

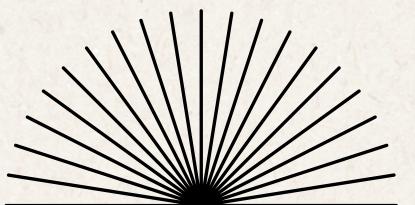


# JAVASCRIPT ES6+ ÖZELLİKLERİ VE REACT KULLANIMI

**Modern JavaScript ile React Geliştirme**

**NAME OF PROJECT:**  
JavaScript React

**PRESENTED BY:**  
Hatice Büşra Arslan



# Konu Başlıklar



Sunumumuzda 3 ana başlığa yer vereceğiz, bunlar:



## Spread ve Rest Operatörleri

- Spread (Yayma Operatörü)
- Rest (Toplama Operatörü)

## Arrow Functions ve Fonksiyon Temelleri

- Arrow Functions (`=>`)
- Default Parameters

## Array Iteration (Dizi Döngüleri)

- `.map()`
- `.filter()`
- `.reduce()`
- `.forEach()`

# Spread ve Rest Operatörleri

## 1) Neden Spread & Rest Kullanıyoruz?

- TeReact'te immutability (değişmezlik) çok önemlidir.
- Veriyi doğrudan değiştirmek hatalara yol açar.
- Daha okunabilir ve modern kod yazarız.

## 2) Spread (Yayma) Operatörü Nedir?

- Bir diziyi veya objeyi parçalayıp yayar.
- Yeni bir dizi / obje oluşturur.
- Orijinal veriye dokunmaz.

```
1 const sayilar = [1, 2, 3];
2 const yeniSayilar = [...sayilar, 4];
```

## 3) React'te Spread Kullanımı

- Mevcut state korunur.
- Yeni state üretilir.
- React doğru şekilde yeniden render eder.

```
1 setItems( [...items, newItem]);
2 items.push(newItem); // ✗
```

## 4) Spread Ne Tür Kopya Yapar?

- Spread operatörü shallow copy yapar.
- İç içe objelerde referans korunur.

```
1 const user = {
2   name: "Ali",
3   address: { city: "Ankara" }
4 };
5 const newUser = { ...user };
```

- ! address hâlâ aynı referansı gösterir.

# Spread ve Rest Operatörleri

## 5) Rest (Toplama) Operatörü Nedir?

- Gelen değerleri tek bir yapı altında toplar.
- Genellikle fonksiyon parametrelerinde kullanılır.

```

1  function topla(...sayilar) {
2    return sayilar.reduce((a, b) => a + b, 0);
3 }
```

## 6) Rest & Destructuring

```

1 const product = {
2   id: 1,
3   name: "Laptop",
4   price: 30000,
5   stock: 5
6 };
7 const { id, ...digerBilgiler } = product;
```

- id ayrıldı.
- Geri kalan her şey digerBilgiler içinde.
- React'te props ayrıştırmada çok kullanılır.

## 7) Spread vs Rest (Karşılaştırma)

SPREAD	REST
Yayma yapar.	Toplama yapar.
Veri üretir.	Veri toplar.
Dizi / Obje kopyalama.	Fonksiyon parametreleri.
State güncelleme.	Destructuring.

# Arrow Functions ve Fonksiyon Temelleri

## 1) Fonksiyon Nedir?

- Tekrar kullanılabilen kod bloklarıdır.
- Bir işi yapar, gerekirse değer döndürür.
- JavaScript'in temel yapı taşlarından biridir.

```
1 function topla(a, b) {  
2   return a + b;  
3 }
```

## 2) Function Declaration

```
1 function selamla(isim) {  
2   return "Merhaba " + isim;  
3 }
```

- `function` anahtar kelimesi kullanılır.
- `this` bağlamı dinamiktir.

## 3) Arrow Function (=>) Nedir?

- ES6 ile gelmiştir.
- Daha kısa ve okunabilir sözdizimi.
- Kendi `this` bağlamını oluşturmaz, dış scope'taki `this`'i kullanır.

```
1 const selamla = (isim) => {  
2   return "Merhaba " + isim;  
3 }
```

## 4) Klasik vs Arrow (Karşılaştırma)

```
1 // Klasik  
2 function kareAl(x) {  
3   return x * x;  
4 }  
5  
6 // Arrow  
7 const kareAl = (x) => x * x;
```

# Arrow Functions ve Fonksiyon Temelleri

## 5) Implicit Return (Gizli return)

- Tek satırlık fonksiyonlarda kullanılır.
- {} ve return yazılmaz.

```
1 const carp = (a, b) => a * b;
```

## 6) Arrow Function ve this

```
1 function klasik() {  
2   console.log(this);  
3 }  
  
4  
5 const arrow = () => {  
6   console.log(this);  
7 };
```

- Klasik fonksiyon → kendi this'ini oluşturur.
- Arrow function → tanımlandığı scope'taki this'i kullanır.

## 7) React'te Arrow Function Kullanımı

- this karmaşasını önler.
- Daha temiz event handler'lar.
- Modern React kodunun standartıdır.

```
1 const handleClick = () => {  
2   console.log("Tıklandı");  
3 };
```

## 8) Default Parameters

- Parametre gelmezse hata oluşmaz.
- Güvenli kod yazmamızı sağlar.

```
1 function selamla(isim = "Ziyaretçi") {  
2   console.log("Merhaba " + isim);  
3 }
```

# Arrow Functions ve Fonksiyon Temelleri



## 9) React'te Default Parameters

```
1 function Profile({ resim = "default.png" }) {  
2   return <img src={resim} />;  
3 }
```

- API'den veri gelmezse uygulama çökmez.

## 10) Ne Zaman Arrow, Ne Zaman Klasik?

DURUM	KULLAN
React component	Arrow
Event handler	Arrow
Basit yardımcı fonksiyon	Arrow
this gereken yer	Klasik

# Array Iteration (Dizi Döngüleri)

## 1) Array Iteration Nedir?

- Diziler üzerinde tek tek dolaşmamızı sağlar.
- Her eleman için işlem yapabiliriz.
- ES6 ile gelen metodlar sayesinde:
  - 1.Daha okunabilir.
  - 2.Daha güvenli.
  - 3.Immutable yapıya uygundur.

## 2) Neden Array Metotları Kullanıyoruz?

- Neden map, filter, reduce?
  - 1.for döngüsüne göre daha temiz.
  - 2.Orijinal diziyi değiştirmez.
  - 3.React'te state yönetimi için idealdir.
- React → for değil, map sever.

## 3) .map() Nedir?

- Bir diziyi alır.
- Her elemanı dönüştürür.
- Yeni bir dizi döndürür.

```
1 const sayilar = [1, 2, 3];
2 const kareler = sayilar.map((sayi) => sayi * sayi);
3 // [1, 4, 9]
```

## 4) React'te .map() Kullanımı

```
1 const users = ["Büşra", "Hatice", "Ali"];
2
3 return (
4   <ul>
5     {users.map((user) => (
6       <li key={user}>{user}</li>
7     )));
8   </ul>
9 );
```

- Listeleri ekrana basmanın standart yoludur.

# Array Iteration (Dizi Döngüleri)

## 5) .filter() Nedir?

- Bir koşula göre elemanları seçer.
- Koşulu sağlayanlar kalır.
- Yeni bir dizi döndürür.

```
1 const sayilar = [1, 2, 3, 4, 5];
2 const ciftler = sayilar.filter(sayi => sayi % 2 === 0);
3 // [2, 4]
```

## 6) React'te .filter() Kullanımı

```
1 const products = [
2   { name: "Laptop", stock: 3 },
3   { name: "Mouse", stock: 0 }
4 ];
5
6 const stoktaOlanlar = products.filter(
7   (urun) => urun.stock > 0
8 );
```

- “Stokta olmayan ürünleri gizle” senaryosu.

## 7) .reduce() Nedir?

- Diziyi tek bir değere indirger.
- En esnek ama en karmaşık metottur.
- (accumulator, current) ile çalışır.

```
1 const fiyatlar = [100, 200, 300];
2
3 const toplam = fiyatlar.reduce(
4   (acc, cur) => acc + cur,
5   0
6 ); // 600
```

## 8) React'te .reduce() Kullanımı

```
1 const sepet = [
2   { price: 100 },
3   { price: 200 }
4 ];
5 const toplamTutar = sepet.reduce(
6   (acc, item) => acc + item.price,
7   0
8 );
```

- Alışveriş sepeti toplamı.

# Array Iteration (Dizi Döngüleri)

## 9) .forEach() Nedir?

- Dizi üzerinde dolasır.
- Değer döndürmez.
- Yan etki (side effect) için kullanılır.

```

1 const sayilar = [1, 2, 3];
2 sayilar.forEach((sayi) => {
3   console.log(sayi);
4 });

```

ÖZELLİK	map	forEach
Yeni dizi döndürür	✓	✗
React'te kullanılır	✓	✗
Yan etki	✗	✓

SENARYO	KULLAN
Listeyi dönüştürme	map
Elemanları filtreleme	filter
Toplam/ tek değer	reduce
Log / yan etki	forEach

# **Beni Dinledığınız İçin Teşekkür Ederim!**

<b>e-mail</b>	arslanhbusra@gmail.com
<b>linkedin</b>	<a href="https://l24.im/zpdt">https://l24.im/zpdt</a>
<b>github</b>	arslanhbusra
<b>medium</b>	arslanhbusra