BÖLÜM 2 — JAVASCRIPT

🌐 5. JavaScript Nedir?

- Web sayfalarına **etkileşim** kazandıran programlama dilidir
- **Tarayıcıda** çalışır (ekstra bir program gerekmez)
- HTML → İskelet | CSS → Görünüm | **JS → Davranış**

Tarayıcıda Kullanma

```
<!-- HTML içine yerleştirme -->
<script>
  alert("Merhaba Dünya!");
</script>

<!-- Dışarıdan dosya bağlama (önerilen) -->
<script src="app.js"></script>
```

Console Kullanımı

```
console.log("Mesaj yazar");          // Tarayıcı konsoluna yaz
console.log(42, "metin", true);      // Birden fazla değer
console.error("Hata mesajı!");       // Kırmızı hata mesajı
console.warn("Uyarı!");              // Sarı uyarı mesajı
console.table([1, 2, 3]);            // Tablo olarak göster
```

> 💡 **Konsolu aç:** F12 → Console sekmesi

📦 6. Değişkenler

let — const — var Farkları

	`var`	`let`	`const`
Kapsam	Fonksiyon	Blok `{}`	Blok `{}`

	var	let	const
Yeniden atama	✅ Evet	✅ Evet	❌ Hayır
Yeniden tanımlama	✅ Evet	❌ Hayır	❌ Hayır
Hoisting	✅ Var	❌ Yok	❌ Yok
Kullanım	⚠️ Eski kod	✅ Değişen değer	✅ Sabit değer

```
// const — değer değişmez
const PI = 3.14;
const isim = "Ahmet";
// PI = 5;  ← HATA verir!

// let — değer değişebilir
let yas = 20;
yas = 21;  // ✅ Geçerli

// var — artık önerilmez
var eskiYontem = "kaçının";
```

> 💡 **Kural:** Önce `const` kullan, değer değişecekse `let` kullan. `var` kullanma!

🔢 7. Veri Tipleri

```
// String — Metin
let ad = "Ayşe";
let mesaj = 'Merhaba';
let sehir = `İstanbul`;            // Template literal (önerilen)

// Number — Sayı
let yas = 25;
let fiyat = 19.99;
let negatif = -5;

// Boolean — Mantıksal
let aktif = true;
let silinmis = false;

// null — Kasıtlı boş değer
let kullanici = null;
```

```
// undefined — Tanımlanmamış
let belirsiz;
console.log(belirsiz);  // undefined

// Object — Nesne (anahtar:değer çiftleri)
let kisi = {
  ad: "Mehmet",
  yas: 30,
  aktif: true
};
console.log(kisi.ad);    // "Mehmet"
console.log(kisi["yas"]); // 30

// Array — Dizi
let meyveler = ["elma", "armut", "kiraz"];
console.log(meyveler[0]);    // "elma"
console.log(meyveler.length); // 3
```

typeof — Tip Kontrolü

```
typeof "merhaba"  // "string"
typeof 42         // "number"
typeof true       // "boolean"
typeof undefined  // "undefined"
typeof null       // "object"  ← JS'nin bilinen hatası!
typeof {}         // "object"
typeof []         // "object"
```

➕ 8. Operatörler

Aritmetik Operatörler

```
5 + 3   // 8  — Toplama
10 - 4  // 6  — Çıkarma
3 * 4   // 12 — Çarpma
15 / 4  // 3.75 — Bölme
15 % 4  // 3  — Kalan (modulo)
2 ** 3  // 8  — Üs alma (2³)
```

```
// Kısa yazım
let x = 10;
x += 5;  // x = 15
x -= 3;  // x = 12
x++;     // x = 13 (1 artır)
x--;     // x = 12 (1 azalt)
```

Karşılaştırma Operatörleri

```
5 == "5"   // true  — Değer eşit (tip kontrolü YOK)
5 === "5"  // false — Değer VE tip eşit (önerilen!)
5 != "5"   // false
5 !== "5"  // true
10 > 5     // true
10 >= 10   // true
3 < 7      // true
```

> ⚠️ **Her zaman `===` kullan!** `==` tip dönüşümü yapar ve beklenmedik sonuçlar çıkabilir.

Mantıksal Operatörler

```
true && true    // true  — VE (ikisi de true olmalı)
true && false   // false
true || false   // true  — VEYA (biri true yeterli)
false || false  // false
!true           // false — DEĞİL (tersine çevirir)
!false          // true
```

🔀 9. Koşullar

if / else if / else

```
let not = 75;

if (not >= 90) {
  console.log("Pekiyi");
} else if (not >= 70) {
```

```
  console.log("İyi");
} else if (not >= 50) {
  console.log("Geçer");
} else {
  console.log("Kaldı");
}
```

Ternary Operator (Üçlü Operatör)

```
// Sözdizimi: koşul ? doğruysa : yanlışsa
let yas = 18;
let durum = yas >= 18 ? "Yetişkin" : "Çocuk";
console.log(durum); // "Yetişkin"

// Daha karmaşık örnek
let fiyat = 100;
let indirimli = fiyat > 50 ? fiyat * 0.9 : fiyat;
console.log(indirimli); // 90
```

🔄 10. Döngüler

for Döngüsü

```
// Sözdizimi: for (başlangıç; koşul; artış)
for (let i = 0; i < 5; i++) {
  console.log(i);  // 0, 1, 2, 3, 4
}

// Geriye doğru
for (let i = 5; i > 0; i--) {
  console.log(i);  // 5, 4, 3, 2, 1
}
```

while Döngüsü

```
let sayac = 0;

while (sayac < 3) {
```

```
  console.log("Sayaç:", sayac);
  sayac++;
}
// Sayaç: 0
// Sayaç: 1
// Sayaç: 2
```

for...of — Dizi Elemanları

```
const meyveler = ["elma", "armut", "kiraz"];

for (const meyve of meyveler) {
  console.log(meyve);
  // elma
  // armut
  // kiraz
}
```

for...in — Nesne Özellikleri

```
const araba = { marka: "Toyota", model: "Corolla", yil: 2022 };

for (const anahtar in araba) {
  console.log(anahtar, ":", araba[anahtar]);
  // marka : Toyota
  // model : Corolla
  // yil : 2022
}
```

🔧 11. Fonksiyonlar

Function Declaration (Fonksiyon Bildirimi)

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

```
console.log(topla(3, 5));  // 8
console.log(topla(10, 20)); // 30
```

Function Expression (Fonksiyon İfadesi)

```
const carp = function(a, b) {
  return a * b;
};

console.log(carp(4, 5));  // 20
```

Arrow Function (Ok Fonksiyonu) — ES6

```
// Tek satır — return otomatik
const kare = (x) => x * x;
const selamla = (isim) => `Merhaba, ${isim}!`;

// Çok satır — {} ve return gerekir
const hesapla = (a, b) => {
  let sonuc = a + b;
  return sonuc * 2;
};

console.log(kare(5));           // 25
console.log(selamla("Zeynep")); // Merhaba, Zeynep!
console.log(hesapla(3, 4));     // 14
```

Parametreler ve Varsayılan Değerler

```
function karsilaData(isim, yas = 18) {
  return `${isim}, ${yas} yaşında`;
}

console.log(karsilaData("Ali", 25));  // Ali, 25 yaşında
console.log(karsilaData("Veli"));     // Veli, 18 yaşında (varsayılan)
```

🗂️ 12. Array Metodları

```
const sayilar = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

.map() — Her Elemanı Dönüştür

```
// Her elemanı 2 ile çarp
const ikiKati = sayilar.map((sayi) => sayi * 2);
console.log(ikiKati);
// [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

// İsimler dizisini büyük harfe çevir
const isimler = ["ali", "veli", "ayşe"];
const buyuk = isimler.map((isim) => isim.toUpperCase());
// ["ALİ", "VELİ", "AYŞE"]
```

.filter() — Koşula Uyanları Al

```
// Sadece çift sayılar
const ciftler = sayilar.filter((sayi) => sayi % 2 === 0);
console.log(ciftler);  // [2, 4, 6, 8, 10]

// 5'ten büyük sayılar
const buyukler = sayilar.filter((sayi) => sayi > 5);
console.log(buyukler);  // [6, 7, 8, 9, 10]
```

.find() — İlk Eşleşeni Bul

```
// 5'ten büyük ilk sayı
const ilkBuyuk = sayilar.find((sayi) => sayi > 5);
console.log(ilkBuyuk);  // 6

// Dizi içinde nesne arama
const kullanicilar = [
  { id: 1, ad: "Ahmet" },
  { id: 2, ad: "Mehmet" },
  { id: 3, ad: "Ayşe" },
];
```

```
const bulunan = kullanicilar.find((k) => k.id === 2);
console.log(bulunan);  // { id: 2, ad: "Mehmet" }
```

.reduce() — Tek Değere İndirge

```
// Tüm sayıları topla
const toplam = sayilar.reduce((birikim, sayi) => birikim + sayi, 0);
console.log(toplam);  // 55

// Maksimum değeri bul
const maks = sayilar.reduce((max, sayi) => sayi > max ? sayi : max, 0);
console.log(maks);  // 10
```

.forEach() — Her Eleman İçin İşlem Yap

```
// Değer DÖNDÜRMEZ, sadece işlem yapar
sayilar.forEach((sayi, index) => {
  console.log(`${index}. eleman: ${sayi}`);
});
// 0. eleman: 1
// 1. eleman: 2
// ...
```

Karşılaştırma Tablosu

Metod	Ne döner?	Ne işe yarar?
`map()`	Yeni dizi	Her elemanı dönüştür
`filter()`	Yeni dizi	Koşula uyanları filtrele
`find()`	Tek eleman	İlk eşleşeni bul
`reduce()`	Tek değer	Diziyi tek değere indir
`forEach()`	undefined	Her eleman için işlem yap

💡 Hızlı Başvuru — Özet

```
// DEĞİŞKENLER
const SABIT = "değişmez";
let degisen = "değişebilir";

// VERİ TİPLERİ
"metin"   42   true   null   undefined   {}   []

// KOŞUL
if (koşul) { } else if (koşul2) { } else { }
koşul ? doğruysa : yanlışsa

// DÖNGÜLER
for (let i = 0; i < n; i++) { }
while (koşul) { }
for (const eleman of dizi) { }
for (const anahtar in nesne) { }

// FONKSİYON
const f = (param) => sonuç;
function f(param) { return sonuç; }

// DİZİ METODLARI
dizi.map(x => ...)       // dönüştür
dizi.filter(x => ...)    // filtrele
dizi.find(x => ...)      // bul
dizi.reduce((acc, x) => ..., başlangıç)  // topla
dizi.forEach(x => ...)   // uygula
```

📚 *Daha fazla kaynak: MDN Web Docs | Tailwind Docs*