

MAKÜ|BTF

BUCAK TEKNOLOJİ FAKÜLTESİ

Web Tasarımı ve Programlama

Ders 4:
JavaScript Next-Gen

Dr. Öğr. Üyesi Elvan DUMAN

eduman@mehmetakif.edu.tr
Yazılım Mühendisliği Bölümü

- **JavaScript'e Giriş**

- JavaScript'in Tanımı ve Önemi
- Nerede Kullanılır

- **JavaScript Temelleri**

- Çıktı Yöntemleri
- Yazım Kuralları ve İfadeler

- **Değişkenler ve Veri Türleri**

- let, const
- Veri Türleri (String, Number, Boolean, Object, Array vb.)

- **Fonksiyonlar**

- Fonksiyon Tanımı ve Kullanımı
- Arrow Fonksiyonlar

- **Nesneler (Objects)**

- Nesne Oluşturma ve Özellikler
- Metotlar ve Erişim Yöntemleri

- **Diziler (Arrays)**

- Dizi Tanımlama ve Elemanlara Erişim
- Dizi Metotları (forEach, push vb.)

- **Spread ve Rest Operatörleri**

- **JavaScript**, web sayfalarına **etkileşim ve dinamik davranış** kazandıran programlama dilidir.
- **HTML** yapıyı, **CSS** görünümü, **JavaScript** ise **işlevi (davranışı)** tanımlar.
- Tüm modern tarayıcılarda **yerleşik olarak çalışır**; ek kurulum gerektirmez.
- **Olay yönetimi, form doğrulama, animasyonlar, veri işleme ve API entegrasyonu** gibi işlevleri sağlar.
- **Tarayıcı (frontend)** tarafında çalışır, ancak **Node.js** ile **sunucu (backend)** tarafında da kullanılabilir.

- JavaScript kodu **<script> ... </script>** etiketleri arasına yazılır.
- **type="text/javascript"** artık gereksizdir; varsayılan dil JavaScript'tir.
- Kod, **<head>**, **<body>** veya her ikisine de yerleştirilebilir.
- **Body'nin sonunda** yerleştirmek sayfa yüklenme hızını artırır.
- JavaScript fonksiyonları, **buton tıklama gibi olaylar (events)** ile çağrılabilir.
- Kodlar **harici dosyalarda (.js)** da saklanabilir:
 - `<script src="myScript.js"></script>`
- **Harici script** kullanmanın avantajları:
 - HTML ve kod ayrılığı sağlar
 - Sayfa bakımını kolaylaştırır
 - Tarayıcı önbelleğiyle yükleme hızını artırır.
- Harici dosyalar şu yollarla çağrılabilir:
 - Tam URL → <https://.../myScript.js>
 - `/js/myScript.js`

- JavaScript veriyi farklı şekillerde gösterebilir:
 - HTML içinde (innerHTML / innerText)
`document.getElementById("demo").innerHTML = "Hello World";`
 - Sayfa çıktısına yazma (document.write): Kullanıcıya bilgi mesajı gösterir.
 - `document.write(5 + 6);`
 - Uyarı kutusu (alert)
 - `alert("Hello JavaScript");`
 - Tarayıcı konsolu (console.log) : Hata ayıklama ve test amaçlı kullanılır.
 - `console.log("Debug message");`

- Programların nasıl yazılacağını belirleyen kurallardır.

```
let x = 5;  
let y = 6;  
let z = x + y; // Toplama örneği
```

- Değişken Tanımlayıcılar

- Harf, _ veya \$ ile başlar
- Sayı içerebilir ama **ilk karakter sayı olamaz**
- Büyük/küçük harf duyarlıdır
- let, const gibi anahtar kelimeler kullanılamaz

Bilgisayara yapılacak işlemleri adım adım anlatan komutlardır.

```
- let x, y, z;    // Değişken tanımlama  
- x = 5; y = 6;  // Değer atama  
- z = x + y;     // İşlem
```

Temel Özellikler

- JavaScript programları birçok **statement** içerir.
- **Sırasıyla (üstten alta)** çalıştırılır.
- Her ifade; **değerler, operatörler, anahtar kelimeler, ifadeler ve yorumlardan** oluşur.

```
- document.getElementById("demo").innerHTML = "Merhaba Dünya!";
```

- **Noktalı virgül (;)** ifadeleri ayırır. let a = 5; let b = 6; let c = a + b;

Kod Blokları

```
function myFunction() {  
    document.getElementById("demo1").innerHTML = "Hello!";  
    document.getElementById("demo2").innerHTML = "How are you?";  
}
```

Sık Kullanılan Anahtar Kelimeler

Keyword	Açıklama
var, let, const	Değişken tanımı
if, switch	Koşul ifadeleri
for	Döngü
function, return	Fonksiyon tanımı ve dönüşü
try	Hata yakalama

Yöntem	Açıklama	Öneri
const	Değeri değişmeyen sabit	✅ kullan
let	Değeri sonradan değiştirilebilir	✅ kullan
var	Eski yöntem (function-scope)	⚠️ Kullanma
Otomatik (x=5 gibi)	Tanımsız değişken oluşturur	❌ Kötü uygulama Kullanma

- let : **Block Scope** sağlayan değişken tanımlayıcısıdır.
- **Kural 1:** Blok içinde ({}) tanımlanan değişken **dışarıdan erişilemez**.
- **Kural 2:** let ile tanımlanan değişken **tekrar tanımlanamaz (redeclaration ❌)**
- **Kural 3:** Kullanılmadan önce **tanımlanmalıdır**.

```
{  
  let x = 2;  
}  
// ❌ x burada erişilemez, var kullanılırsa erişilir
```

Özellik	var	let
Scope	Global / Function	Block {}
Redeclare	✅	❌
Hoisting	Evet	Hayır
Kullanım Önerisi	Eski (kullanma)	Modern

- `const` ifadesi, değeri değiştirilemeyen bir sabit tanımlar.
- Tekrar tanımlanamaz.
- Tekrar değer atanamaz.
- Blok kapsamına (block scope) sahiptir.
- Tanımlandığında mutlaka başlangıç değeri verilmelidir.
 - `const PI = 3.14; // doğru`
 - `PI = 3.15; // hata`

Veri Türü	Açıklama
String	Tırnak içinde yazılan metin verisidir.
Number	Sayısal değeri temsil eder.
BigInt	Çok büyük tamsayılar için kullanılır.
Boolean	true veya false değerlerini tutar.
Object	Anahtar-değer çiftlerinden oluşan yapıdır.
Undefined	Henüz değer atanmamış değişkendir.
Null	Boş ya da olmayan nesneyi temsil eder.

```
let name = "Elvan";           // String
let age = 35;                  // Number
let big = 1234567890123456789n; // BigInt
let isStudent = false;        // Boolean
const person = {name: "John"}; // Object
const cars = ["Saab", "Volvo"]; // Array (object türü)
let x;                        // Undefined
let y = null;                 // Null
```

- Javascript Operatörler, Koşul Cümleleri, Döngüler, objects, string ve fonksiyonlarından sorumlusunuz.
- <https://www.w3schools.com/js>

- Fonksiyonlar, belirli bir görevi yerine getirmek için yazılmış, yeniden kullanılabilir kod bloklarıdır.
- Kodun düzenli, modüler ve verimli olmasını sağlar. Bir fonksiyon yalnızca **çağrıldığında (invoke edildiğinde)** çalışır.
- Parantez içinde isteğe bağlı parametreler bulunur. Fonksiyon return ifadesiyle değer döndürebilir.

```
function myFunction(p1, p2) {  
  return p1 * p2;  
}
```

```
function toCelsius(f) {  
  return (5/9) * (f - 32);  
}  
let value = toCelsius(77);
```

Arrow Fonksiyonu

Kısa yazım biçimidir ve kullanımı yaygınlaşmıştır. Return kısaltılabilir.

- `let multiply = (a, b) => a * b;`

Eğer fonksiyon parametre almıyorsa parantez kullanılır.

- `const logMessage = () => console.log("JavaScript is fun!");`

Fonksiyon tek bir parametre alıyorsa parantez kullanılmayabilir.

```
// Parantezli
const square = (x) => x * x;
```

```
// Parantezsiz
const square = x => x * x;
const greet = name => "Hello " + name;
const double = n => n * 2;
```

Bir **object**, birden fazla veriyi (özellikleri ve işlevleri) tek bir yapıda tutan değişkendir.

Nesneler, **anahtar-değer (key-value)** çiftlerinden oluşur.

- `const car = {type: "Fiat", model: "500", color: "white"};`

Temel Kavramlar

- **Property (özellik):** Bir nesnenin bilgisini tanımlar.
Örnek: `car.color = "white"`
- **Method (yöntem):** Nesne üzerinde işlem yapan fonksiyondur.
Örnek: `car.start()`

Nesne Oluşturma Yöntemleri

```
const person = {  
  firstName: "Ayşe",  
  lastName: "Doğan",  
  age: 50  
};
```

Boş nesneye sonradan özellik ekleme

```
const person = {};  
person.name = "John";  
person.age = 30;
```

new Object() kullanımı

- `const person = new Object({name: "John", age: 30});`

Özelliklere erişim

```
person.age;           // Nokta gösterimi  
person["age"];        // Köşeli parantez gösterimi
```

Nesne Oluşturma Yöntemleri

```
const person = {  
  firstName: "John",  
  lastName: "Doe",  
  fullName: function() {  
    return this.firstName + " " + this.lastName;  
  }  
};  
  
console.log(person.fullName()); // () parantezleri unutmayın
```

Dizi Oluşturma Yöntemleri

Array, birden fazla veriyi tek bir değişkende saklamaya yarayan özel bir nesne (object) türüdür.

```
const cars = ["Saab", "Volvo", "BMW"];
```

Temel Özellikler

- **Element (eleman):** Her bir veri.
- **Ordered:** Elemanlar sıralıdır.
- **Zero Indexed:** İlk eleman index 0'dır.
- **Dynamic Size:** Boyut değişebilir.
- **Heterogeneous:** Farklı veri tipleri içerebilir.

Neden Array Kullanılır?

Tek tek değişken tanımlamak yerine, bir listeyi tek değişkende tutmak için kullanılır. Örneğin 300 araba ismini saklamak için 300 değişken yerine tek bir array yeterlidir.

Dizi Tanımlama

```
const cars = ["Saab", "Volvo", "BMW"]; // Dizi literal yöntemi
const cars = []; // Boş dizi
cars[0] = "Saab"; cars[1] = "Volvo";
const cars = new Array("Saab", "Volvo"); // new Array (önerilmez)
```

Elemanlara Erişim

```
let car = cars[0]; // İlk eleman
cars[0] = "Opel"; // Değer değiştirme
```

Diziyi String'e Çevirme

```
const fruits = ["Banana", "Apple", "Mango"];
fruits.toString(); // "Banana,Apple,Mango"
```

Dizinin Uzunluğu

```
const fruits = ["Banana", "Apple", "Mango"];
let length = fruits.length;
```

For Döngüsüyle Dolaşma

```
const fruits = ["Banana", "Apple", "Mango"];  
for (let i = 0; i < fruits.length; i++) {  
    console.log(fruits[i]);  
}
```

forEach() Metodu

```
const fruits = ["Banana", "Apple", "Mango"];  
fruits.forEach(function(value) {  
    console.log(value);  
});
```

Yeni Eleman Ekleme

```
const fruits = ["Banana", "Apple", "Mango"];  
fruits.push("Lemon");
```

'...' operatörü

Spread (...), mevcut bir dizi veya nesnenin tamamını veya bir kısmını başka bir dizi veya nesneye hızlı bir şekilde kopyalamak için kullanılır.

```
const numbersOne = [1, 2, 3];
const numbersTwo = [4, 5, 6];
const numbersCombined = [...numbersOne, ...numbersTwo];
Console.log(numbersCombined);
```

```
const myVehicle = {
  brand: 'Ford',
  model: 'Mustang',
  color: 'red'
}
const updateMyVehicle = {
  type: 'car',
  year: 2021,
  color: 'yellow'
}
const myUpdatedVehicle = {...myVehicle,
  ...updateMyVehicle}
```

Console

```
[object Object] {
  brand: "Ford",
  color: "yellow",
  model: "Mustang",
  type: "car",
  year: 2021
}
```



Nesne Örneği

```
const person = {name: "John", age: 30};  
const details = {city: "London"};  
const fullInfo = {...person, ...details};  
// {name: "John", age: 30, city: "London"}
```

Fonksiyon Argümanlarında Kullanımı

```
function add(a, b, c) {  
  return a + b + c;  
}  
const numbers = [1, 2, 3];  
console.log(add(...numbers)); // 6
```

Spread

Bir dizi veya nesneyi **tek tek elemanlarına ayırır**.

Genellikle dizileri veya nesneleri birleştirmek için kullanılır.

Bir fonksiyonun parametrelerinde kullanılır ve **belirsiz sayıda argümanı tek bir dizi** içinde toplar.

```
function sum(...numbers) {  
  let total = 0;  
  for (let n of numbers) {  
    total += n;  
  }  
  return total;  
}
```

```
console.log(sum(1, 2, 3, 4)); // 10
```

Spread: **Yaymak (expand)**

Rest: **Toplamak (collect)**

İkisi de ... ile yazılır, ancak ters mantıkla çalışır.

Ders 4 Sunum Sonu

İletişim : eduman@mehmetakif.edu.tr

