

# Web Yazılım Geliştirme

---

## Ders 01 - Giriş

Erciyes Üniversitesi  
Bilgisayar Mühendisliği Bölümü

**Eğitmen:** Ömür ŞAHİN



# Sunum Planı

1 Giriş

2 JavaScript



# 1-Giriş

---



## Dersin Amacı/İçerik

- Önyüz (frontend) odaklı Web Uygulamaları
- Javascript
- SPA
  - İstemci taraflı uygulamalar
  - React 19+
- RESTful web servislere giriş
  - NodeJS ile sunucu uygulamaları
- Websockets
- Güvenlik
- Typescript



## Dersin Yapısı

- 12 hafta boyunca haftada 2 saat
- Ders dışında 2 saat çalışma ve araştırma



# Gerekli Araçlar

- pnpm / npm / yarn
- Node.js v22+
- Git
- IDE
  - WebStorm kullanılacaktır.
  - İsteyen Visual Studio Code kullanabilir.
- Bash komut terminali
  - Mac/Linux kullananlar mevcut terminallerden birini kullanabilir.
  - Windows için GitBash kullanımı önerilmektedir.
- Modern build tool'ları (Vite)
- Browser dev tools
- AI code asistanları (Github CoPilot)



## 2-JavaScript

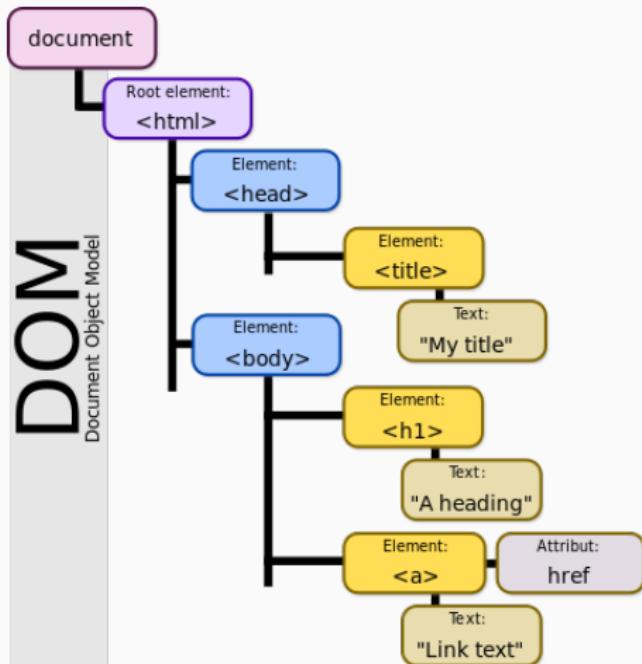
---



- JavaScript ile Java arasında bir bağ bulunmamaktadır.
- Tarayıcıda çalıştırılan programlama dilidir.
  - NodeJS ile birlikte sunucuda da çalıştırılabilir.
- Diğer kaynaklarda (resim veya css dosyaları) olduğu gibi referans verilebilir veya HTML dosyasına doğrudan eklenebilir.
- JS web sayfalarının yapısını ve içeriğini değiştirmek için Document Object Model (DOM) üzerinde işlemler gerçekleştirmektedir.



# JavaScript



Şekil 1: DOM örneği.



- Tarayıcı üzerinde bir kodun yürütülmesi gerekirse günümüzde yaygın olarak JS kullanılmaktadır.
- Geçmişte farklı alternatifleri bulunmaktadır:
  - Java Appletler (artık kullanılmıyor)
  - Flash (Eski web sayfalarında bulunsa da tarayıcılar desteklerini çekmeye başladı)
  - Silverlight (2014 yılında son sürümü yayınlandı)
  - ve diğerleri...

Yukarıdaki alternatiflerin kullanılabilmesi için tarayıcıya harici eklenti yüklenmesi gerekmekteydi.



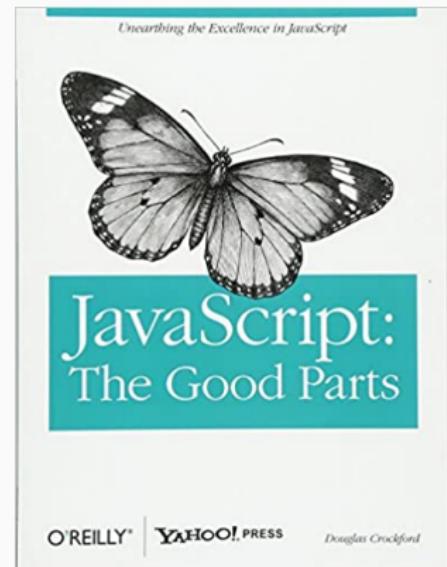
# Günümüzde JavaScript Alternatifleri

- **WebAssembly (WASM)**: C++, Rust, Go ile native performans
- **Modern Transpiler'lar**: TypeScript, SWC, esbuild
- **Compile-to-JS**: Kotlin/JS, ReScript, Elm
- **Meta-framework'leri**: Next.js, Nuxt, SvelteKit



Ancak JavaScript oldukça kötü tasarlanmış bir dildir.

- JavaScript için yazılmış en önemli kitabın adında "**The Good Parts**" geçmektedir.
- TypeScript veya Kotlin gibi JS'e dönüştürülen (transpile) farklı programlama dilleri bulunmaktadır.
- Günümüzde ciddi iyileştirmeler gerçekleşmiştir.
  - TypeScript kullanımı (projelerin %85'inde)
  - V8, SpiderMonkey gibi motor iyileştirmeleri



# JavaScript'in Genel Karakteristiği

- **Yorumlayıcı:** Derlemeye ihtiyaç duymamaktadır.
  - Yüksek performans elde etmek için bazı runtime uygulamaları (tarayıcı gibi) genelde JS kodunu makina koduna derlemektedir.
- **Dinamik Yazım (Dynamic Typed):** Değişken tanımlarken tür belirtmeye gerek yoktur. Ayrıca farklı türde atamalara da izin vermektedir.
- **Zayıf Yazım (Weakly Typed):** Farklı türler arasında "+", "-" gibi operatörleri hata almadan kullanabilirsiniz.



## Yorumlayıcı (Interpreter)

- Tarayıcıya doğrudan kaynak kod verilebilir.
- HTML içerisinde doğrudan gömülebilir veya css, resim dosyaları gibi harici bir dosya import edilebilir.
- Transpile işleminin günümüzdeki en yaygın kullanımı:
  - NPM/Yarn gibi build araçlarının kullanımı
  - React/Angular/Vue gibi kütüphanelerin kullanılması
  - Eski tarayıcıları destekleyecek şekilde dönüşümlerin uygulanması
  - TypeScript kullanarak kodların yazılması gibi...



# Dinamik Yazım

- **var x = 10;**
  - 10 değerine eşit nümerik bir x tanımlaması
  - Tür olarak "numeric" belirtilmesine gerek yoktur.
- **var x = 10; x = 'Merhaba';**
  - Farklı türde atamalara izin vermektedir..
- **x = 10**
  - "var" ve ";" ihmal edilebilir ancak genel öneri bunların kullanılmasıdır.
  - **var** kullanılırsa yerel bir değişken oluşturulur aksi takdirde global olacaktır.
  - ; kullanılmazsa çözümü zor bug'lar ortaya çıkmasına sebep olabilir.



## let/const vs var

- **x = 1** gibi bir tanımda bulunursanız global değişken olacaktır ve bundan sakınmalısınız.
- **var x = 1** *fonsiyon kapsamında* yerel bir değişken oluşturur.
- **let x = 1**, **var**'a benzer ancak blok kapsamında çalışmaktadır.
- **const x = 1**, Java'daki final gibi değiştirilemeyen ve blok içerisinde geçerli yerel değişken oluşturur.
- Yani, let/const kullanılması önerilmektedir.
- Typescript kullanımı önemlidir.

```
1 const userName: string = 'john';
2 const userAge: number = 25;
```



## Zayıf (Weakly) Yazım

- String + sayı → Birleştirme
  - "a" + 1 → "a1"
- String - sayı → Not a Number (NaN)
  - "a" - 1 → NaN
- Boş Object + Boş Dizi → Nümerik 0
  - {} + [] → 0
- Diğer dinamik yazılımlı diller (Python gibi) çalışma zamanında hata fırlatmaktadır.
  - Bu tür programlama dillerine Güçlü (Strongly) Yazım denilmektedir.
- Statik yazılımlı dillerde (Java gibi) kod derlenememektedir.
  - Tek istisna String türünde "+" operatöründür.



## When you mistype `x = obj.fiedl` instead of `x = obj.field`

	Compile time	Runtime
C++	<p>wrong, wrong wrong! stop everything</p>	
Python	<p>hmm maybe that's right?</p>	<p>no, wait, that's wrong!</p>
Javascript	<p>ಠ_ಠ</p>	<p>ಠ_ಠ</p>

## Soru?

Aşağıdaki ifadenin çıktısı ne olabilir?

1 ( 'a' + + 'a' + 'a' + 's' ) . toLowerCase()



# ananas

- '`a'+'a'`' ifadesi '`a'+(+a')`' olarak değerlendirilir.
- `(+a')` string ifadenin içeriğini pozitif sayıya dönüştürmeye çalışır ancak '`a`' sayı olmadığı için **NaN** sonucunu alır.
- '`a'+NaN+'a'+'s'`' yani '`aNaNas`' sonucu olur.
- '`toLowerCase()`' fonksiyonu N harflerini n harfine dönüştürür.



# Soru?

Aşağıdaki ifadenin çıktısı ne olabilir?

$+(![]+![]+![]+![]+[]+(![]+![]))$



# Soru?

42

- []: Boş Dizi
- ![]: Dizinin değili (**false**)
- !![]: !false, yani true
- !![]+!![]: true+true, 1+1
- !![]+!![]+!![]+!![]: 1+1+1+1 = 4

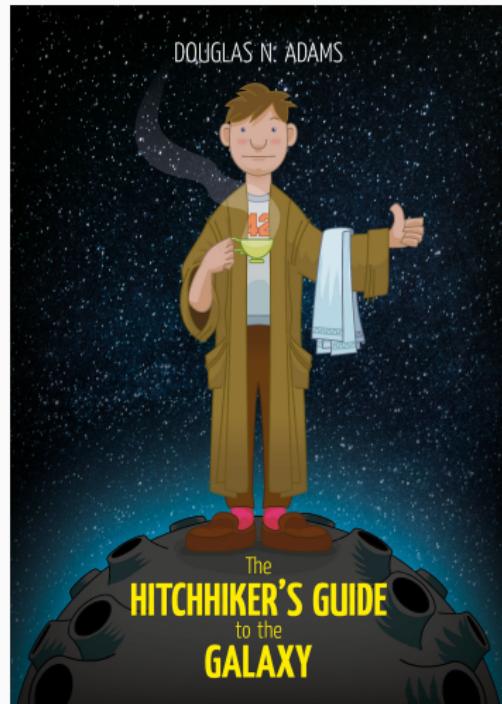


# Soru?

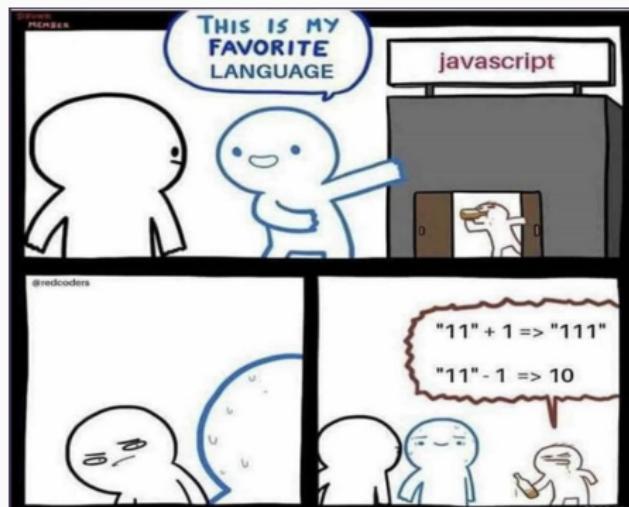
- `!![]+!![]+!![]+!![]+[]`:  $4+[]$  JS string birlestirmesi olarak algilar ve "4" degeri döner.
- `!![]+!![]+!![]+!![]+[]+(!![]+!![])`:  $"4"+2 \rightarrow "42"$
- `+("42")`  $\rightarrow 42$



# Neden 42?



- Ancak Web programlama üzerine geliştirme yapmak istiyorsanız JS mecburen öğrenmeniz gereken bir dildir.
- Modern linting araçlar (ESLint, Prettier) pek çok sorunu önler.
- IDE'ler type checking ile uyarı verir.
- TypeScript olmazsa olmazdır.



Console What's New

Filter

top

> 2 + 2  
< 4  
> "2" + "2"  
< "22"  
> 2 + 2 - 2  
< 2  
> "2" + "2" - "2"  
< 20



# Neden JavaScript?



B  
A  
N  
S

- JavaScript (ve diğer dinamik yazılımlı diller) büyük çaplı projelerde problem yaratmaktadır.
- Bu ders boyunca oldukça küçük projeler üzerinde çalışacağız.
- Küçük projeleri oluşturdukça dinamik yazılımlı diller hakkında tecrübe sahibi olacaksınız.

,

**“Beni öldürmeyen acı  
güçlendirir.”**

*Friedrich Nietzsche*



## Eşitlik için "==" kullanın

- **false==0**
  - Sonuç **true** olacaktır. Boolean false değerinin eşdeğeri nümerik 0'dır.
- **false====0**
  - Sonuç **false** olacaktır. Bool değer nümerik bir değere eşit değildir.
- **0==[]**
  - Sonuç **true** olacaktır. Nümerik 0 değer boş diziyeye eşittir.
  - <https://dorey.github.io/JavaScript-Equality-Table>
- Eşitsizlik için **!=** yerine **!==** kullanın.



# Öneriler

- Compile time kontrol için:

```
1     let age: number = 25;
2     age = "25";  Hata: 'string' tipini 'number'
                  tipine atayamazsin
```

- Nullish coalescing (??)

```
1 const name = inputName ?? "Guest";
2 // inputName null veya undefined ise "Guest"
  atanir.
```

- Optional chaining (?.):

```
1     const city = user?.address?.city;
2 // user veya address undefined/null ise hata
  firlatmaz, city = undefined olur.
```



## Boolean

- 6 değer bool olarak kullanıldığında false olur. Geri kalanların tamamı true değerdir.
  - false
  - 0
  - ""
  - null
  - undefined
  - NaN



## Fonksiyon Tanımı

- function test(){ return 1; }
- topla = function(x,y){return x+y;}
  - topla(2,3), **5** değerini verir.
  - topla("a","b"), "**ab**" değerini verir.
- add = (x,y) => {return x+y}
  - arrow notasyonu olarak geçmektedir.
  - Daha temiz bir gösterim olduğu için genelde bunu kullanacağız.



## Değişken olarak fonksiyonlar

- `function topla(x,y){ return x+y; }`
  - **topla** adında bir fonksiyon oluşturur.
- `x=topla; x(1,2)`
  - **topla** fonksiyonun bütün kodları `x` değişkenine atanır ve ardından `x` fonksiyonu 1,2 giriş değerleri ile çağırılır.
- `x=()=>topla(1,2); x()`
  - Giriş değerine sahip olmayan ve **topla(1,2)** fonksiyonunu çağrıran bir fonksiyon oluşturur. Bu fonksiyonu çağırmak için `()` kullanılır.
- `birEkle = y => topla(y,1); birEkle(5);`
  - `y` girdisi alan yeni bir fonksiyon oluşturur ve bu girdi ile **topla** fonksiyonunu `y,1` giriş değerleri ile çağırır. **birEkle(5)** çağrıSİ ile de **6** değeri elde edilir.



## Yorumlar (Comment)

- Yazmış olduğunuz kodlarda yorum satırları ile açıklamış olmanız oldukça önemlidir.
- Java gibi diğer dillere benzer syntax'a sahiptir.
- Tek satır için: //
- Çoklu satır için: /\* ile başlar \*/ ile sonlandırılır.



## Filter Örneği

- **[-5,4,-3,2].filter(e=>e>0)** komutu **[4,2]** değerlerini döndürür.
- e dizideki her bir elemanı ifade etmektedir.
- İçerideki fonksiyon true veya false değerlerinden birini döndürmek zorundadır.
- Sonuç değeri true olan elemanlardan oluşmaktadır.



## Map Örneği

- `["Radio","Tarifa"].map(e=>e.length)` komutu **[5,6]** değerlerini döndürür.
- Her bir `e` elementi bir başka değere dönüştürüülerek return edilir.
- `["Radio","Tarifa"].map((e,i)=>""+i+"_"+e.length)` komutu **[0\_5,1\_6]** değerlerini döndürür.
- `(e,i)` yerine farklı parametreler de kullanabilirsiniz.



# Map

- const len = d => d.length  
["Les","Tetes","Raides"].map(len)
- Sonuç: [3,5,6]
- len fonksiyonu d adında 1 girişe sahiptir.



## Aşağıdaki ifadenin değeri nedir?

```
["10","10","10"].map(parseInt)
```

- `parseInt("10")` değeri "number" türünde 10 değerini vermektedir.



## Aşağıdaki ifadenin değeri nedir?

[10, NaN, 2]

- parseInt bir string ifade ve taban olmak üzere 2 input değer almaktadır.
- map fonksiyonu (**e,i**) olmak üzere 2 giriş değeri sağlamaktadır. Bunlardan **i** taban olarak kullanılır.
- `parseInt("10",0)====10`
- `parseInt("10",1)====NaN`
- `parseInt("10",2)====2` (binary değer)
- Doğru yazım stili: `["10","10","10"].map(e=>parseInt(e,10))`



# DOM Manipülasyonu

- Document Object Model (DOM): Görüntülenen HTML'in nesne gösterimidir.
- JS kullanımının temel amaçlarından biri DOM'un manipülasyonunun sağlanmasıdır.
- DOM'a erişim için "document" isimli obje çağrırlır.



# DOM Manipülasyonu

```
1 yaziTemizle = function () {  
2   const yaziAlani = document.getElementById("textId");  
3   const sonucAlani = document.getElementById("resultId");  
4   yaziAlani.value = '';  
5   sonucAlani.value = '';  
6 };
```

- DOM objelerine erişmenin en kolay yolu id ile gerçekleşmektedir.
- HTML özelliği olarka id alanının atanması gerekmektedir.

```
1 <textarea id="textId"></textarea>
```



- Bir sayfada JS çalıştırmanın çeşitli yöntemleri vardır.
- En basit yöntemlerinden biri HTML tagları arasında event handler'a doğrudan fonksiyonu vermektedir.

1    `<div onclick="yaziTemizle()">Temizle</div>`

- Event Handler:
  - onclick, onchange, onmouseover, onmouseout, onkeydown, vs.
  - [https://www.w3schools.com/js/js\\_events.asp](https://www.w3schools.com/js/js_events.asp)



# Derleme ve Çalıştırma

The screenshot shows a browser window with developer tools open. The left side displays a portion of a web page with text and a green 'Next >' button. The right side shows the developer tools interface with the 'Console' tab selected. The console output is as follows:

```
> false == 0
< true
> false === 0
< false
> 0 == []
< true
> 0 != []
< false
> 0 !== []
< true
> tuple = (x,y) => x*y;
< (x,y) => x*y
> tuple("a","b")
< "ab"
> tuple(2,1)
< 3
> ["10","10","10"].map(parseInt)
< [2, NaN, 2]
> ["10","10","10"].map(e=>parseInt(e,10))
< [10, 10, 10]
```



I want to do something.

ten events are detected.

is, with **JavaScript code**, to be added to HTML elements.