';

import { ProductComponent } from './product/product.component';

// Routlar burada tanımlanır.

const routes: Routes = [

  {path:'products', component:ProductComponent},

  {path:'',redirectTo:'products', pathMatch:'full'},

  {path:'products/category/:categoryId', component:ProductComponent}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

 // Route'a göre ürün getirme.

  getProducts(categoryId):Observable<Product[]>{

    let newPath=this.path;

    if(categoryId){

      newPath+="?categoryId="+categoryId

    }

    return this.http.get<Product[]>(newPath).pipe(

      tap(data=>console.log(JSON.stringify(data))),

      catchError(this.handleError)

    );

  }

Template Driven Forms / Klasik Formlar

-- ng g component product-add-forms1 diyerek form componentimizi oluşturduk.

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

import { Product } from '../product';

import { CategoryService } from 'src/app/services/category.service';

import { Category } from 'src/app/category/category';

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

import { ProductService } from 'src/app/services/product.service';

import { AlertifyService } from 'src/app/services/alertify.service';

@Component({

  selector: 'app-product-add-forms1',

  templateUrl: './product-add-forms1.component.html',

  styleUrls: ['./product-add-forms1.component.css'],

  providers:[CategoryService, ProductService]

})

export class ProductAddForms1Component implements OnInit {

  // Açılır kutuda kategori adlarını getirmek için categoryService'in instance'ını oluşturduk.

  constructor(private categoryService:CategoryService, private productService:ProductService, private alertifyService:AlertifyService) { }

  // Product instance'ı oluşturduk ki bunun içerisine ekleyeceğimiz ürünü atabilelim.

  // Bu componentin html dosyasında input olarak model'in propertylerini alacağız.

  model:Product=new Product();

  //Formdaki select taginin optionlarını buradan çekeceğiz.

  categories:Category[];

  // ngOnInit ile kategorileri çektik.

  ngOnInit(): void {

    this.categoryService.getCategories().subscribe(data=>{

      this.categories=data;

    });

  };

  // Formdaki submit eventine bu fonksiyonu atayacağız.

  add(form:NgForm){

    this.productService.addProduct(this.model).subscribe(date=>{

      this.alertifyService.success("Ürün Başarıyla Eklendi")

    });

  }

}

Reactive Forms

// Validation işlemlerini bu sayfada yaptık.

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

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

import { Product } from '../product';

import { Category } from 'src/app/category/category';

import { CategoryService } from 'src/app/services/category.service';

import { ProductService } from 'src/app/services/product.service';

import { AlertifyService } from 'src/app/services/alertify.service';

@Component({

  selector: 'app-product-add-forms2',

  templateUrl: './product-add-forms2.component.html',

  styleUrls: ['./product-add-forms2.component.css'],

  providers: [CategoryService, ProductService],

})

export class ProductAddForms2Component implements OnInit {

  // FormBuilder ile validation işlemlerini yapacağız.

  constructor(

    private formBuilder: FormBuilder,

    private categoryService: CategoryService,

    private productService: ProductService,

    private alertifyService: AlertifyService

  ) {}

  productAddForm: FormGroup;

  product: Product = new Product();

  categories: Category[];

  createProductAddForm() {

    this.productAddForm = this.formBuilder.group({

      // "" arasına yazılan yazı base değer olur.

      name: ['', Validators.required],

      description: ['', Validators.required],

      imageUrl: ['', Validators.required],

      price: ['', Validators.required],

      categoryId: ['', Validators.required],

    });

  }

  ngOnInit(): void {

    this.createProductAddForm();

    this.categoryService.getCategories().subscribe((data) => {

      this.categories = data;

    });

  }

  add() {

    //productAddForm valid ise formun değerini product'a atayacak.

    if (this.productAddForm.valid) {

      this.product = Object.assign({}, this.productAddForm.value);

      this.productService.addProduct(this.product).subscribe((date) => {

        this.alertifyService.success('Ürün Başarıyla Eklendi');

      });

    }

  }

}

Login Componentinin Yazılması

-- Login isminde bir component generate ettik.

-- Bootstrapten hazır bir login sayfası kopyaladık ve onu kendimize göre düzenledik.

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

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

import { AccountService } from '../services/account.service';

import { User } from './user';

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  model:User=new User();

  constructor(private accountService:AccountService) { }

  login(form:NgForm){

    this.accountService.login(this.model);

  }

  ngOnInit(): void {

  }

}

Account Servisi ve Localstorage

-- user classını oluşturduk.

-- fake bir account service yazarak login işlemini yapmak için account service oluşturduk.

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

@Injectable({

  providedIn: 'root'

})

export class AccountService {

  constructor() { }

  loggedIn=false;

  login(user:User):boolean{

    if(user.userName=="Furkan"&&user.password=="456x654x")

    {

      this.loggedIn=true;

      localStorage.setItem("isLogged",user.userName);

      return true;

    }

    return false;

  }

  isLoggedIn(){

    return this.loggedIn;

  }

  logOut(){

    localStorage.removeItem("isLogged");

    this.loggedIn=false;

  }

}

CanActivate Guard ile Çalışmak

import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';

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

import { AccountService } from '../services/account.service';

// app-routing'deki routlarımıza canActivete:[Login] ekleyerek giriş yapılmadan o routa ulaşılmasını engelleriz.

@Injectable()

export class LoginGuard implements CanActivate{

    constructor(private accountService:AccountService, private router:Router){}

    // Gitmek istediği yer, bulunduğu yer

    canActivate(route:ActivatedRouteSnapshot, state:RouterStateSnapshot):boolean{

        let logged=this.accountService.isLoggedIn();

        if(logged){

            return true;

        }

        // login değilse login sayfasına yönlendirecek.

        this.router.navigate(["login"]);

        return false;

    }

}

Login İşleminin Yapılması

<form class="form-signin" #loginForm="ngForm" (ngSubmit)="login(loginForm)">

    <div class="text-center mb-4">

      <h1 class="h3 mb-3 font-weight-normal">Üye Girişi</h1>

    </div>

    <div class="form-label-group">

      <input type="text"  #username="ngModel" [(ngModel)]="model.userName" id="userName" name="userName" class="form-control" placeholder="Kullanıcı Adı" required autofocus>

      <label for="inputEmail">Kullanıcı Adı</label>

    </div>

    <div class="form-label-group">

      <input type="password" id="password" #password="ngModel" [(ngModel)]="model.password"  name="password"  class="form-control" placeholder="Şifre" required>

      <label for="inputPassword">Şifre</label>

    </div>

    <button class="btn btn-lg btn-primary btn-block" type="submit">Giriş</button>

    <p class="mt-5 mb-3 text-muted text-center">&copy; 2017-2020</p>

  </form>

Login ve Logout Bölümünün Yazılması

-- App.component.ts içerisine account service’imizi import ederek login ve logout fonksiyonlarımızı oluşturacağımız fonksiyonlar içerisinde kullandık.

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

import { AccountService } from './services/account.service';

@Component({

  selector: 'app-root',

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

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  constructor(private accountService:AccountService){}

  title = 'shop';

  isLoggedIn(){

    return this.accountService.isLoggedIn();

  }

  logOut(){

    this.accountService.logOut();

  }

}

-- Daha sonra bu fonksiyonları app.component.html sayfasındaki inputlarda kullandık.

<a \*ngIf="isLoggedIn()==false" class="nav-link" routerLink="/login">

          Login

        </a>

        <a \*ngIf="isLoggedIn()==true" (click)="logOut()" class="nav-link" routerLink="/login">

          LogOut

        </a>