Platforma özgü kod nasıl yazılır? (Platform.OS kullanımı)

React Native'de platforma özgü kod yazmak için Platform API'sini kullanabilirsin. Bu API, hangi platformda çalıştığını kontrol ederek Android, iOS ve Web gibi platformlara göre farklı bileşenler, stiller veya fonksiyonlar yazmanı sağlar. Özellikle Platform.OS özelliği ile hangi platformda çalıştığını anlayarak ona göre kod yazabilirsin.

**Platform Spesifik Dosyalar**

Platforma özgü kod yazmanın diğer bir yolu da platforma özgü dosyalar oluşturmaktır. Örneğin, aynı bileşeni iOS ve Android için ayrı ayrı yazmak istiyorsan:

1. MyComponent.ios.js (iOS için)
2. MyComponent.android.js (Android için)

Bu şekilde dosyaları adlandırarak, React Native hangi platformda çalışıyorsa o dosyayı otomatik olarak kullanacaktır.

Bu yaklaşımlar, aynı projede platforma özel davranışları kolayca yönetmeni sağlar.

Platforma özgü özelleştirmeler nasıl yapılır?

### 1. Platform.OS ile Platforma Özgü Özelleştirmeler

Platform.OS, hangi platformda çalışıldığını öğrenmemizi sağlar ve bu bilgiye göre koşullu işlemler yapabiliriz. İşte bu yöntemi kullanarak stil, bileşen veya işlevleri platforma özgü hale getirebiliriz.

**Açıklama:** Platform.select() ile her platforma göre farklı mesaj ve stil ayarlanmıştır. Böylece Platform.OS kullanarak koşul yazmaya gerek kalmadan daha düzenli bir yapı elde edilir.

### 3. Platforma Özel Dosyalar

Platforma özgü bileşenler için ayrı dosyalar oluşturabilirsin. React Native, dosya isimlerinde .ios.js veya .android.jskullanarak platforma göre dosyayı otomatik seçer.

#### Örnek:

* ButtonComponent.ios.js - iOS’a özel buton bileşeni
* ButtonComponent.android.js - Android’e özel buton bileşeni

**Açıklama:** Aynı bileşeni farklı dosyalar içinde platforma özgü olacak şekilde tanımlayabilirsin. React Native çalıştığı platforma uygun dosyayı otomatik seçer.

### 4. Platforma Özgü API'ler ve Özellikler

Bazı cihaz özellikleri veya API'ler sadece belirli platformlarda bulunur. Örneğin, ActionSheetIOS sadece iOS'ta vardır. Android'de bu tarz bir bileşen olmadığında alternatif bir yöntem kullanabilirsin.

**Açıklama:** ActionSheetIOS sadece iOS'ta çalışır, Android'de alternatif olarak ToastAndroid kullanılmıştır.

React Navigation nedir ve neden kullanılır?

**React Navigation**, React Native projelerinde gezinme (navigation) işlemlerini yönetmek için kullanılan popüler bir kütüphanedir. Uygulamalar genellikle birden fazla ekran veya bileşen içerir ve kullanıcılar bu ekranlar arasında geçiş yaparlar. React Navigation, bu geçişleri yönetmek, ekranlar arasında veri paylaşmak ve geri dönüşler gibi gezinme işlemlerini kolaylaştırır.

**Neden Kullanılır?**

1. **Gezinme İşlemlerini Kolaylaştırır**: Birden fazla ekranı olan bir uygulama geliştirdiğinde, kullanıcıların bu ekranlar arasında geçiş yapmasını sağlaman gerekir. React Navigation, bu işlemi yönetmek için kolay bir yapı sunar.
2. **Farklı Gezinti Türlerini Destekler**:
   * **Stack Navigation**: Ekranların üst üste dizildiği (stack) ve önceki ekrana geri dönülebilen bir yapı sağlar. Bu, tipik mobil uygulamalardaki ekran geçişlerini (geri butonuyla önceki sayfaya dönme) simüle eder.
   * **Tab Navigation**: Uygulama içinde alt kısımda (bottom tab) ya da üst kısımda (top tab) sekme tabanlı gezinme sağlar.
   * **Drawer Navigation**: Kullanıcıların ekranın kenarından kaydırarak açtığı bir çekmece (drawer) menüsü oluşturur.
3. **Geri Dönme ve Yönlendirme (Push, Pop, Replace)**: Uygulama içerisinde bir ekrana gitmek, önceki ekrana dönmek, mevcut ekranı yeni bir ekranla değiştirmek gibi işlemleri kolayca yönetir.
4. **Parametre Gönderme ve Veri Paylaşımı**: Ekranlar arasında veri paylaşımı yapılması gerekiyorsa, React Navigation üzerinden bir ekrandan diğerine parametre gönderilebilir.
5. **Platforma Uygun Davranış**: React Navigation, hem iOS hem de Android platformlarında tutarlı bir gezinme deneyimi sağlar ve her iki platforma özel davranışları optimize eder.
6. **Modüler ve Esnek Yapı**: Uygulamanın gezinme mantığını modüler şekilde tasarlamaya olanak tanır, farklı gezinme yapılarını bir arada kullanabilirsin (örn. Stack ve Tab navigasyonunu birlikte kullanma).

Stack Navigator, Tab Navigator, Drawer Navigator nedir?

React Navigation’da kullanılan üç temel gezinme türü vardır: **Stack Navigator**, **Tab Navigator** ve **Drawer Navigator**. Her biri, kullanıcıların ekranlar arasında geçiş yapmalarını farklı yollarla sağlar ve farklı kullanıcı deneyimlerine uygun olarak geliştirilmiştir.

### 1. Stack Navigator

**Stack Navigator**, ekranları üst üste dizilmiş bir yığın (stack) şeklinde organize eder. Bu yığından ekranlar üst üste eklenip, geri dönülerek çıkarılır. Mobil uygulamalarda en sık kullanılan gezinme şeklidir, çünkü kullanıcılar genellikle önce bir ekrana gider ve geri düğmesine basarak önceki ekrana dönerler.

#### Özellikler:

* **Yığın (Stack) mantığı**: Yeni bir ekrana gittiğinde, bu ekran yığının en üstüne eklenir. Geri döndüğünde ise bu ekran çıkarılır.
* **Geçiş animasyonları**: Ekranlar arasında kayma veya fade gibi geçiş animasyonları sunar.
* **Geri gitme özelliği**: Kullanıcı, bir ekrandan önceki ekrana dönebilir.

### 2. Tab Navigator

**Tab Navigator**, ekranlar arasında gezinmeyi sekmeler (tabs) ile organize eder. Kullanıcı uygulamanın alt kısmında (bottom tab) yer alan sekmeleri tıklayarak ekranlar arasında geçiş yapabilir. Bu tarz gezinme, genellikle sosyal medya veya e-ticaret uygulamalarında kullanılır.

#### Özellikler:

* **Sekmeler (Tabs)**: Uygulamanın alt veya üst kısmında bir dizi sekme sunar. Kullanıcı bir sekmeye dokunduğunda ilgili ekran görüntülenir.
* **Sabit gezinme**: Ekranlar arasında geçiş yapıldığında sekmeler sabit kalır, böylece kullanıcı kolayca geri dönebilir.
* **Ekranlar arasında geçiş**: Kullanıcı istediği zaman herhangi bir sekmeye tıklayarak farklı bir ekrana geçiş yapabilir.

### 3. Drawer Navigator

**Drawer Navigator**, ekranlar arasında bir çekmece menüsü (drawer) ile gezinme sağlar. Kullanıcı ekranın sol kenarından sağa doğru kaydırarak bir menü açabilir ve buradan farklı ekranlara geçiş yapabilir. Çekmece menüsü genellikle uygulamanın daha az kullanılan veya ikincil ekranlarına ulaşmak için kullanılır.

#### Özellikler:

* **Yan menü (Drawer)**: Ekranın kenarından kaydırılarak açılan bir menü sunar.
* **Gizli gezinme**: Menü gizlidir ve kullanıcı istediğinde açıp ekranlar arasında geçiş yapabilir.
* **Daha az kullanılan ekranlar için uygun**: Genellikle ayarlar, yardım gibi ekranlara erişim için kullanılır.

Ekranlar arası veri geçişi ve navigation yapısı nasıl kurulur?

React Navigation kullanarak ekranlar arası veri geçişi yapmak oldukça kolaydır. Genellikle iki senaryo vardır:

1. Bir ekrandan diğerine veri göndermek.
2. Gönderilen veriyi diğer ekranda almak ve işlemek.

**Adımlar:**

1. **Veriyi Gönderme**: navigation.navigate fonksiyonu, diğer ekrana gitmek için kullanılırken bir parametre de alabilir.
2. **Veriyi Alma**: Hedef ekranda bu parametreyi alıp işleyebilmek için route objesi kullanılır.