SetOnApplyWindowInsetsListener Fonksiyonu

SetOnApplyWindowInsetsListener fonksiyonu, pencere girintilerini ele almak ve bunlara göre ekranınızın içeriğini ayarlamak için kullanılan bir fonksiyondur. Pencere girdisinden kastımız, ekranın en üstünde bulunan statüs bar veya ekranın en altında bulunan navigasyon bardır. Bu fonksiyonu ViewCompat sınıfı üzerinden çağrıyoruz. ViewCompat sınıfı, view içindeki özelliklere erişebilmemiz için bize yardımcı olan bir yapıdır.

Aktivite içindeki uygulamamızın başlama noktası olan onCreate fonksiyonu içinde bu şekilde çağrıyoruz.

```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
    insets ^setOnApplyWindowInsetsListener
}
```

Buradaki yapılara odaklanalım. setOnApplyWindowInsetsListener fonksiyonunu 2 tane parametre bekliyor. Birincisi View tipinde ikincisi ise bir interface olan OnApplyWindowInsetsListener yapısıdır.

Üstte belirttiğim gibi arka planda 2 parametreli tanımlanmış halini görüyoruz.

```
    public static void setOnApplyWindowInsetsListener(@NonNull final View view,
                                                       final @Nullable OnApplyWindowInsetsListener listener) {
        if (Build.VERSION.SDK_INT >= 21) {
            Api21Impl.setOnApplyWindowInsetsListener(view, listener);
        }
    }
```

Ek olarak 2. paramatredeki interface yapısına da bakalım:

```
public interface OnApplyWindowInsetsListener {
    When set on a View, this listener method will be called instead of the view's own onApplyWindowInsets
    method.
    Params: v – The view applying window insets
            insets – The insets to apply
    Returns: The insets supplied, minus any insets that were consumed
    @NonNull
    WindowInsetsCompat onApplyWindowInsets(@NonNull View v, @NonNull WindowInsetsCompat insets);
}
```

Görüldüğü gibi bu interface içinde bir fonksiyon var ve bunun 2 tane parametresi var. Birisi View tipinde diğeri insets isminde WindowInsetsCompat tipinde. Ayrıca bu fonksiyonun dönüş tipi de WindowInsetsCompat. Buradaki 2 parametreye dikkat edelim. Bu interface yapısını çağırırken Kotlin'de higher order fonksiyon yapısını kullanıp oraya parametre olarak geçiriyoruz.

Fonksiyonumuzu tekrar hatırlayalım:

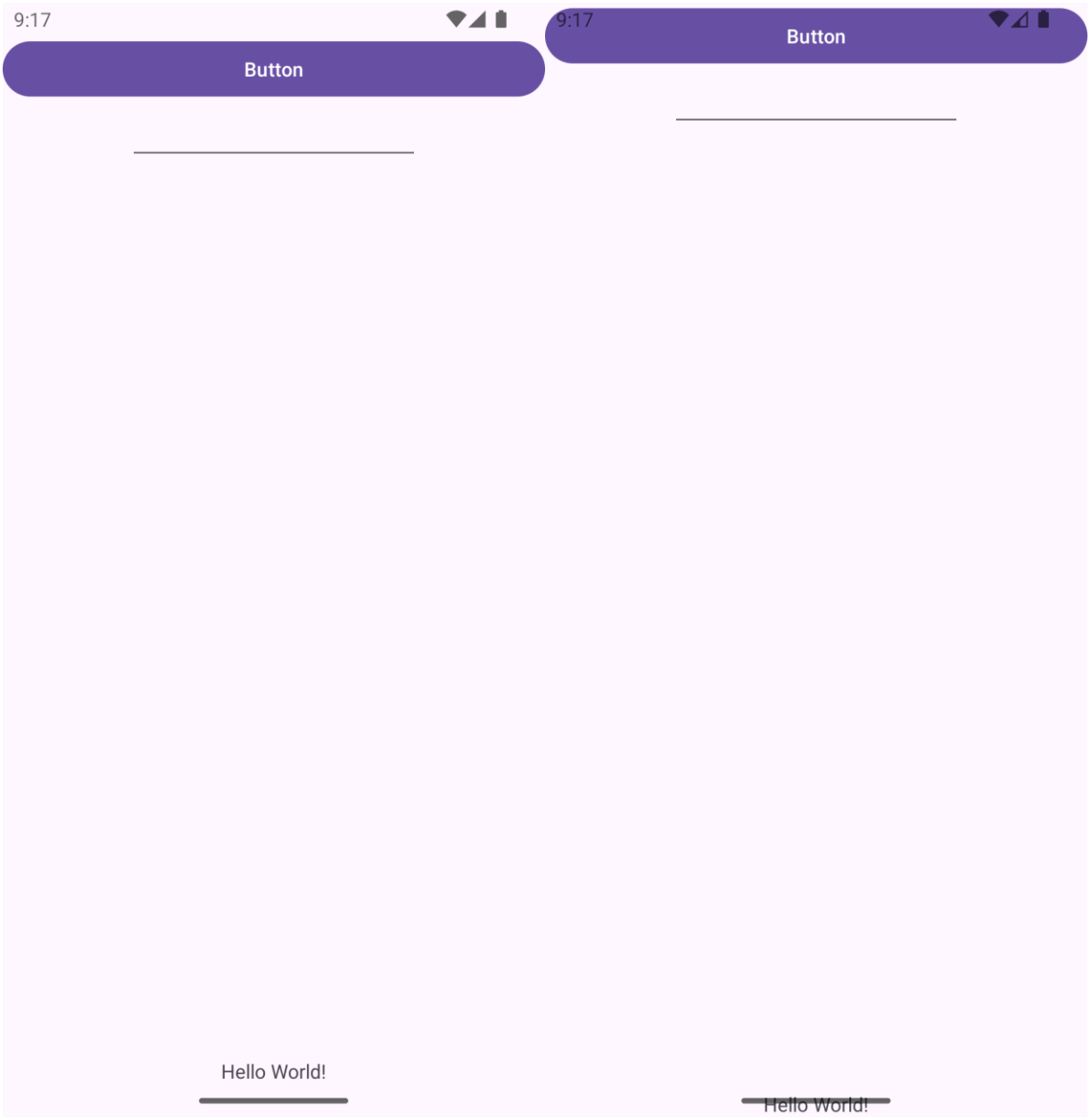
```
ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)) { v, insets ->
    val systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars())
    v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom)
    insets ^setOnApplyWindowInsetsListener
}
```

Burada ilk parametrenin view, ikincisinin de bir interface'e ait olduğunu söyledik. İkinci parametreyi bir lambda bloğu şeklinde kullanabiliriz. Fakat bu lambda bloğunda, interface içinden gelen view ve insets parametreleri de olacağından belirtmek zorundayız. Ardından bu body içinde istediğimiz işlemi gerçekleştirebiliriz.

İncelediğimiz fonksiyonun arka plandaki yapısını gördük. Şimdi ne, ne işe yarıyor ona bakalım.

View tipi için ilk verdiğimiz parametre olan findViewById(R.id.main) ile ekranımızdaki layout yapısını verdik. Buradaki layout, ekranı tamamen kaplayan bir kutu gibi düşünülebilir. Bu kutu içerisine compenentlerimizi (buton, text vs.) yerleştiriyoruz. Ikinci parametredeki v ve insets parametrelerini görüyoruz. Buradaki v, ilk verdiğimiz parametre ile eşleşiyor. Yani layoutumuzun kendisi oluyor. Insets ise pencere giren alanların boyutlarını ve konumlarını temsil eden bir `WindowInsetsCompat` nesnesidir. Bu nesne sayesinde, sistem çubukları (status bar ve navigation bar), kesiklikler (kamera çentiği gibi) gibi farklı pencere girinti türlerinin bilgilerine erişiriz. Her cihaz için ayrı boyutları verdiği için oldukça yararlı bir kullanım sağlayabiliyoruz.

systemBars isimli değişkenimizde, cihazın statüs bar ve navigasyon bar ölçülerini almış oluyoruz. Bir alttaki satırda ise view yapımız için padding veriyoruz. Padding verirken bu üstte aldığımız ölçüyü kullanıyoruz. Bu sayede constraintleri tam belli olmayan componentlerde de bir hata yaşamıyoruz. Çünkü bu yapılar ne statüs barın ne de navigasyon barının üstüne biner. Son olarak bunu da bir örnek ile gösterelim.



Sağdaki ekranımızda, butonumuzun statüs barın üstüne bindiğini ve textView'ın ise navigasyon barı üstüne bindiğini görüyoruz. Fakat soldaki ekranımızda bu problemi engelliyoruz. Nasıl engelliyoruz? SetOnApplyWindowInsetsListener fonksiyonu ile cihazın pencere girdileri için padding veriyoruz ki diğer componentler oradan başlamasın.

- ÖMER SUNGUR