

Web Programlama

7. Hafta

04.11.2024



Mühendislik
Fakültesi
Bilgisayar Mühendisliği

Hazırlayan: Dr. Ercan Ezin

GİRİŞ

CSS Konumlandırma, Javascript Giriş

Not: Bu dersin içeriđi kaynakçada belirtilen materyallerde derlenerek üretilmiştir.

CSS: POSITIONS

CSS için 4 farklı konumlandırma değişkeni vardır.

- STATIC
- FIXED
- ABSOLUTE
- RELATIVE

CSS : STATIC

- Referans noktası yoktur. Öntanımlı olarak sayfada hareket etmez.

```
<body>
  <h1>Puppy</h1>
  <p>A puppy is a juvenile dog. Some puppies
  <h2>Development</h2>
  <p>At first, puppies spend the large major
  <div id="box1"></div>
</body>
```

```
#box1 {
  height: 100px;
  width: 100px;
  background-color: red;

  top: 0;
  left: 0;
}
```

Puppy

A puppy is a juvenile dog. Some puppies can weigh 1–3 lb up to 15–23 lb (6.8–10.4 kg). All healthy puppies grow quickly change as the puppy grows older, as is commonly seen in vernacular English, puppy refers specifically to dogs, while such as seals, giraffes, guinea pigs, or even rats.

Development

At first, puppies spend the large majority of their time sleeping pile together into a heap, and become distressed if separated by even a short distance.

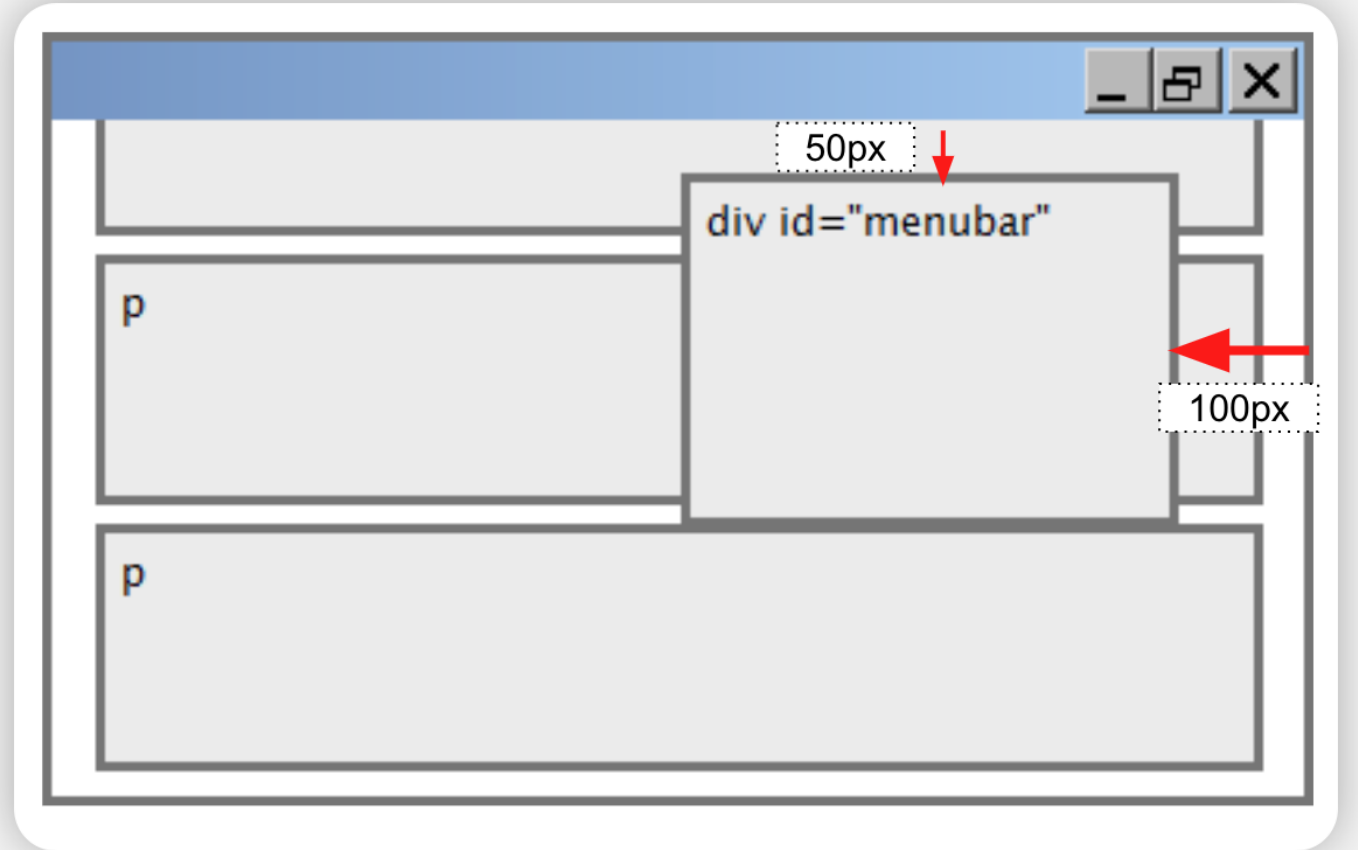


Örnek: [Link](#)

CSS: FIXED

- Viewport içerisinde hizalandırılmış alana göre konumlanır.

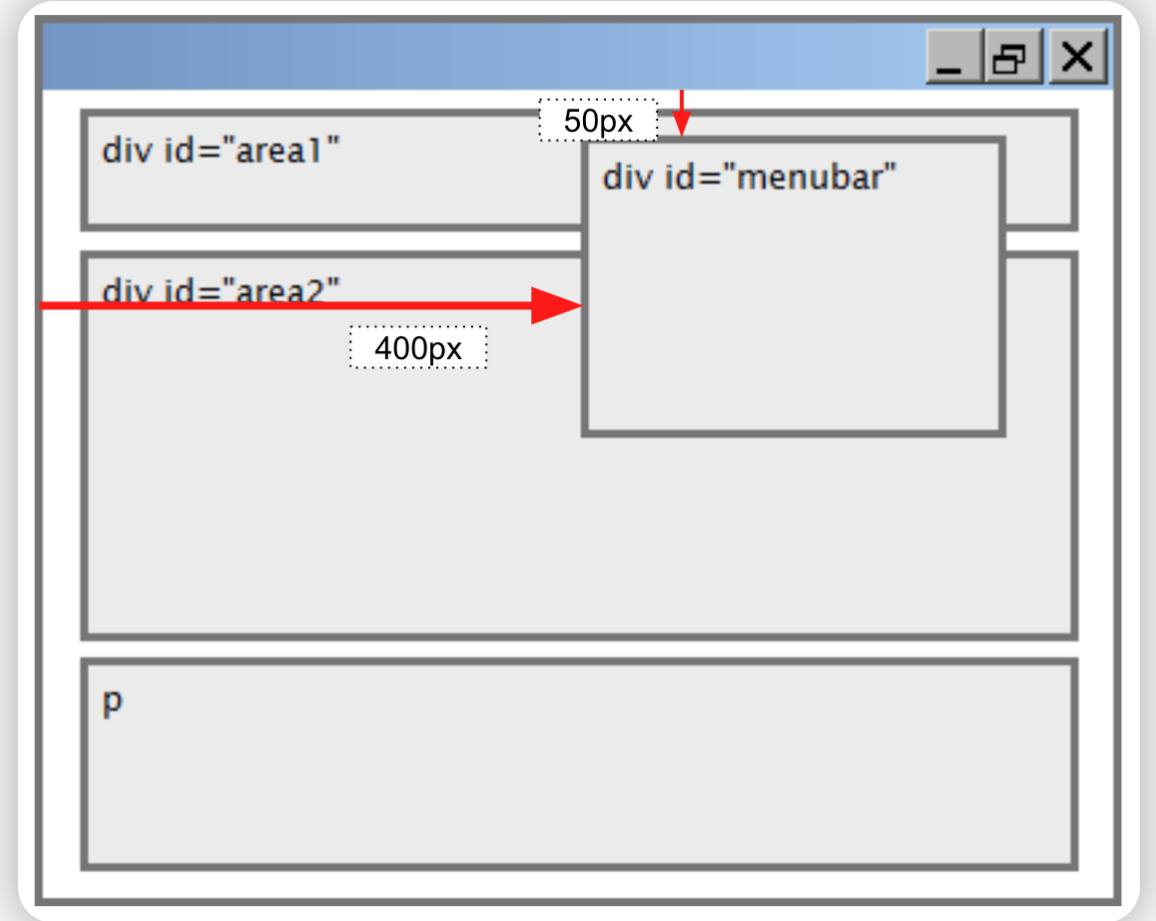
```
#menubar {  
  position: fixed;  
  top: 50px;  
  right: 100px;  
}
```



CSS : ABSOLUTE

- Bir elemanın içinde bulunduğu elemana göre konumlandırmaya yarar.

```
#menubar {  
  position: absolute;  
  left: 400px;  
  top: 50px;  
}
```



Örnek : [Link](#)

CSS: RELATIVE

- Bir elemanın normalde olması gereken yere göre göreceli yeni lokasyonudur.

```
#box2 {  
  height: 100px;  
  width: 100px;  
  background-color: blue;  
  
  position: relative;  
  top: 50px;  
  left: 50px;  
}
```

Puppy

A puppy is a juvenile dog. Some puppies can weigh 1–3 lb (0.45–1.36 kg), while larger ones can weigh up to 15–23 lb (6.8–10.4 kg). All healthy puppies grow quickly after birth. A puppy's coat color may change as the puppy grows older, as is commonly seen in breeds such as the Yorkshire Terrier. In vernacular English, puppy refers specifically to dogs, while pup may often be used for other mammals such as seals, giraffes, guinea pigs, or even rats.

Development

At first, puppies spend the large majority of their time sleeping and the rest feeding. They instinctively pile together into a heap, and become distressed if separated from physical contact with their littermates, by even a short distance.



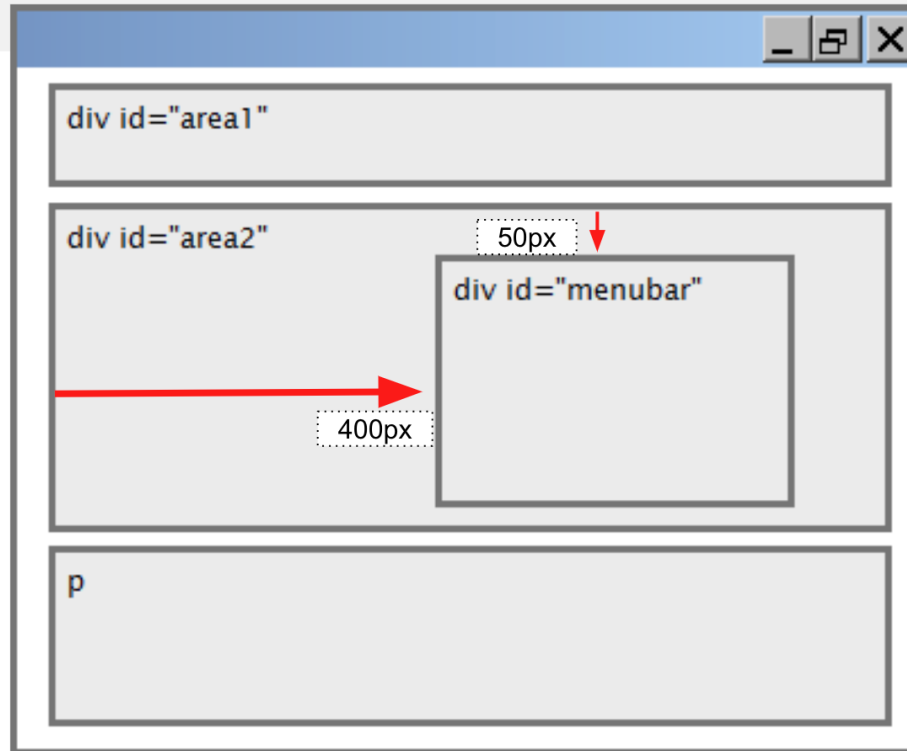
Örnek: [Link](#)

CSS : RELATIVE ABSOLUTE

- Normalde absolute olarak belirlenmiş bir eleman, eğer başka bir eleman içerisinde yer alıyorsa(parent), absolute lokasyonunu içeren elemana atamak için içeren elemana position:relative tanımlanır. (parent relative olunca child absolute olabilir.)

CSS: RELATIVE ABSOLUTE

```
#area2 {  
  position: relative;  
}  
  
#menubar {  
  position: absolute;  
  left: 400px;  
  top: 50px;  
}
```



CSS : PRACTICE

- **Mozilla Developer Network (MDN):** [Link](#)
- **W3 School:** [Link](#)
- **FLEXBOX FROGGY:** [Link](#)
- **CSS ZEN GARDEN:** [Link](#)
- **CSS TRICKS:** [Link](#)
- **GRID BY EXAMPLE:** [Link](#)

...

JAVASCRIPT

JAVASCRIPT (JS)

- JavaScript bir programlama dilidir(1995).
- Şu an tarayıcıda ön tanımlı olarak çalışabilen tek programlama dilidir.
- Web sitelerinde animasyon veya programlama(text manipülasyonu, kullanıcı yönlendirme vs.) gerektiren işlemler için en ideal dildir.
- Java dili ile pazarlama amacı dışında bir ilgisi yoktur.


JAVASCRIPT: IN HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>WEBP</title>
    <link rel="stylesheet" href="style_sheet.css" />
    <script src="dosyaadi.js"></script>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

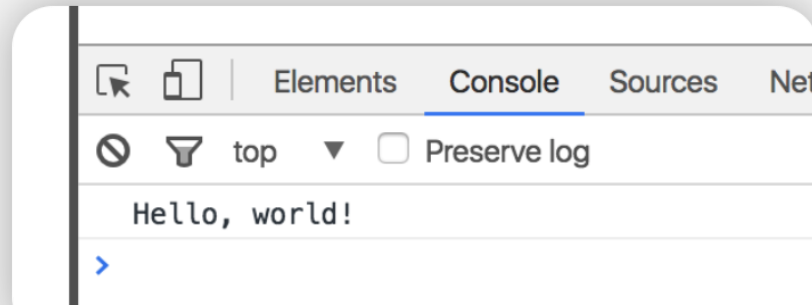
JAVASCRIPT: PRINT FUNC

- Javascript'te bir mesaj yazdırmak istiyorsak `console.log` kullanabiliriz. Derleme gerektirmez. Main metodu olmadan yukarıdan aşağıya doğru çalışır.

Örnek:

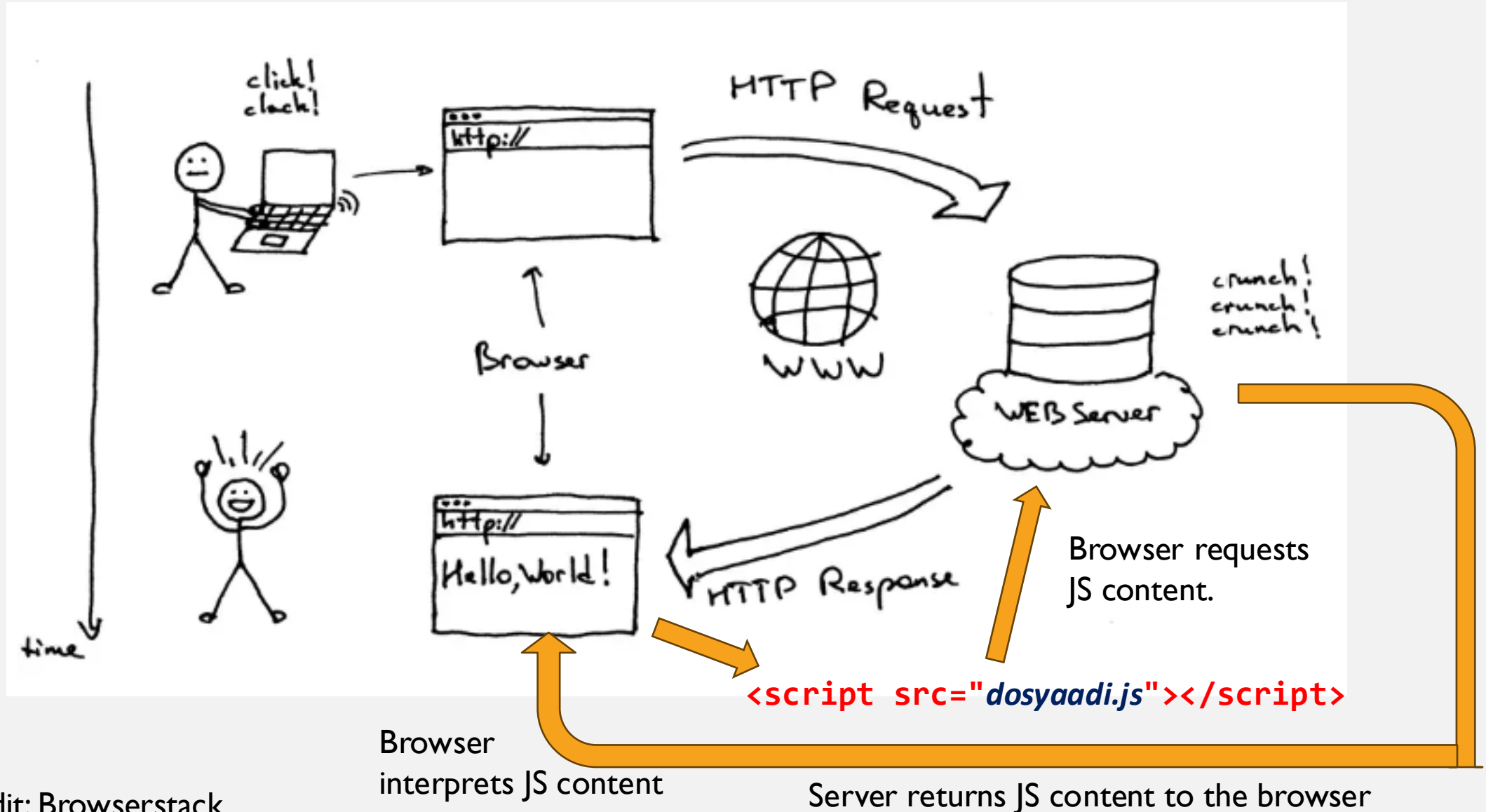
`dosyaadi.js`  `console.log('Bu mesajı konsola yolladım!')`

Chrome tarayıcınızda console sekmesinde görüntüleyebilirsiniz.



Console sekmesi aynı zamanda interaktif olarak komutları yorumlar

HOW JS WORKS



JS CONSOLE PRACTICE

- **ÖRNEK UYGULAMA**

<https://web.harran.edu.tr/bilgisayar/tr/>

Yukarıda bulunan websitesinde yayınlanan duyuru başlıklarını konsola yazdıralım.

Çözüm:

```
document.querySelectorAll('.block-news .title a').forEach(link=> console.log(link.innerText));
```


JS LANG PROPERTIES

for-loops:

```
for (let i = 0; i < 5; i++) { ... }
```

while-loops:

```
while (notFinished) { ... }
```

comments:

```
// comment or /* comment */
```

conditionals (if statements):

```
if (...) {  
    ...  
} else {  
    ...  
}
```

JS FUNCTIONS

Javascript fonksiyonu şöyle tanımlanır;

```
function fonksiyonAdi()  
{  
  işlemler;  
  işlemler;  
  ...  
}
```

HOW JS FUNCTION WORKS

script.js

```
function hello() {  
  console.log('Hello!');  
  console.log('Welcome to JavaScript');  
}
```

```
hello();  
hello();
```

Önce fonksiyon tanımlanır. →

Sonra fonksiyon çağrılır →

SORU: Önce fonksiyon çağrılırsa ne olur?

Cevap: Bazı durumlarda çalışabilir(hoisted). Ama bu durumdan kaçınılmalıdır.

```
⊘ 🔍 top ▼  
Hello!  
Welcome to JavaScript  
Hello!  
Welcome to JavaScript  
> |
```

Console output

JS : VARIABLES

- JS içerisinde bir değişken tanımlamak için aşağıdaki 3 yöntemi durumuna göre kullanabilirsiniz.
 - `var plaka = 63; // Fonksiyon aralığında değişken`
 - `let ilce = 'Haliliye'; // Blok aralığında değişken`
 - `const isStudent = true; // Blok aralığında sabit-
// Yeniden atanamaz`

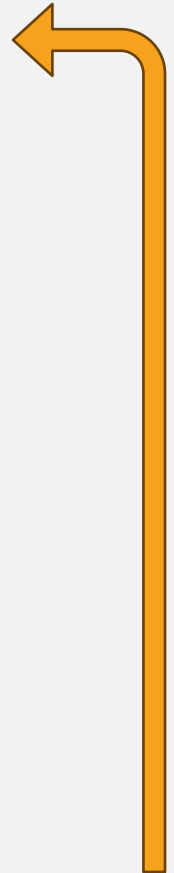
UYARI: JS için veri tipini bir değişkeni kullanmadan tanımlayamazsınız.

JS: FUNCTIONS

```
function printMessage(mesaj, tur) {  
  for (var i = 0; i < tur; i++) {  
    console.log(mesaj);  
  }  
}
```


Fonksiyon parametreleri var, let, yada const, içermez.

Bknz:



JS: VAR SCOPE

```
function printMessage(mesaj, tur) {  
  for (var i = 0; i < tur; i++) {  
    console.log(mesaj);  
  }  
  console.log('i : ' + i);  
}  
  
printMessage('hello', 3);
```



Fonksiyon Aralığı

Yukarıdaki kodun çıktısı nedir?

HATIRLATMA: `i` değişkeni `var` olarak tanımlandığı için **fonksiyon** aralığında yeniden erişilip kullanılabilir. Bu yüzden `hello*3` ve `i: 3` çıktısı alınır.

JS: LET SCOPE

```
function printMessage(mesaj, tur) {  
  for (let i = 0; i < tur; i++) {  
    console.log(mesaj);  
  }  
  console.log('i : ' + i);  
}
```



Blok Aralığı

```
printMessage('hello', 3);
```

Yukarıdaki kodun çıktısı nedir?

HATIRLATMA: `i` değişkeni `let` olarak tanımlandığı için sadece **blok** aralığında yeniden erişilip kullanılabilir. Bu yüzden `i` dışarıda ulaşılmak istendiğinde **hata** alınır.


JS : LET

```
let y = 10;  
y = 0;           // HATASIZ  
y++;            // HATASIZ  
let list = [1, 2, 3];  
list.push(4);   // HATASIZ
```

let türünden değişkenler yeniden atanabilir ve güncellenebilir.

JS : CONST SCOPE

```
function printMessage(mesaj, tur) {  
  for (const i = 0; i < tur; i++) {  
    console.log(mesaj);  
  }  
  console.log('i : ' + i);  
}  
  
printMessage('hello', 3);
```

 Blok Aralığı

Yukarıdaki kodun çıktısı nedir?

HATIRLATMA: `i` değişkeni `const` olarak tanımlandığı için sadece **blok** aralığında yeniden erişilip kullanılabilir. Bu yüzden `i` dışarıda ulaşılmak istendiğinde `let` gibi **hata** alınır.

JS : CONST

```
const sehir = 63;  
y = 34;           // hata!  
y++;             // hata!  
const list = [1, 2, 3];  
list.push(4);    // HATASIZ!
```

const ile tanımlanmış değişkenlerin genelde değeri değişmez ancak yapısı itibariyle güncellenebilen **list** gibi değişkenlerin elemanları değiştirilebilir.

JS : VARIABLES

- Genel olarak **const** kullanmaya özen gösterin. JS Kaynak yönetimi için verimli bir pratiktir.
- Eğer bir değişkenin değeri ilerde değişecekse **let** kullanmak en iyisidir.
- **var** değişken tipi kullanmaktan elinizden geldikçe **kaçının**. İnternette **var** kullanan bayağı çok kaynak olsa bile **let** ve **const** yeni değişken tipleri olduğu için kuralına göre uygun olanı kullanın.

JS : TYPES

JavaScript değişkenlerinin türleri yoktur, ancak değerlerin türleri vardır.

Altı temel veri türü vardır:

- **Boolean:** true ve false
- **Number:** Tüm sayılar **ondalıklıdır** (tamsayı yoktur) Ör: 63.0
- **String:** 'tek tırnak' veya "çift tırnak" içinde
- **Null:** "bir değere sahip değil" anlamına gelen bilerek atanmış bir değer
- **Undefined:** Değeri atanmamış bir değişkenin değeri.

Ör: let c; şeklinde tanımlanmış bir değer için => c: undefined

Ayrıca, **Nesne (Object)** türleri de vardır; bunlara **Dizi (Array)**, **Tarih (Date)**, **String** (temel tür için nesne sarmalayıcı) gibi türler dahildir.

JS : TYPES

- Numbers:

```
const oran = 0.45;
```

```
const sayi = 100*oran. // sonuc sayi = 45.0 olur.
```

- Strings:

```
let yemek = 'cig'
```

```
yemek+='kofte'
```

```
console.log('Urfada '+ yemek + 'yenir.')
```

String değerleri tek veya çift tırnak ile tanımlanabilir.

String değeri oluşturulduğunda bir daha değiştirilemez, karakter eklenirse her seferinde yeni bir string oluşturulur. (Immutable)

JS : TYPES

- Boolean:

```
let isStudent = true;
let isAdult = age > 18 ;
if (isStudent && !isAdult){
    işlemler...
}
```

Boolean **true** ve **false** değeri alır ve **&&**, **||** , yada **!** ifadeleriyle birlikte kullanılabilir.

JS : BOOLEAN

- Kontrol ifadelerinde boolean olmayan ifadeler kullanabilirsiniz. Bunlar doğrusu(truthy) yada yanlışımı(falsy) ifadelerle çevrilerek kullanılır.

```
if (kullaniciadi) {  
    // kullaniciadi değişkeni tanımlıysa yapılacak işlemler  
}
```

JS : EQUALITY CHECK

- JS kontrol ifadelerinde eşitlik kontrolü çok sağlıklı çalışmamakta.

```
' ' == '0' // false
' ' == 0 // true
0 == '0' // true
NaN == NaN // ??? false
[''] == '' // true
false == undefined // false
false == null // false
null == undefined // true
```

```
' ' === '0' // false
' ' === 0 // false
0 === '0' // false
NaN == NaN // halen false
[''] === '' // false
false === undefined // false
false === null // false
null === undefined // false
```

Bu yüzden her zaman === ve !== kullanmaya çalışın. Bu ikiliyi kullanmaktan kaçınin: == yada !=

JS : ARRAYS

Arrays tipi bir objedir ve bir data listesi oluşturmaya yarar.

```
// boş bir liste oluşturmak için
let list = [];
let alisveris = ['patlican', 'domates'];
alisveris[1] = 'biber';
```

Array tipinde indeks 0'dan başlar.

Değiştirilebilir(Mutable) bir tiptir ve length özelliği ile Array tipinin içerdiği data miktarı kontrol edilebilir. Ör: console.log(alisveris.length)

JS : PRACTICE

W3 School online partik imkanı sunuyor.

<https://www.w3schools.com/js/default.asp>

MIDTERM EXAM

- Çoktan seçmeli
- 30 soru
- 90 dakika. 10:30-12:00 arası.
- 11.11.2024 tarihinde E-204 nolu sınıfta.
- Slaytlarda verilen her şeyden sorumlusunuz.
- Sorunuz varsa bana ulaşabilirsiniz.

- **BAŞARILAR**



EOF

REFERENCES / CREDITS

1. Stepp M, Miller J, Kirst V. Web Programming Step by Step. Step by Step Publishing; 2012.
2. CS193X Web Programming Fundamentals Course Slides at Stanford Uni by Victoria Kirst
3. <https://medium.com/swlh/an-introduction-to-git-and-github-22ecb4cb1256>