Wise Quarter

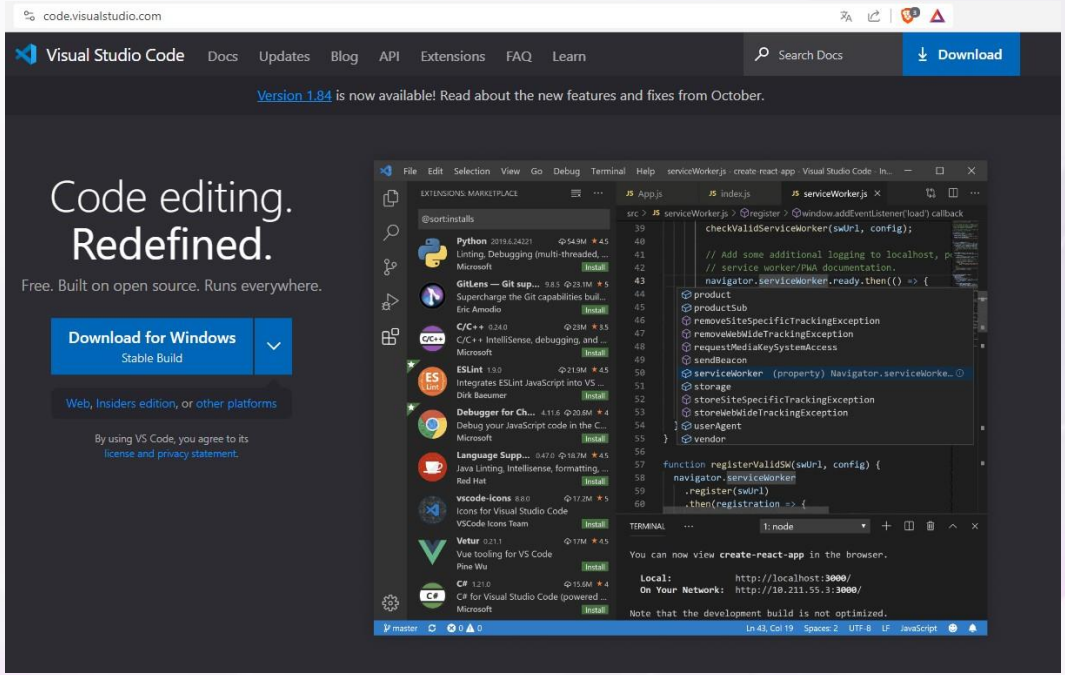
T135

HTML – CSS - JAVASCRIPT

Murat BABAYİĞİT

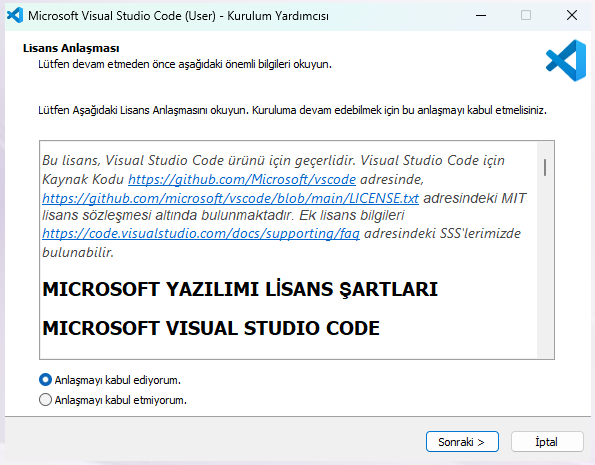
Wise Quarter

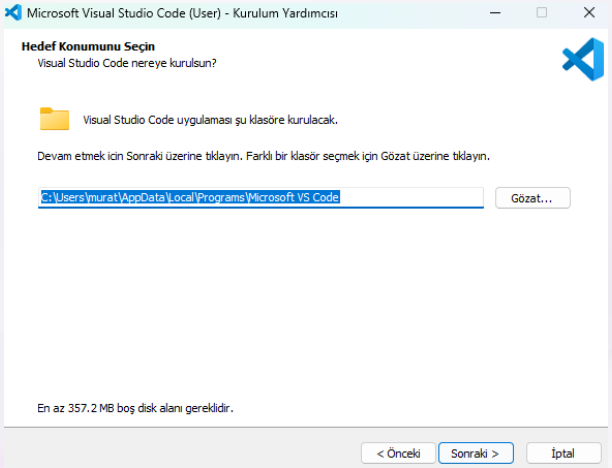
VSCODE KURULUM

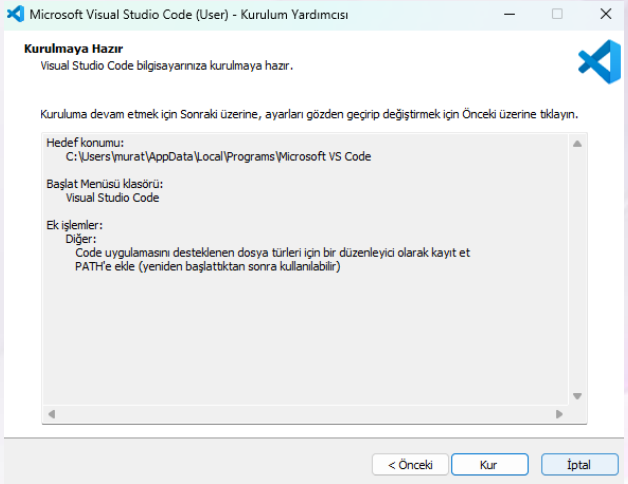


Wise Quarter

VSCODE KURULUM

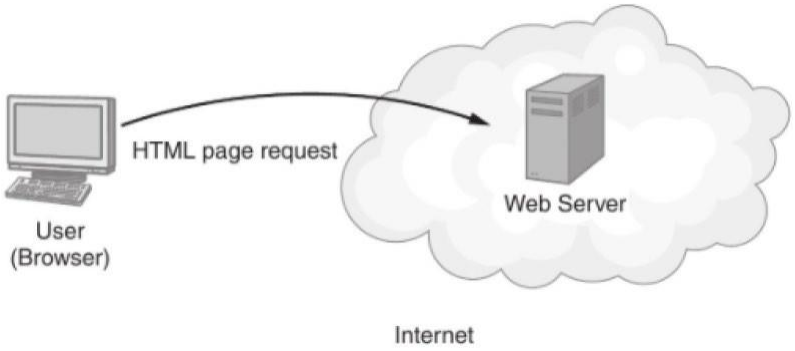


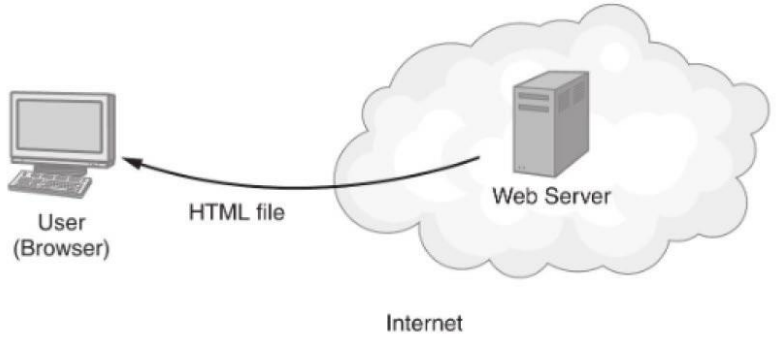




HTML

Hypertext Markup Language


HTML page request
Web Server
User
(Browser)
Internet


HTML file
Web Server
User
(Browser)
Internet

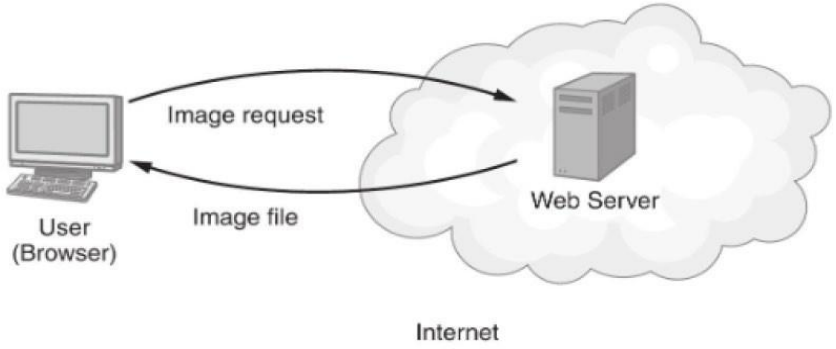
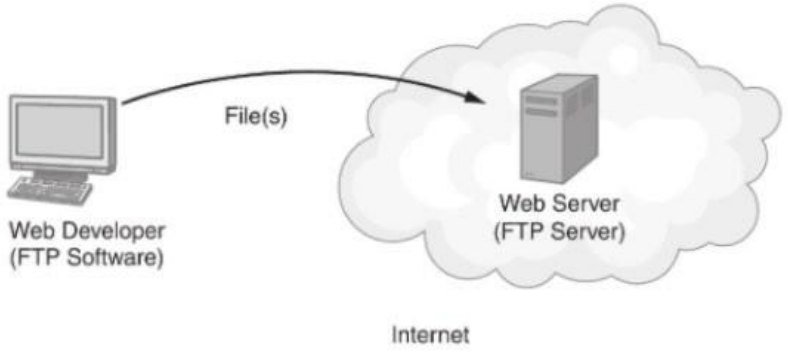


Image request
Image file
Web Server
User
(Browser)
Internet


File(s)
Web Server
(FTP Server)
Web Developer
(FTP Software)
Internet

HTML NEDİR?

Web sayfaları hazırlamak için kullanılan bir işaretleme dilidir.
HTML bir programlama dili değildir, işaretleme dilidir.
(Hyper Text Markup Language)

- Html komutlarına **tag** denir.
- Taglar büyüktür küçüktür işaretlerinin arasına yazılır **<>**

  Örnek : <html> <u> <img src="……">
- İstisnalar dışında tagların başlangıç ve bitişleri vardır.

  Bitiş tagları "/" işareti ile belirtilir.

  Örnek : Merhaba Nasılsın

Version	Date
HTML 1.0	1989
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML 1.0	2001
XHTML 2.0	discontinued in 2009
HTML 5.0	2012
HTML 5.2	2017


two-sided tag enclosing element content
empty elements, which do not contain content
an element attribute
several elements nested within another element
<!DOCTYPE html>
<html>
<head>
<title>Curbside Thai</title>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<link href="ct_base.css" rel="stylesheet" />
<link href="ct_layout1.css" rel="stylesheet" />
</head>
<body>
<header>

</header>
<nav>

</nav>
<footer>
Curbside Thai • 411 Belde Drive, Charlotte NC 28201 • 704-555-1151
</footer>
</body>
</html>

Element Hiyerarşisi

Bir HTML belgesinin tüm yapısı iç içe geçmiş hiyerarşik bir ağaçtaki öğeler gibi bir dizi olarak düşünülebilir. Ağacın en üstünde html elemanı bulunur vetüm belge. Html öğesinin içinde, bilgileri çevreleyen head öğesi bulunurbelgenin kendisi ve web sayfasının içeriğini çevreleyen gövde öğesi hakkında.Bu nedenle bir HTML dosyasının genel yapısı şöyledir

```
<!DOCTYPE html>
<html>

    <head>
      <title> BAŞLIK </title>
    </head>
    <body>

    </body>

</html>
```

burada head content ve body content iç içe yerleştirilmiş öğelerdir.belge başı ve gövdesi. Gövde öğesinin her zaman baş öğesinden sonra yerleştirildiğini unutmayın

HTML, web sayfaları oluştururken kullanılan bir işaretleme dilidir. İngilizce adıyla, "Hyper-text Markup Language", Türkçe adıyla "Hiper Metin İşaretleme Dili"nin kısaltmasıdır.

Bir web sayfasına girildiğinde, son kullanıcının gördüğü her zaman HTML çıktısıdır. Onun daha güzel ve düzenli görünmesini sağlayan CSS, daha dinamik bir yapıda kullanılmasını sağlayan ise JavaScripttir.

CSS ve JavaScript konularına, HTML'den sonra bakılması yeni başlayacak olanların faydasına olacaktır. Zira HTML'i iyi öğrendikten sonra sırasıyla CSS ve JavaScript, front-end geliştirme yapmak isteyenlerin izlemesi gereken bir yoldur.

- HTML, uzun adıyla "Hyper Text Markup Language" demektir.
- HTML, Türkçe adıyla "Hiper Metin İşaretleme Dili" demektir.
- HTML ile yazılar işaretlenerek, web sayfalarının yapıları belirlenir.
- HTML ile işaretleme yapılırken belli başlı html etiketleri kullanılır.
- HTML etiketleri, "başlık", "paragraf", "tablo" vb. bölümleri işaretlemek için kullanılır.

UNUTMAYIN

Web tarayıcıları, HTML etiketlerini göstermez ancak sayfaların içeriklerini oluşturmak için bu etiketleri kullanır. Bu yüzden kurallarına uygun HTML yazmak, hem tarayıcıların kodlarınızı yorumlamasını kolaylaştırır hem de arama motoru dostu sayfalar oluşturmanızı sağlar.

HTML sayfası oluşturmak

Bir web sayfası oluşturmak için yapılması gerekenler:

Html uzantılı dosya oluştur ⇌ **html**
head
body
/html

Sunucu Notları
2024-01-06 20:59:15

HTML bır ebeveyn ıse HTMLin kac cocugu var
Title htmlın neyı oluyor

HTML sayfası oluşturmak

Bir web sayfasının en temel şablonu :

```
<html>
  <head>
    <title>Ilk sayfam</title>
  </head>
  <body>
    Merhaba. Bu benim ilk web sayfam
  </body>
</html>
```

HTML5

Bir html sayfasının tarayıcılar tarafından Html5 olarak algılanabilmesi için:

Sayfanın en üst kısmına

<!doctype html>

eklenmesi gerekir.

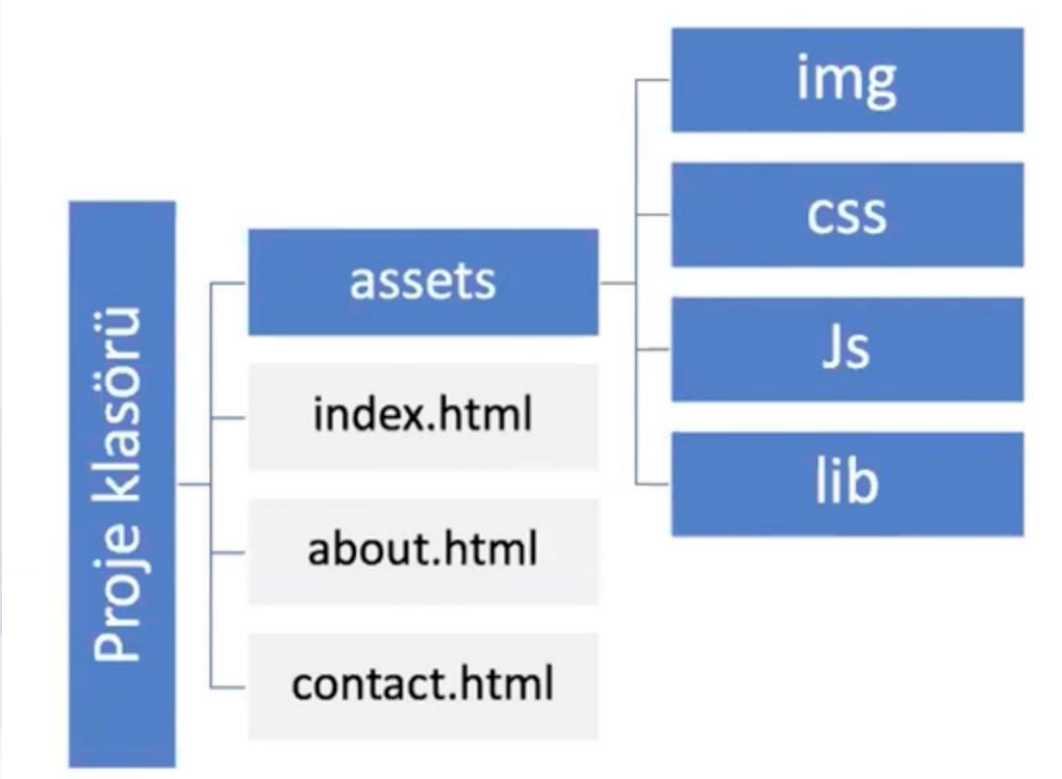
```
<!DOCTYPE html>
<html>

</html>
```

HTML bir site oluşturma

1. Desktop'umuzda MyPage isimli bir klasör, içine de index.html dosyası ve assets klasörü, assts içine de img, css, js, lib isimli klasörler oluşturuyoruz.
2. index.html dosyasına web sayfası için gerekli olan tagları oluşturuyoruz.

İpucu: VS Code'da ! işareti yazıp enter'a basarsanız gerekli taglar oluşturulur.

KLASÖR YAPISI



Temel biçimlendirme tagları

p

paragraph

Yazıları paragraf haline getirir.

Lorem Ipsum, adı bilinmeyen bir matbaacının bir hurufat numune kitabı oluşturmak üzere bir yazı galerisini alarak karıştırdığı 1500'lerden beri endustri standardı sahte metinler olarak kullanılmıştır.

br

break

Metin içinde satır başı yapmak için kullanılır.

```
<p>Lorem Ipsum, adı bilinmeyen bir matbaacının bir
hurufat numune kitabı oluşturmak üzere <br>
bir yazı galerisini alarak karıştırdığı 1500' lerden beri
endüstri standardı sahte metinler olarak kullanılmıştır.</p>
```

HTML kodlarken alt satıra geçmek için enter tuşuna basılmasının veya kelimeler arasında birden fazla boşluk bırakmak için space tuşuna basılmasının ziyaretçi tarafında bir etkisi yoktur.

Temel biçimlendirme tagları

hx

head

Yazıları başlık haline getirir.

Standart 6 tane başlık tagı bulunmaktadır.

```
<h1>Başlık 1</h1>
<h2>Başlık 2</h2>
<h3>Başlık 3</h3>
```

hx taglarının kullanımı

```
<h1>IT ALANINDAKI MESLEKLER</h1>
<p>Bu yazımızda IT sektöründe
uzmanlaşabileceğiniz meslekleri anlattım.</p>

<h2>Automation Engineer</h2>
<p>...</p>

<h3>Full Stack Developer</h3>
<p>...</p>
```

Head taglarının kullanımı

IT ALANINDAKİ MESLEKLER

Bu yazımızda IT sektöründe uzmanlaşabileceğiniz meslekleri anlattım.

Automation Engineer

...

Full Stack Developer

.....

Temel biçimlendirme tagları

hr

horizontal row

Yatay çizgi oluşturmak için kullanılır.

<hr>

Lorem Ipsum, adı bilinmeyen bir matbaacının bir hurufat numune kitabı

---

oluşturmak üzere bir yazı galerisini alarak karıştırdığı 1500'lerden beri endüstri standardı sahte metinler olarak kullanılmıştır.

Temel biçimlendirme tagları

strong → **Metinleri kalınlaştırmak için kullanılır.**

Lorem Ipsum, adi bilinmeyen bir matbaacinin bir hurufat numune kitabi olusturmak uzere bir yazi galerisini alarak karistirdigi 1500'lerden beri endustri standardi sahte metinler olarak kullanilmistir.

Temel biçimlendirme tagları

i

italic

Yazıyı eğik *(italik)* yazdırır.

Lorem Ipsum, adi bilinmeyen bir matbaacinin bir hurufat numune kitabi olusturmak uzere bir yazi galerisini alarak karistirdigi 1500'lerden beri endustri standardi sahte metinler olarak kullanilmistir.

İç içe taglar

İç içe tag kullanıldığında dikkat edilmesi gereken kural :
Önce açılan tag en son kapatılır.

```
<p>
        <strong>Merhaba<u>Dunya</u></strong>
</p>
```

Temel biçimlendirme tagları

U

underline

Yazının altını çizmek için kullanılır.

<u>Bu yazının altını çizer**</u>**

Wise Quarter

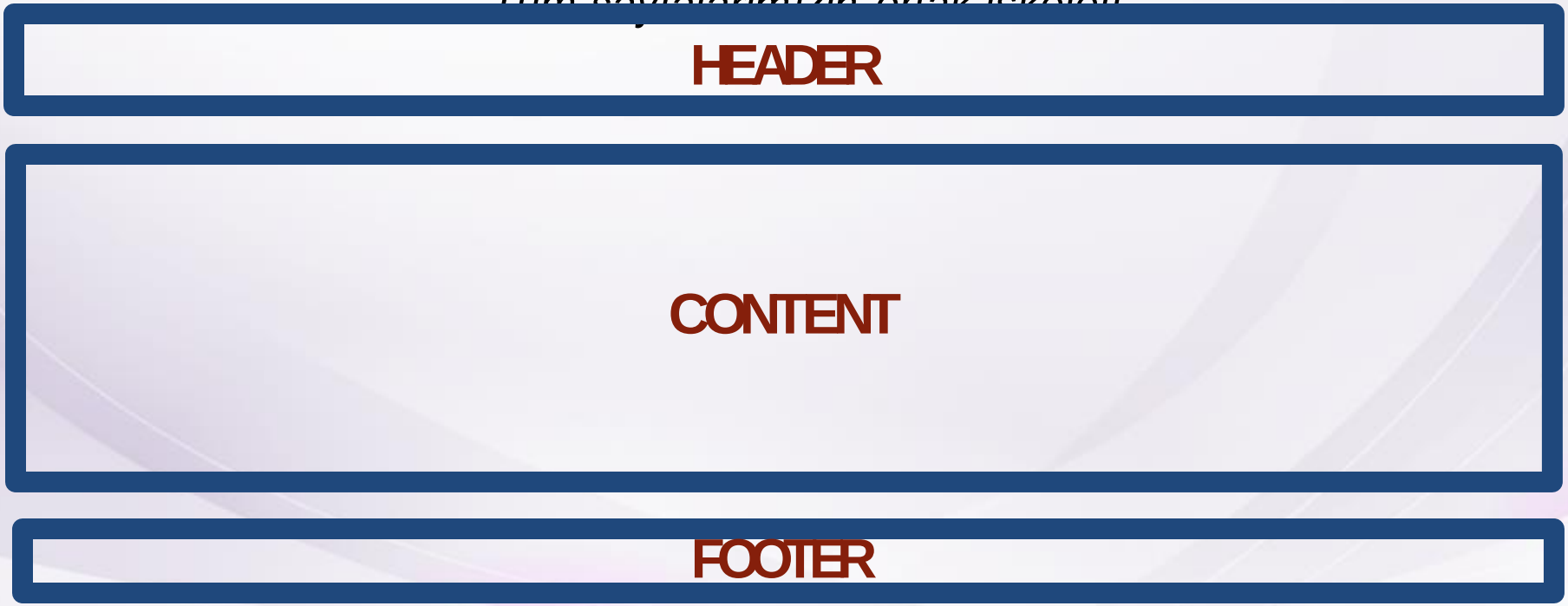
div & span

div → Blok elemandır, kapladığı alan ne olursa olsun, tüm satırları işgal eder.

span → Inline elemandır, sadece kapladığı alan kadar yer işgal eder.

Resume Sitesi Oluşturma

Tüm sayfalarımızın ortak iskeleti



Resume Sitesi Oluşturma

Home sayfamızın iskeleti

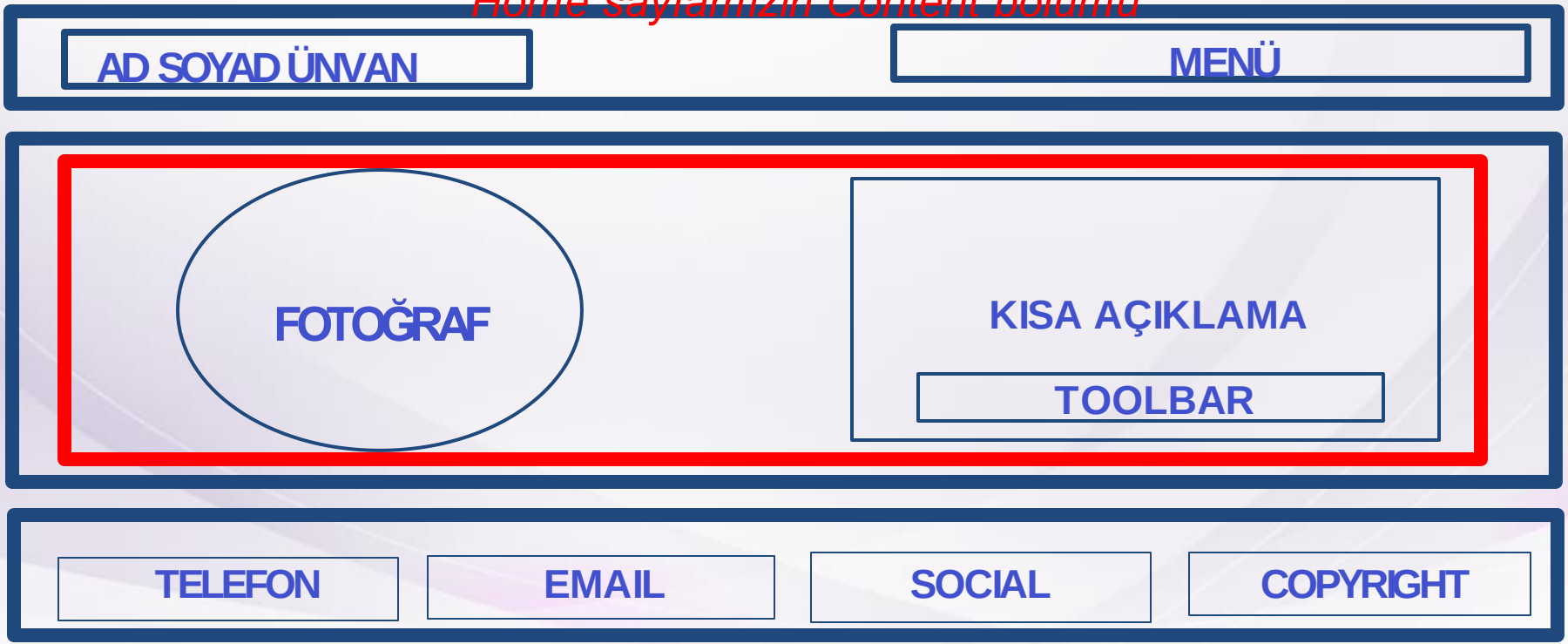
Wise Quarter

Resume Sitesi Oluşturma

- Home sayfası için header, content ve footer bölümlerini oluşturunuz.
- Header bölümündeki ad, soyad, unvan ve menü bölümlerini oluşturunuz.
- Footer bölümündeki telefon, email, sosyal medya ve copyright bölümlerini oluşturunuz.
- Bu işlemleri tamamladıktan sonra **resume.html** ve **contact.html** dosyalarımıza kopyalıyoruz. Bu sayede ortak olan alanları hepsi için tek seferde tamamlamış oluyoruz. Her bir sayfanın Content bölümünü de ayrı ayrı şekillendireceğiz.

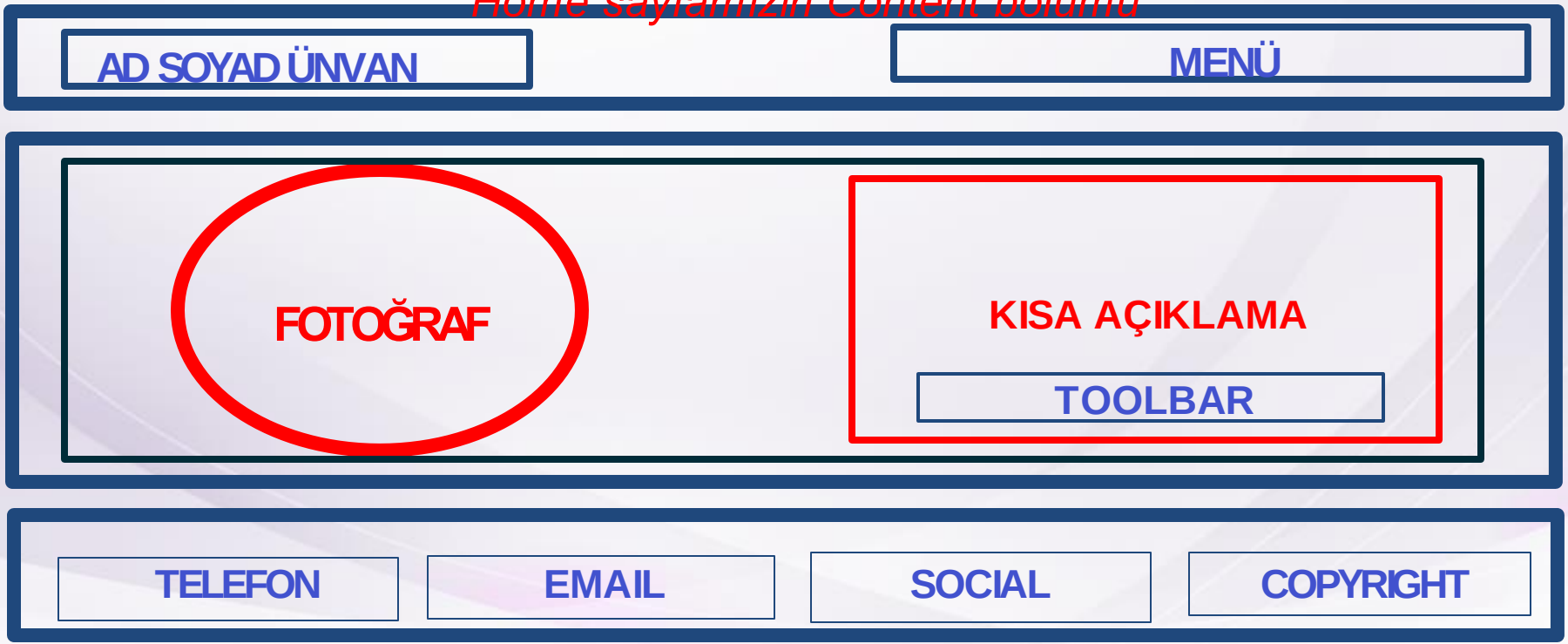
Resume Sitesi Oluşturma

Home sayfamızın Content bölümü



Resume Sitesi Oluşturma

Home sayfamızın Content bölümü



Resume Sitesi Oluşturma

- Home sayfası content bölümündeki about kısmını oluşturunuz.
- Fotoğraf için gerekli alanı oluşturunuz.
- Kısa açıklama kısmını oluşturunuz.
- Kısa açıklama içinde toolbar bölümünü oluşturunuz.

Bağlantı tagı

a

anchor

Web sitelerinde,

➔ Bir sayfadan başka bir sayfaya,

➔ Farklı bir siteye

Bağlantı vermek için <a> tagı kullanılır.

Wise Quarter

Bağlantı tagı

a

anchor

Başka bir siteye link vermek için;

```
<a href="https://google.com.tr">Google' a Git</a>
```

Bir dosyaya link vermek için;

```
<a href="img/resim1.jpg">Fotoğrafı aç</a>
```

Bağlantıyı yeni bir sekmede açmak için;

```
<a href="https://google.com.tr" target="_blank">Google' a Git</a>
```

Resume Sitesi Oluşturma

- Header bölümündeki menü bölümünün bağlantılarını oluşturunuz.
- Content kısmındaki toolbar bölümünün bağlantılarını oluşturunuz.
- Footer bölümündeki linklerin bağlantılarını oluşturunuz.

Fotoğraf yükleme

img

image

Fotoğraf ekleme işlemleri img tagı ile yapılır.

```
<img src="img/acer-aspire-one-notebook.jpg" alt="Acer Aspire One Notebook”>
```

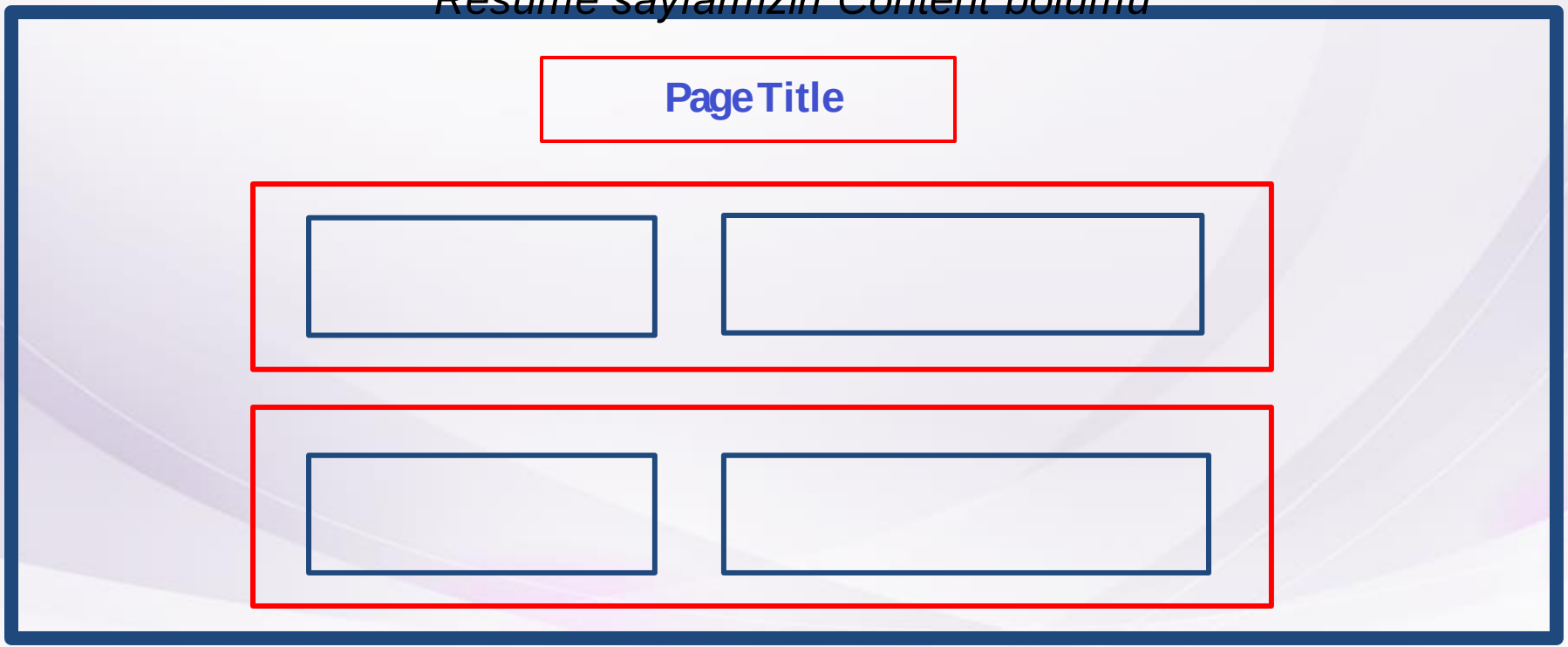
Resume Sitesi Oluşturma

- Home page'deki Content bölümündeki fotoğrafı ekleyiniz.

unsplash.com, pexels.com, flaticon.com ücretsiz fotoğraf kullanabileceğiniz bir sitedir.

Resume Sitesi Oluşturma

Resume sayfamızın Content bölümü

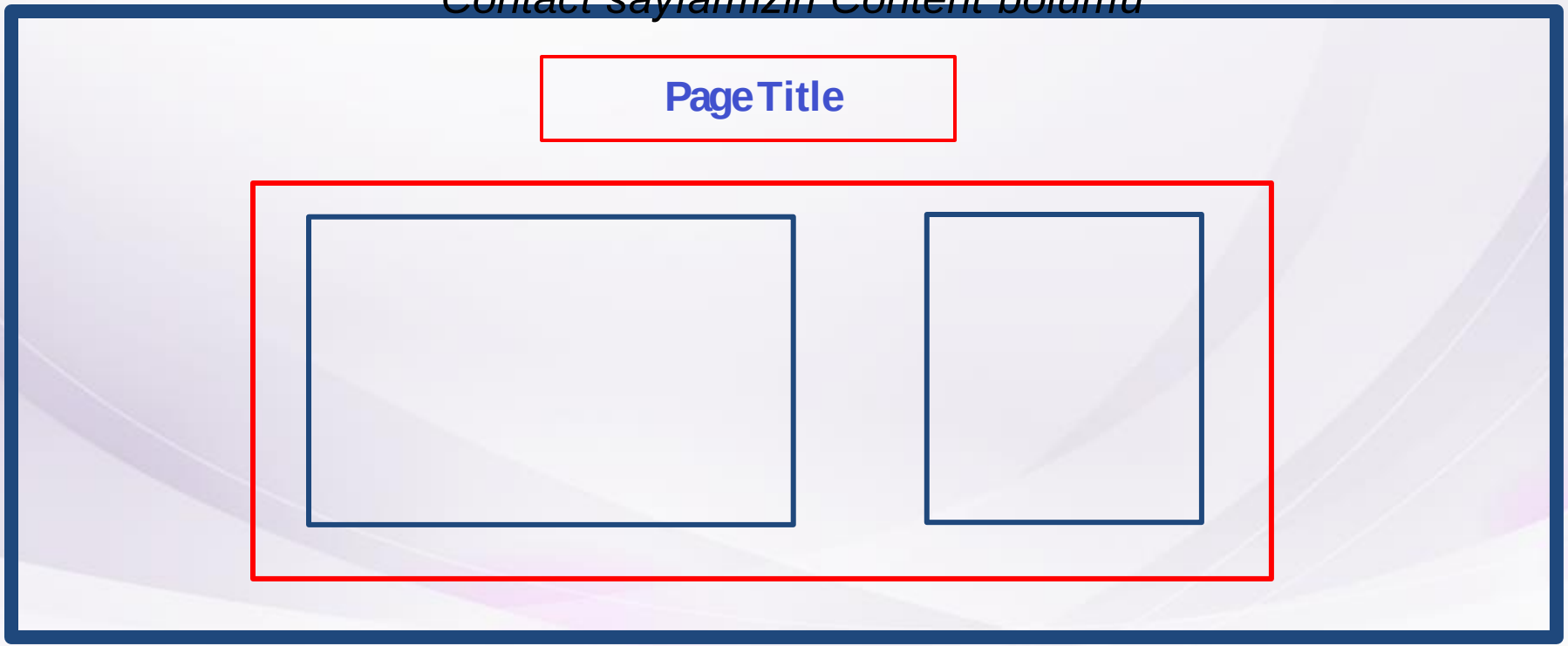


Resume Sitesi Oluşturma

- Resume page'deki Content bölümündeki Page Title, Education ve Work Experience alanlarını oluşturunuz.

Resume Sitesi Oluşturma

Contact sayfamızın Content bölümü



Resume Sitesi Oluşturma

- Contact page'deki Content bölümündeki Page Title ve Form alanlarını oluşturunuz.

Form oluşturma

Kullanıcıdan bilgi almak için kullanılan yapılara form denir. Form içinde textbox, button, radio button, checkbox vb elemanlar olabilir.

- **form :** Tüm form elemanlarını içinde barındıran taşıyıcıdır.
- **input :** Kullanıcıdan bilgi almak için kullanılan form elemanlarının genel ismi.
- **label :** Kullanıcıya form elemanı ile ilgili bilgi vermek için kullanılan eleman.

Form oluşturma

```
<form>
<label>Eposta</label>
<input type="text">
<input type="button" value="Kaydet">
</form>
```

Wise Quarter

Input

```
<input type="text | email | hidden ...." name="adi" value="ali" placeholder="">
```

text → Genel bilgi girişi için kullanılır.

number → Sadece rakam girişi için kullanılır.

email → Sadece e-posta girişlerini kabul eder.

password → Şifre girişi için kullanılır. Girilen değer gösterilmez.

Wise Quarter

Input

```
<input type="radio | checkbox | file ...." name="hobiler" value="futbol">
```

checkbox → Birden fazla işaretleme yapılabilecek seçenekler sunar.

radio → Sadece tek seçim yapabileceği seçenekler sunar. Cinsiyet gibi. Name attribute'u aynı olan radio butonlar kendi aralarında grup olurlar.

file → Dosya yüklemek için kullanılır.

Wise Quarter

Select

select → Kullanıcıya listeden seçim yaptırmak için kullanılır. Ülke, şehir gibi seçimler bu şekilde yapılır. Bunun için select tagı kullanılır. Her bir seçenek ise option tagı içine konulur.

```
<select name="iller">
        <option value="01">Adana</option>
        <option value="02">Adıyaman</option>
        <option value="03">Afyon</option>
        <option value="04">Ağrı</option>
</select>
```

Wise Quarter

Textarea

textarea → Çok uzun miktarda yazı girilmesini sağlayan form elemanı textarea'dır. Geçerli değer textarea tagları arasına yazılabilir.

```
<textarea name="mesaj" rows="4"></textarea>
```

Rows attribute ile textarea nın aynı anda gösterebileceği satır sayısı belirlenir

Button

- Genel amaç için button oluşturur. Butonun default olarak bir görevi yoktur.

```
<input type="button|submit|reset" name="btn" value="Kaydet">
```

button → Form bilgilerini backend'e gönderen bir buton oluşturur.

submit →

- Formdaki bilgileri resetleyen bir buton oluşturur

reset →

Form oluşturma

Submit butonlara basıldığında formdaki bilgiler sunucuya gönderilir. Bu gönderme işlemi yapılırken inputların name attribute'u ile value attribute'u kullanılır.

```
<form action="saveDatabase.php">
        <label>Ad</label>
        <input type="text" name="ad" value="">
        <br>
        <label>Şifre</label>
        <input type="password" name="sifre" value="">
        <br>
        <label>Yaş</label>
        <input type="number" name="yas" value="">
</form>
```

→

ad=Ali
sifre=12345
yas=25

Resume Sitesi Oluşturma

→ Contact bölümündeki formu oluşturunuz.

Wise Quarter
first class IT courses

CSS

Cascading Style Sheet
(Basamaklı Stil Şablonları)

1. Ders

Murat BABAYIGIT

Full-Stack Automation Engineer - Instructor

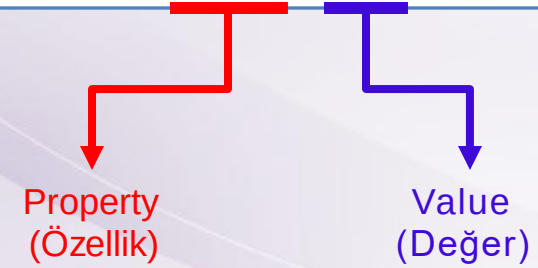
Wise Quarter

Css nedir?

HTML elemanlarının stil özelliklerini belirleyen bir işaretleme dilidir. CSS sayesinde tek bir tanımlama ile birçok elementin özelliği aynı anda değiştirilebilir. HTML ile sayfanın iskeleti ve içeriği tanımlanırken, CSS ile biçimlendirmesi yapılır.

CSS eklemek için ilgili tagın içine style attribute'u eklenir. Buna inline stil denir.

```
<p style="color: red">Merhaba</p>
```



Birden fazla stil ekleneceği zaman, stil özelliklerinin arasına ; (noktalı virgül) konulur.

```
<div style="color:red; text-
align:center">Merhaba</div>
```

İnline stiller sadece bulundukları tagın içinde etkilidirler.

Wise Quarter

Temel CSS özellikleri

font-size

Yazının büyüklüğünü değiştirmek için kullanılır.

```
<a href="sayfa1.html" style="font-size: 20px">click</a>
```

text-align

Yazının bulunduğu tag içindeki yataydaki hizasını belirlemek için kullanılır. Kullanılabilecek değerler: **left, right, center, justify**

```
<div style="text-align : left">Sitemize hoşgeldiniz</div>
```

Temel CSS özellikleri

color

Yazının rengini değiştirmek için kullanılır. Kullanılabilecek değerler: **hazır renk tanımları(red, green, yellow ...), hexadecimal system (#12AF45), rgb, rgba**

```
<p style="color:#1255FA">Merhaba</p>
```

background-color

Html elemanının zemin rengini değiştirmek için kullanılır. Kullanılabilecek değerler color ile aynıdır.

```
<p style="background-color:#1255FA">Merhaba</p>
```

Sayfa seviyesinde style (external)

Site içindeki tüm sayfalarda etkili olabilecek stil tanımlaması yapmak için harici stil sayfaları kullanılır.

- Bunun için css uzantılı bir dosya oluşturulup tüm stil işlemleri bunun içinde yapılır.
- Oluşturulan bu stil dosyası ile web sayfaları <link> komutu ile ilişkilendirilir. <link> tagı head tagı içinde tanımlanır.

```
<link href="css/stil.css" rel="stylesheet">
```

Seçiciler (Selectors)

- Sadece istenilen belli html elemanlarına stil uygulamak için selector'lerin doğru kullanılması gerekir.

1. Tüm html elemanlarını seçmek için * işareti kullanılır.

```
* {
  text-align: center;
  color: blue;
}
```

2. Belli türdeki html elemanlarını seçmek için elemanın ismi seçici olarak kullanılır.

```
h1 {
  text-align: right;
  font-weight: 300;
}
```

Temel CSS özellikleri

3 Bazı html elemanlarını seçmek için class tanımlaması yapılır.

CSS

```
.ortala {
  text-align: center;
  color: blue;
}
```

HTML

```
<p
class="ortala">Merhaba</p>
```

Wise Quarter

Metin Stilleri

text-decoration

Metnin altına, üstüne veya üzerine çizgi çekmek için kullanılır. Alabileceği değerler **underline, overline, line-through, none**

```
a{
        text-decoration: none;
}
```

font-style

Yazıyı italik hale getirmek için kullanılır. Alabileceği değerler **normal, italic**

```
font-style : italic;
```

font-weight

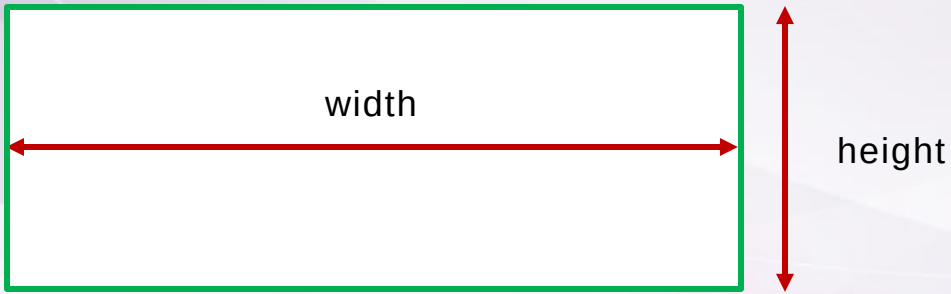
Yazıyı kalınlaştırmak için kullanılır. Alabileceği değerler **100 – 900, bold, bolder, lighter**

```
font-weight : 500;
```

Temel CSS özellikleri

width Html taglarının genişliğini (yatayda kapladığı alanı) ayarlamak için kullanılır.

height Html taglarının yüksekliğini (dikeyde kapladığı alanı) ayarlamak için kullanılır.



Width ve height için %, px birim olarak kullanılabilir.

Border (Çerçeve)

- **Elementlere çerçeve vermek için border stili uygulanır.**

```
border: genişlik tip renk;
```

```
border: 3px solid red;
```

```
border: 3px dashed blue;
```

```
border: 5px dotted gray;
```

Tip için verilebilecek değerler: **dotted, dashed, solid, double, groove, ridge, inset, outset, none**

Border (Çerçeve)

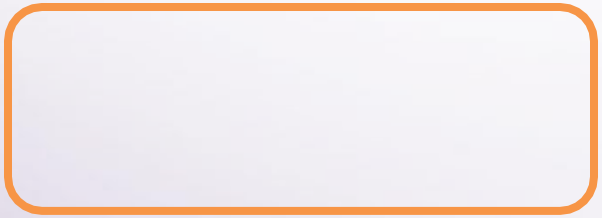
- **Elementlerin 4 tarafına border verilebildiği gibi sadece istenilen tarafına da verilebilir.**

```
border-left | border-right | border-bottom | border-top
```

```
border-top: 3px solid red;
```

Border (Çerçeve)

- **Çerçevelerin köşelerini yumuşatmak için border-radius stili kullanılır.**



```
div{
        border:2px solid red;
        border-radius: 8px;
}
```

```
div{
        background-color:red;
        border-radius: 8px;
}
```

Wise Quarter

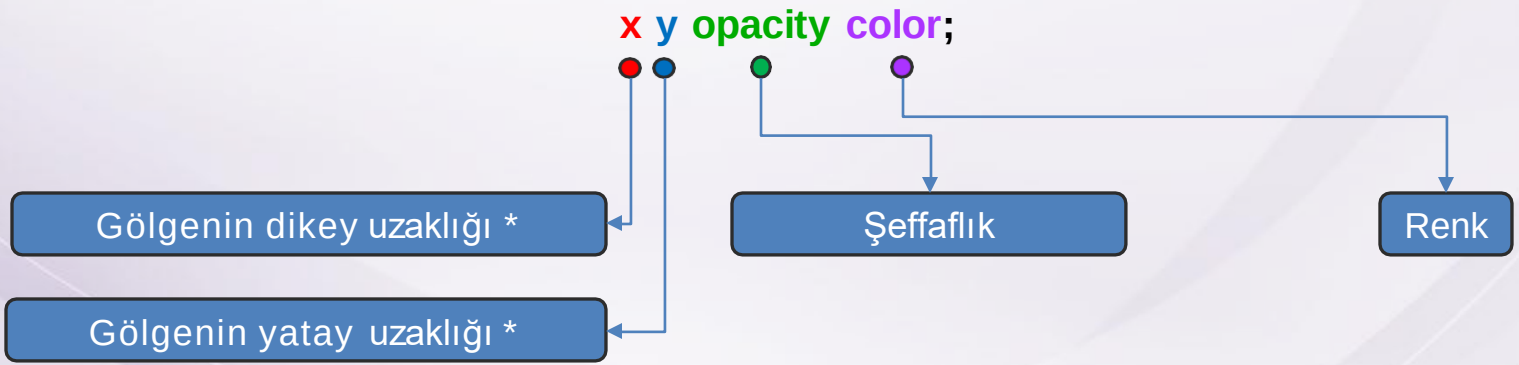
Shadow (Gölge)

Yazılara gölge vermek için

text-shadow

Elementlere gölge vermek için

box-shadow




```
box-shadow: 5px 5px 10px red
```

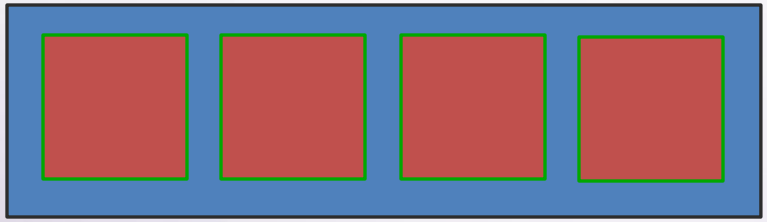
Resume Sitesi

- Resume sitesi için harici stil dosyası oluşturunuz.
- Tüm elemanlar için box-sizing tanımlaması yapalım.
- Body için font ve padding ayarlaması yapınız.
- Tüm linklerdeki alt çizgiyi kaldırınız.
- Form inputlara border, padding ve border-radius verelim.
- Buton'lara border, border-radius, padding ve background-color verelim.

Wise Quarter

Flexbox

- **Flexbox, html elemanlarının sayfa üzerinde yerleşimlerini ayarlamak için kullanılan bir CSS özelliğidir.**



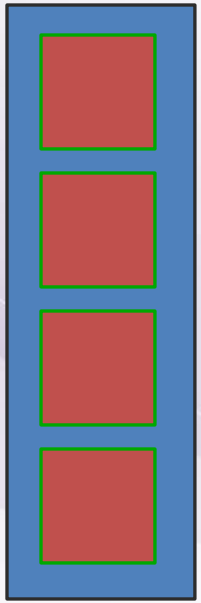
```
<div class="container">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
</div>
```

```
.container{
        display: flex;
}

.container div{
        width:160px;
        height: 160px;
        background-color:
#13aa45;
}
```

Flexbox

flex-direction

Flex-direction ile bu dizilimin yönü belirlenebilir. Row olursa yan yana, column olursa alt alta gösterilir. Default değer: row



```
<div class="container">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
</div>
```

```
.container{
        display: flex
        flex-direction:
column;
}

.container div{
        width:160px;
        height: 160px;
        background-color:
#13aa45;
}
```

Wise Quarter

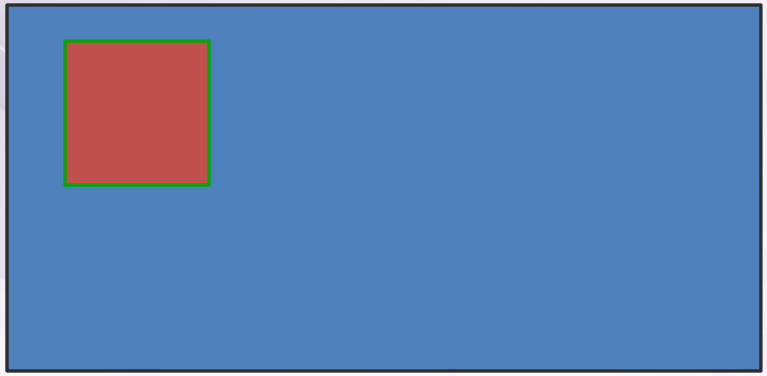
Flexbox

justify-content

Flex içindeki elemanları hizalamak için kullanılır.

flex-start | flex-end | center | space-between | space-around

- flex-start: Baş tarafa hizalar
- flex-end: Son tarafa hizalar
- center: Ortalar
- space-between: Eşit boşluklarla satıra yayar.
- space-around: Eşit boşluklarla satıra yayar. Başta ve sonda da boşluk verir.



```
.container{
        display: flex;
        justify-content: flex-
start;
}
```

```
<div class="container">
        <div></div>
</div>
```

Wise Quarter

Flexbox

align-items

Flex içindeki elemanları hizalamak için kullanılır.

stretch | center | flex-start | flex-end

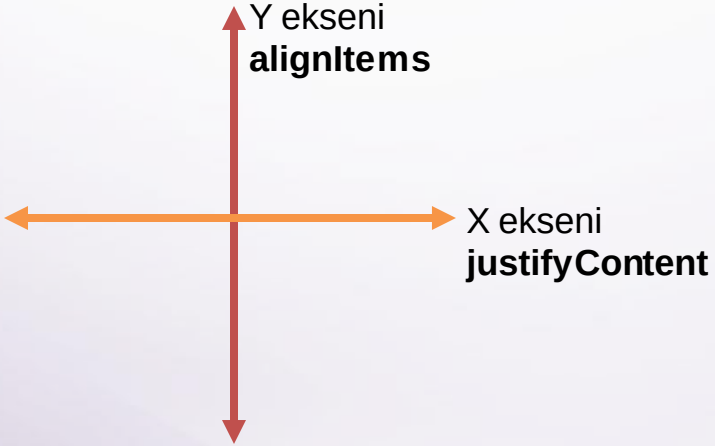
- stretch: Taşıyıcı yüksekliğine sığdırır
- center: Ortalar
- flex-start: Üste hizalar
- flex-end: Alta hizalar

```
.container{
        display: flex;
        align-items : center;
}
```

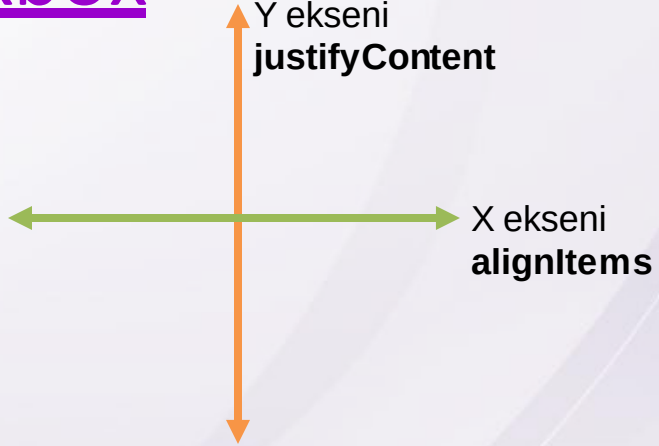
```
<div class="container">
        <div></div>
</div>
```

Wise Quarter

Flexbox



flexDirection: row



flexDirection: column

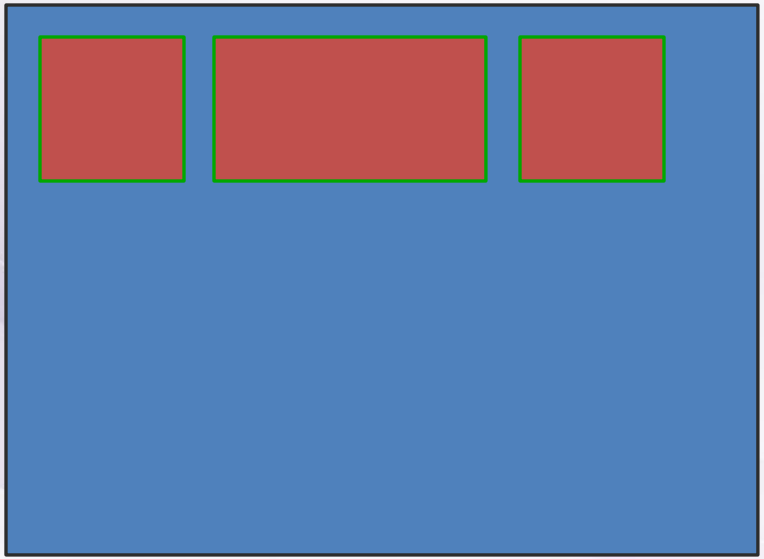
Flex-direction:
row olarak ayarlandığında **justifyContent** X ekseninde, **alignItems** Y ekseninde hizalama yapar.

Flex-direction:
column olarak ayarlandığında ise tam ters davranırlar.

Wise Quarter

Flexbox

flex — Aynı flexbox içindeki elemanlara göre büyüklük oranını belirler.



```
.container{
          display: flex;
}

.container .second{
          flex:2
}
```

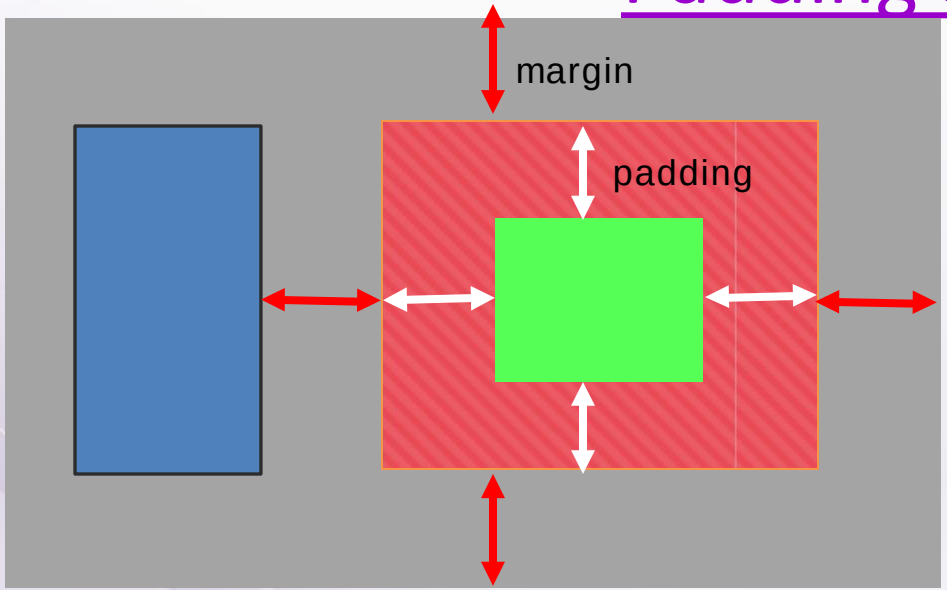
```
<div class="container">
          <div></div>
          <div class="second"></div>
          <div></div>
</div>
```

Resume Sitesi

Aşağıdaki elemanlar için flexbox ayarlaması yapınız :

- Header
- Content
- Card
- About
- Description
- Toolbar
- Footer

Padding & Margin




```
<div>
        <div class="kutu">
                <div>

                </div>
        </div>
</div>
```

```
.kutu{
        padding: 50px;
        margin: 90px;
}
```

padding Elementin içindeki diğer elementlerin kenarlardan olan uzaklığını belirler.

margin Elementin dıştaki elementlerden olan uzaklığını belirler.

Resume Sitesi

- Header’daki ad soyad bölümün başına yuvarlak bir ikon oluşturunuz.
- Page title bölümün başına yuvarlak bir ikon oluşturunuz.
- Toolbar ve card’lar için gölge oluşturunuz.

Yüksek Seviyeli: Hafıza yönetimi gibi karmaşık görevleri düşünmemize gerek yoktur.

Nesne Yönelimli: Nesne özelliklerinin (kalıtım, çok şekillilik v.b) kullanılmasına imkan sağlamaktadır.

Yorumlayıcı Tabanlı: Derleyicide olduğu gibi tüm komutların bir kere de makine koduna çevirmek yerine tek-tek alınıp makine koduna çevrilip çalıştırılmasını sağlar.

Wise Quarter

Javascript Fundamentals

- Node.js “Hello World” application
- Variables, Constants and Data Types
- Concatenation and Interpolation
- Objects and arrays
- Equality and Comparison operators
- Logical operators
- Conditional statement
- Loops
- Introduction to JavaScript Functions
- Introduction to JavaScript Classes and methods

INSTALLATION

Visual Studio Code

İlk kodlar

```
<html>
    <head>

    </head>
    <body>


        <script>
            ...
        </script>
    </body>
</html>
```

- Javascript kodlarını yazmak için script tagı kullanılır.
- Tüm javascript kodları script tagları arasına yazılır
- Script taglarının body kapanmadan hemen önce konulması tavsiye edilir.
- Bu kullanıma internal script denir ve javascript kodları sadece mevcut sayfa için geçerli olur.

İlk kodlar

```
<html>
    <head>

    </head>
    <body>


        <script src="assets/js/script.js">
    </body>
</html>
```

- Javascript kodları harici bir dokümana konularak kullanılırsa buna external script denir
- Bu dosyayı kullanan tüm html sayfalarında kullanılabilir hale gelir.

Kodların çalıştırılması

1

Web sayfalarında bulunan Javascript kodları, tarayıcı tarafından çalıştırıldığı için html dosyasına dosyaya çift tıklayarak açmak javascript dosyalarının çalışması için yeterlidir.

2

VSCode Live Server eklentisi ile çalıştırılırsa, yapılan değişikliklerde anında yansıtılmış olur. Ve gerçeğe yakın bir ortamda geliştirme yapılabilir.

3

node

Terminal node dosya ismi

Javascript syntax

- Case sensitive bir dildir.
- Satır sonlarına genellikle noktalı virgül (;) konulur.
- Identifier lar(variables, constants, functions) harf, _ veya $ ile başlayabilir.
- Identifier lar camel case ile yazılır.

Package.json oluşturalım

- Masaüstünde bir Klasör oluşturalım
- Klasörü VSCode içine taşıyalım
- Terminal açalım ve npm init basit ayarlamaları yapıp Json dosyasını oluşturacağız
- dependencies ve devDependencies kısımlarını oluşturalım. Buraya ihtiyaç halinde kullanabileceğimiz bazı kütüphaneleri koyacağız.

YORUM SATIRI

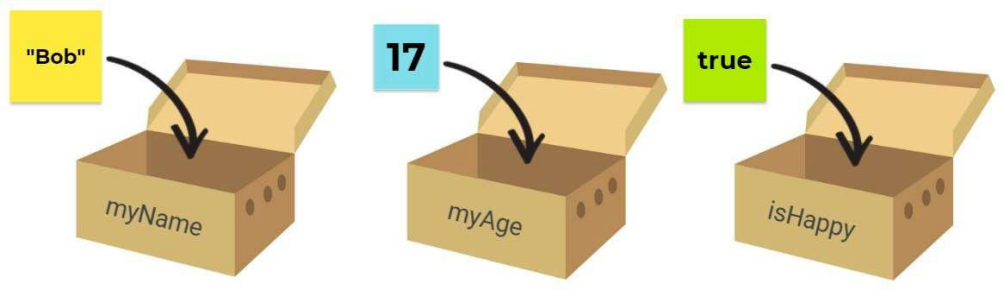
```
// Bu bir comment satırıdır.


/*
 Bu şekilde birden çok satırı aynı
 anda comment içine alabilirsiniz.
*/
```

Hello World

- P01.js dosyasını oluşturalım
- Terminal açıp cd day01 ile klasör içine girelim.
- node P01.js yazıp çalıştıralım

```
// Hello World yazalım
console.log("Hello World")
console.log(" ")
console.log(" ")

// Variable oluşturalım
var firstName="Murat"
let lastName="Yiğit"
var age,dateOfBirth,
age=43
dateOfBirth=1980
const occupation="Instructor"
console.log("First Name:",firstName,
"\nLast Name:",lastName,
"\nAge:",age,
"\nDate Of Birth:",dateOfBirth,
"\nOccupation:",occupation)
console.log("")
//bazı variable değerlerini değiştirelim
age=44
occupation="Teacher"
console.log("First Name:",firstName,
"\nLast Name:",lastName,
"\nAge:",age,
"\nDate Of Birth:",dateOfBirth,
"\nOccupation:",occupation)
console.log("")
```

Variables (Değişkenler), Constants (Sabitler)

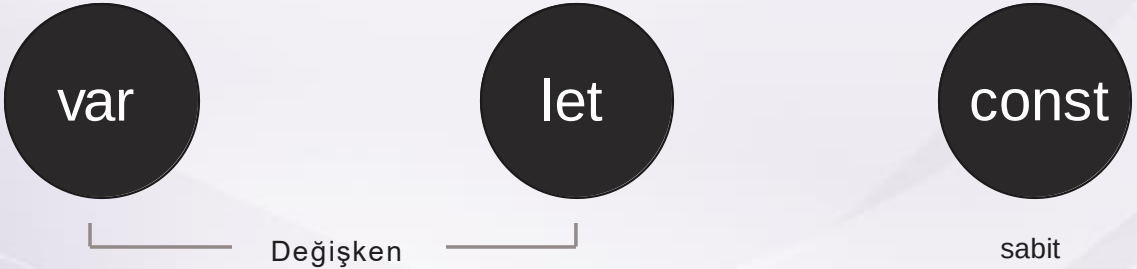


Programlama dillerinde geçici olarak değerleri saklamak için kullanılan yapılara değişken ya da sabit adı verilir.

Variables - Constants

✓ ders	✗ 1not	✓ gen_ort
✓ $maas	✗ ilk sayi	✗ son_şekil
✗ gun%	✗ harf-bir	✓ dGunu

Variables - Constants



Variables

Değişken tanımlamak için var, let, veya const anahtar kelimeleri kullanılır.

- **var:** Genişletilmiş scope, eski ve global scope değişken tanımlama için kullanılır.
- **let:** Block scope, modern ve değişebilir (mutable) değişken tanımlama için kullanılır.
- **const:** Block scope, modern ve değişmez (immutable) değişken tanımlama için kullanılır.

```
// Hello World yazalım
console.log("Hello World")
console.log(" ")
console.log(" ")

// Variable oluşturalım
var firstName="Murat"
let lastName="Yiğit"
var age,dateOfBirth,
age=43
dateOfBirth=1980
const occupation="Instructor"
console.log("First Name:",firstName,
"\nLast Name:",lastName,
"\nAge:",age,
"\nDate Of Birth:",dateOfBirth,
"\nOccupation:",occupation)
console.log("")
//bazı variable değerlerini değiştirelim
age=44
occupation="Teacher"
console.log("First Name:",firstName,
"\nLast Name:",lastName,
"\nAge:",age,
"\nDate Of Birth:",dateOfBirth,
"\nOccupation:",occupation)
console.log("")
```

Data Types

- 5 Primitive Data Types.
  - var firstname=''Murat"
  - var age=44
  - var isHeRetired=false
  - var dateOfRetiring=null
  - var numberOfRetired=undefined

```
// Data Types
// Java scriptte String,Number,Boolean,
//Null ve undefined olmak üzere 5 primitive data tipi vardır

var middleName='Berk'
//String

var dateOfDay=25
//Number '25' yazarsak matematiksel işlemlerde kullanılamaz

var isHeRetired=false
//Boolean false ise değil true ise öyle anlamı verir

var yearsOfRetiring=null
//herhangi bir değer atamadan kullanılır

var numberOfRetired=undefined
//Teknik bir data tanımsız olduğu için
//kullanılmaması gerekitğini ifade eder.
```

DATA TYPES

Temel Veri Tipleri

1. **String (Dizi)**: Metin verilerini temsil eder. Tek tırnak ('') veya çift tırnak ("") içinde yazılır.

```
let ad = "Ahmet";
let soyad = 'Yılmaz';
```

2. **Number (Sayı)**: Matematiksel sayıları temsil eder.

```
let yas = 25;
let fiyat = 19.99;
```

3. **Boolean (Mantıksal)**: Sadece iki değeri temsil eder: `true` (doğru) ve `false` (yanlış).

```
let dogruMu = true;
let yanlisMi = false;
```

DATA TYPES

Özel Veri Tipleri

1. **Undefined**: Tanımsız bir değişkenin değeridir.

```
let tanimsizDegisken;
```

2. **Null**: Hiçbir şeyi temsil etmez.

```
let degerSifir = null;
```

DATA TYPES

Veri Türü Belirleme

```
let isim = "Ali";
console.log(typeof isim); // Çıktı: string
```

Data Casting

```
let sayi = 10;
let metinSayi = String(sayi);
console.log(typeof metinSayi); // Çıktı: string
```

Concatenation & Interpolation

1. Birleştirme (Concatenation)

Birleştirme, birden fazla metni veya değeri tek bir dize içinde birleştirmek için kullanılan bir işlemdir. JavaScript'te, string birleştirme için '+' operatörü kullanılır.

```
var firstName = "John";
var lastName = "Doe";
var fullName = firstName + " " + lastName;

console.log(fullName); // "John Doe"
```

Concatenation & Interpolation

2. Ekleme (Interpolation)

Ekleme, bir dizenin içinde dinamik olarak değerleri veya ifadeleri yerleştirmek için kullanılan bir tekniktir. Bu, daha okunaklı ve esnek kod oluşturmanıza olanak tanır. JavaScript'te, ekleme için Template Literals (`) kullanılır.

.

```
var name = "Jane";
var age = 30;
var message = `My name is ${name} and I am ${age} years old.`;

console.log(message); // "My name is Jane and I am 30 years old."
```

Wise Quarter

Objects

JavaScript

JavaScript'te nesneler, özellikleri (properties) ve davranışları (methods) içeren karmaşık veri yapılarıdır. Nesneler, çeşitli veri türlerini (string, sayı, dizi, vb.) içerebilir ve genellikle gerçek dünya nesnelerini temsil etmek için kullanılır.

Nesne Oluşturma

Nesneler, {} süslü parantezler arasına özelliklerin ve değerlerin çiftleri şeklinde tanımlanır. Özellikler, anahtar-değer çiftleri şeklinde belirtilir.

```
// Bir öğrenci nesnesi oluşturalım
let ogrenci = {
    ad: "Ahmet",
    yas: 20,
    okul: "ABC Üniversitesi"
};
```

Objects

Özelliklere Erişim

Nesne özelliklerine nokta (.) veya köşeli parantez ([]) notasyonuyla erişilebilir.

```
console.log(ogrenci.ad); // Ahmet
console.log(ogrenci["yas"]); // 20
```

Yeni Özellik Ekleme ve Değiştirme

Bir nesneye yeni özellikler eklenebilir veya varolan özellikler değiştirilebilir.

```
ogrenci.numara = "12345"; // Yeni bir özellik ekleme
ogrenci.ad = "Mehmet"; // Varolan özelliği değiştirme
```

Arrays

Giriş

JavaScript'te Arrays (diziler), birden fazla öğeyi tek bir değişken altında saklamak için kullanılır. Bu öğeler herhangi bir veri tipine sahip olabilir ve dizilerin kendileri de bir veri tipidir.

Dizi Oluşturma

Dizi, köşeli parantez içinde virgülle ayrılmış öğeler listesi olarak tanımlanır.

```
// Boş bir dizi oluşturma
let emptyArray = [];

// Stringlerden oluşan bir dizi oluşturma
let colors = ["red", "green", "blue"];

// Sayılardan oluşan bir dizi oluşturma
let numbers = [1, 2, 3, 4, 5];
```

Arrays

Dizi Elemanlarına Erişim

Dizi elemanlarına, sıfırdan başlayarak indeks numarası kullanılarak erişilir.

```
let colors = ["red", "green", "blue"];
console.log(colors[0]); // Çıktı: "red"
console.log(colors[2]); // Çıktı: "blue"
```

Arrays Methods

JavaScript'te bir dizi üzerinde kullanılabilen birçok yerleşik yöntem bulunmaktadır. Bunlardan bazıları:

- **push():** Dizinin sonuna yeni bir öğe ekler.
- **pop():** Dizinin sonundaki öğeyi kaldırır ve geri döndürür.
- **shift():** Dizinin başındaki öğeyi kaldırır ve geri döndürür.
- **unshift():** Dizinin başına yeni bir öğe ekler.
- **splice():** Dizinin belirli bir konumundan öğeler ekler veya kaldırır.
- **concat():** İki veya daha fazla diziyi birleştirir.

Bu metotlar, dizileri dinamik bir şekilde yönetmemize olanak tanır.

Arrays

Diziye Eleman Ekleme ve Kaldırma

Ekleme

Diziye yeni bir eleman eklemek için push() yöntemini kullanabiliriz.

```
let colors = ["red", "green", "blue"];
colors.push("yellow");
console.log(colors); // Çıktı: ["red", "green", "blue", "yellow"]
```

Arrays

Kaldırma

Diziden son elemanı kaldırmak için pop() yöntemini kullanabiliriz.

```
let colors = ["red", "green", "blue"];
let lastColor = colors.pop();
console.log(lastColor); // Çıktı: "blue"
console.log(colors); // Çıktı: ["red", "green"]
```

Arrays

Dizi Boyutu

Dizinin eleman sayısını length özelliği ile alabiliriz.

```
let colors = ["red", "green", "blue"];
console.log(colors.length); // Çıktı: 3
```

Arrays Methods

shift()

shift() yöntemi, bir dizinin başındaki öğeyi kaldırır ve bu öğeyi döndürür.

```
let colors = ["red", "green", "blue"];
let firstColor = colors.shift();
console.log(firstColor); // Çıktı: "red"
console.log(colors); // Çıktı: ["green", "blue"]
```

Arrays Methods

unshift()

unshift() yöntemi, bir dizinin başına bir veya daha fazla öğe ekler.

```
let colors = ["red", "green", "blue"];
colors.unshift("yellow", "orange");
console.log(colors); // Çıktı: ["yellow", "orange", "red", "green", "blue"]
```

Arrays Methods

splice()

splice() yöntemi, bir dizideki belirli bir konumdan öğeleri ekler veya kaldırır.

```
let colors = ["red", "green", "blue"];
// 1. indexten başlayarak 0 öğe kaldır ve yerine "yellow" ve "orange" ekle
colors.splice(1, 0, "yellow", "orange");
console.log(colors); // Çıktı: ["red", "yellow", "orange", "green", "blue"]

// 2. indexten başlayarak 1 öğe kaldır ve yerine "purple" ekle
colors.splice(2, 1, "purple");
console.log(colors); // Çıktı: ["red", "yellow", "purple", "green", "blue"]
```

Arrays Methods

concat()

concat() yöntemi, iki veya daha fazla diziyi birleştirir ve yeni bir dizi döndürür.

```
let colors1 = ["red", "green"];
let colors2 = ["blue", "yellow"];
let mergedColors = colors1.concat(colors2);
console.log(mergedColors); // Çıktı: ["red", "green", "blue", "yellow"]
```

Operatörler

Operatör Nedir?

JavaScript'te, operatörler değerler arasında işlemler yapmak için kullanılan sembollerdir. Operatörler, atama, aritmetik, karşılaştırma ve mantıksal gibi farklı kategorilere ayrılabilir.

Atama Operatörleri

Atama operatörleri, bir değeri bir değişkene atamak için kullanılır. İşte en yaygın kullanılan atama operatörü:

- **Eşittir (=)**: Bir değeri bir değişkene atar.

```
let e = 10;
let f = 5;

f = e; // f artık 10
```

Operatörler

Aritmetik Operatörler

Aritmetik operatörler matematiksel hesaplamalar yapmak için kullanılır. İşte bazı yaygın aritmetik operatörler:

- **Toplama (+)**: İki değeri toplar.
- **Çıkarma (-)**: Bir değeri diğerinden çıkarır.
- **Çarpma (*)**: İki değeri çarpar.
- **Bölme (/)**: Bir değeri diğerine böler.
- **Mod (%)**: Bir sayının diğerine bölümünden kalanı verir.

```
let x = 10;
let y = 5;

let toplam = x + y; // 10 + 5 = 15
let fark = x - y;   // 10 - 5 = 5
let carpim = x * y; // 10 * 5 = 50
let bolum = x / y;  // 10 / 5 = 2
let kalan = x % y;  // 10 % 5 = 0
```

Operatörler

Karşılaştırma Operatörleri

Karşılaştırma operatörleri, değerler arasında karşılaştırmalar yapmak için kullanılır. İşte bazı karşılaştırma operatörleri:

- **Eşitlik (==)**: İki değerin eşit olup olmadığını kontrol eder.
- **Eşit Değil (!=)**: İki değerin eşit olmadığını kontrol eder.
- **Büyük (>)**: Bir değerin diğerinden büyük olup olmadığını kontrol eder.
- **Küçük (<)**: Bir değerin diğerinden küçük olup olmadığını kontrol eder.
- **Büyük Eşit (>=)**: Bir değerin diğerinden büyük veya eşit olup olmadığını kontrol eder.
- **Küçük Eşit (<=)**: Bir değerin diğerinden küçük veya eşit olup olmadığını kontrol eder.

```
let a = 10;
let b = 5;

console.log(a == b); // false
console.log(a != b); // true
console.log(a > b);  // true
console.log(a < b);  // false
console.log(a >= b); // true
console.log(a <= b); // false
```

Operatörler

Mantıksal Operatörler

Mantıksal operatörler, mantıksal ifadeleri değerlendirmek için kullanılır. İşte bazı mantıksal operatörler:

- **VE (&&)**: İki koşul da doğruysa true döner.
- **VEYA (||)**: Koşullardan en az biri doğruysa true döner.
- **DEĞİL (!)**: Bir koşulu tersine çevirir.

```
let c = true;
let d = false;

console.log(c && d); // false
console.log(c || d); // true
console.log(!c);     // false
console.log(!d);     // true
```

Wise Quarter

Operatörler

JavaScript

Toplama ve Atama (+=): Bir değişkenin değerine belirli bir değeri ekler ve sonucu değişkene atar.

```
let x = 5;
x += 3; // x'in değeri şimdi 8 (x = x + 3 ile aynı)
```

Çıkarma ve Atama (-=): Bir değişkenin değerinden belirli bir değeri çıkarır ve sonucu değişkene atar.

```
let y = 10;
y -= 4; // y'nin değeri şimdi 6 (y = y - 4 ile aynı)
```

Çarpma ve Atama (*=): Bir değişkenin değerini belirli bir değerle çarpar ve sonucu değişkene atar.

```
let z = 3;
z *= 2; // z'nin değeri şimdi 6 (z = z * 2 ile aynı)
```

Bölme ve Atama (/=): Bir değişkenin değerini belirli bir değere böler ve sonucu değişkene atar.

```
let w = 20;
w /= 4; // w'nin değeri şimdi 5 (w = w / 4 ile aynı)
```

Mod ve Atama (%=): Bir değişkenin değerini belirli bir değere böler ve kalanı değişkene atar.

```
let v = 15;
v %= 4; // v'nin değeri şimdi 3 (v = v % 4 ile aynı)
```

Koşullu İfadeler

Koşullu ifadeler(Conditional Statements), bir programın belirli koşullara göre farklı işlemler yapmasını sağlar. JavaScript'te koşullu ifadeler if, else if ve else anahtar kelimeleriyle oluşturulur.

1. if İfadesi

if ifadesi, belirli bir koşul doğru olduğunda bir kod bloğunun çalıştırılmasını sağlar.

```
let sayi = -5;

if (sayi % 2 === 0) {
    console.log("Sayı çift.");
}
```

Koşullu İfadeler

2. else if İfadesi

Birden fazla koşulu kontrol etmek için else if ifadesi kullanılır.

```
let mevsim = "kış";

if (mevsim === "ilkbahar") {
    console.log("Çiçekler açar.");
} else if (mevsim === "yaz") {
    console.log("Hava sıcak.");
} else if (mevsim === "sonbahar") {
    console.log("Yapraklar dökülür.");
} else {
    console.log("Kar yağar.");
}
```

Koşullu İfadeler

3. else İfadesi

else ifadesi, if koşulu yanlış olduğunda çalışacak kod bloğunu tanımlar

```
let sayi = -5;

if (sayi > 0) {
    console.log("Sayı pozitif.");
} else {
    console.log("Sayı negatif veya sıfır.");
}
```

Koşullu İfadeler

SORULAR

1. Bir öğrencinin aldığı notu temsil eden bir sayıyı kullanarak, bu notun harf karşılığını belirleyen bir program yazın.

   91 – 100 arası ise A
   81 – 90 arası ise B
   71 – 80 arası ise C
   61 – 70 arası ise D
   60 ve altı ise F

2. Bir hava durumu uygulaması yazıyorsunuz ve kullanıcıya hava durumunu göstermek istiyorsunuz. Kullanıcıdan hava durumu bilgisini almadan, belirli koşullara göre hava durumunu belirleyen ve kullnıcıya hava Soğuk, hava Ilıman, Hava Sıcak vb. ifadelerle söyleyen bir JavaScript programı yazın.

3. Bir öğrenci notlarına göre sınıf geçip geçmediğini belirlemek istiyor. Öğrencinin sınav notları şu şekildedir: Matematik (mat), Fen Bilimleri (fen) ve Türkçe (tur). Öğrencinin her ders için geçme notu 60'tır. Öğrenci herhangi bir dersden başarısız olursa, sınıfı geçemeyecektir. Öğrencinin geçip geçmediğini belirleyen bir JavaScript programı yazın

JavaScript Fonksiyonlarına Giriş

Fonksiyon

Fonksiyonlar, belirli bir işlevi yerine getirmek için kullanılan kod bloklarıdır. JavaScript'te fonksiyon tanımlamak ve çağırmak oldukça basittir.

Fonksiyon Tanımlama

Fonksiyon tanımlamak için function anahtar kelimesi kullanılır. Bir fonksiyonu tanımlamak için şu şekilde bir yapı kullanılır:

```
function sayHello() {
    console.log("Merhaba!");
}
```

Fonksiyon Çağırma

Bir fonksiyonu çağırmak için sadece fonksiyon adını ve parantezleri kullanmanız yeterlidir:

```
sayHello(); // "Merhaba!" çıktısını verir
```

JavaScript Fonksiyonlarına Giriş

Fonksiyon Scope

JavaScript'te değişkenlerin erişilebilir olduğu alanlara "scope" denir. Fonksiyonlar da kendi scope'larına sahiptir.

```
var x = 10;

function scopeTest() {
    var y = 20;
    console.log(x); // 10
    console.log(y); // 20
}

console.log(x); // 10
console.log(y); // ReferenceError: y is not defined
```

JavaScript Fonksiyonlarına Giriş

Fonksiyon Parametreleri ve Döndürülen Değerler

Fonksiyonlara parametreler geçirerek dinamik davranmalarını sağlayabilir ve döndürülen değerler aracılığıyla sonuçları alabiliriz.

```
function greet(name) {
    console.log("Merhaba, " + name + "!");
}

greet("Ahmet"); // "Merhaba, Ahmet!" çıktısını verir
```

Döndürülen Değerler

Fonksiyonlar, return anahtar kelimesiyle bir değer döndürebilir:

```
function toplam(a, b) {
    return a + b;
}

console.log(toplam(3, 5)); // 8 çıktısını verir
```

JavaScript Fonksiyonlarına Giriş

Anonim Fonksiyon İfadesi

Anonim fonksiyonlar, isimsiz olarak tanımlanabilen ve değişkenlere atanabilen fonksiyonlardır.

```
var sayHello = function() {
    console.log("Merhaba!");
};

sayHello(); // "Merhaba!" çıktısını verir
```

Arrow (Ok) Fonksiyonları (ES6 ve Sonrası)

Ok fonksiyonları, daha kısa ve okunaklı bir syntax sunarlar. ES6 ile birlikte tanıtılmışlardır.

```
const toplam = (a, b) => a + b;

console.log(toplam(3, 5)); // 8 çıktısını verir
```

Döngüler

Döngüler (Loops) Nedir?

Döngüler, bir işlemi tekrarlamak için kullanılan yapısal programlama öğeleridir. JavaScript'te yaygın olarak kullanılan üç tür döngü vardır: for, while, ve do...while. Bu döngüler, belirli koşullara bağlı olarak bir kod bloğunu tekrar tekrar çalıştırmanızı sağlar.

Döngülerin Kullanım Alanları

- Bir dizi elemanı üzerinde işlem yapma
- Belirli bir işlemi belirli bir süre boyunca tekrarlama
- Belirli bir koşul gerçekleşene kadar bir işlemi tekrarlama

Döngülerde Dikkat Edilmesi Gerekenler

- Sonsuz döngü riskini önlemek için döngü içindeki koşulun dikkatlice kontrol edilmesi gerekir.
- Döngü içinde değişkenlerin düzgün bir şekilde güncellendiğinden emin olunmalıdır, aksi takdirde istenmeyen davranışlar olabilir.

Döngüler

1. for Döngüsü

for döngüsü, bir başlangıç değeri, bir bitiş koşulu ve bir artış/değişim adımı belirterek belirli bir aralıktaki tekrarlanan işlemleri yönetir.

```
for (başlangıç; koşul; artış/değişim) {
   // Kod bloğu
}
```

```
for (let i = 0; i < 5; i++) {
    console.log(i); // 0'dan 4'e kadar sayıları yazdırır
}
```

Döngüler

2. while Döngüsü

while döngüsü, bir koşul doğru olduğu sürece belirli bir kod bloğunu çalıştırır.

```
let i = 0;
while (i < 5) {
    console.log(i); // 0'dan 4'e kadar sayıları yazdırır
    i++;
}
```

3. do...while Döngüsü

do...while döngüsü, while döngüsüne benzer, ancak koşul döngünün sonunda kontrol edilir, bu nedenle en az bir kez çalıştırılır.

```
let i = 0;
do {
    console.log(i); // 0'dan 4'e kadar sayıları yazdırır
    i++;
} while (i < 5);
```

Döngüler

switch case İfadeleri Nedir?

switch case ifadeleri, bir değerin birden fazla olası durumuna göre farklı işlemler yapmak için kullanılır. Bu ifadeler, belirli bir değişkenin veya ifadenin değerine bağlı olarak farklı kod bloklarını yürütmek için kullanılır.

- switch: Kontrol edilecek değişkenin veya ifadenin başlatıldığı anahtar kelime.
- case: Belirli durumlar için karşılaştırılacak değerler.
- break: Her case bloğunun sonunda yer alır ve switch yapısını sonlandırır.
- default: Hiçbir case bloğunun eşleşmediği durumlar için varsayılan olarak belirlenen bloktur.

```
let meyve = "elma";

switch (meyve) {
  case "elma":
  case "armut":
    console.log("Yaz meyveleri");
    break;
  case "portakal":
  case "mandalina":
    console.log("Kış meyveleri");
    break;
  default:
    console.log("Bilinmeyen meyve!");
}
```

Class Oluşturma ve Kullanma


```
// MyClass.js
export default class MyClass {
  constructor() {
    // constructor logic
  }

  // methods
}
```

Ve bu sınıfı başka bir dosyada kullanmak için:

```
// main.js
import MyClass from './MyClass.js';

const myInstance = new MyClass();
```