19.11.2025 Ders Notları

Meta Android Developer Professional Certificate - Ders Notları
Tarih: 19.11.2025

Bu notlar, **Meta Android Developer Professional Certificate** programındaki derslerden alınmıştır. İçerdiği konular:

- **Programlama Paradigmaları** (Fonksiyonel Programlama vs. Nesne Yönelimli Programlama)
- **Fonksiyonel Programlama (FP)** Temelleri
- **Rekürsif Fonksiyonlar**
- **Scope (Kapsam)** ve **Scope Chain**
- **Değişken Tanımlama**: `var`, `let`, `const` karşılaştırması

Programlama Paradigmaları

- İnsan dilleri gibi, programlama dillerinin de farklı stilleri var. Buna **programlama paradigmaları** deniyor.
- İki yaygın paradigm:
  - **Fonksiyonel Programlama (FP)**
  - **Nesne Yönelimli Programlama (OOP)**
- Hiçbiri diğerinden üstün değil, sadece farklı yaklaşımlar.

Fonksiyonel Programlama (FP)

- **Veri ve fonksiyonlar ayrı tutulur**. Veri, fonksiyonların dışında var olabilir.
- Fonksiyonlara veri **argüman** olarak geçilir, işlem yapılır ve **return** ile döndürülür.
- **OOP'de ise** veri ve fonksiyonlar birleştirilip **nesneler** oluşturulur.

Örnek: Para Birimi Dönüşümü

```
let currencyOne = 100;
let currencyTwo = 0;
let exchangeRate = 1.2;

function convertCurrency(amount, rate) {
    return amount * rate;
}

currencyTwo = convertCurrency(currencyOne, exchangeRate);
console.log(currencyTwo); // 120
```

- `convertCurrency` fonksiyonu, veriyi alıp işliyor ve döndürüyor. FP'de mantık bu.

Rekürsif Fonksiyonlar

- **Kendi kendini çağıran fonksiyonlardır**.
- **Dikkat**: Sonsuz döngüye girebilir!
- Örnekte, bir sayaç ekleyip durma koşulu koyuyoruz:

```
let counter = 3;

function example() {
    console.log(counter);
    counter = counter - 1;
    if (counter === 0) {
        return;
    }
    example(); // Kendi kendini çağırıyor
}

example(); // 3, 2, 1 yazdırır ve durur
```

- **Recursion**, döngüler olmadan tekrarlı işler yapmak için kullanışlı.

Scope (Kapsam)

- **Hangi kodun nereye erişebileceğini** belirler.
- **Global Scope**: Fonksiyonların dışındaki kod. Her yerden erişilebilir.
- **Local/Function Scope**: Sadece tanımlandığı fonksiyon içinden erişilebilir.
- **Scope Chain**: Her fonksiyon, parent scope'una referans tutar.

Two-Way Mirror Analojisi:

- Dışarıdakiler içeriyi göremez (global scope → local scope erişemez).
- İçeridekiler dışarıyı görebilir (local scope → global scope erişebilir).

Block Scope & var, let, const

- **ES6 ile `let` ve `const` geldi, block scope kavramı eklendi**.
- **Block Scope**: `{}` içinde tanımlanan değişkenler sadece o blokta erişilebilir.

Özellik	`var`	`let`	`const`
Tekrar tanımlanabilir	Evet	Hayır	Hayır
Değer değiştirilebilir	Evet	Evet	Hayır*

Özellik	`var`	`let`	`const`
Hoisting	Evet (undefined)	Hayır (ReferenceError)	Hayır
Scope	Function	Block	Block

**const* ile primitive değerler değiştirilemez, ama objelerin içeriği değiştirilebilir.*

Önemli Kurallar:

- `var` ile aynı değişkeni tekrar declare edebilirsin, hata vermez.
- `let` ve `const` **declare edilmeden kullanılamaz**.
- `const` **declare edilirken değer atanmalı**.

Pro Tip:

- Değer **değişecekse** → `let`
- Değer **sabit kalacaksa** → `const`
- `var` kullanma, modern JS'de `let` ve `const` tercih et.
