SETONAPPLYWINDOWINSETSLISTENER

```
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
}
```

SetOnApplyWindowInsetsListener fonksiyonun işlevini öğrenmek için yukardaki kod bloğunu anlamak gerekiyor. Bunun için öncelikle ViewCompat nedir onu sormalıyız. Viewcompat , Android'de sürümler arası görünümlerin birbiriyle uyuşmasını sağlayan ViewCompat.java sınıfından gelen bir sınıftır. Bu sınıfta Android sürümler arası uyuşmazlığı çözen birçok fonksiyon vardır. Bu fonksiyonlardan biri de setOnApplyWindowInsetsListener fonksiyonudur.

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

Yukarıdaki ViewCompat.java sınıfında görüldüğü üzere setOnApplyWindowInsetsListener fonksiyonu zorunlu bir view ve zorunlu olmayan bir windowlistener parametresi alıyor ve eğer cihazımızın api seviyesi 21'e eşit veya üzerinde ise setOnApplyWindowInsetsListener fonksiyonunu cihazımızda bulunduğu yere uyguluyor.

```
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);
}
```

İkinci parametreyi inceleyelim. Bu onApplyWindowInsets fonksiyonu, OnApplyWindowInsetsListener interface'inde bulunan ve pencere kenarlık değerlerinin bir view'a uygulanması durumunda çağrılır. Bu yöntem, belirli bir view'a pencere boşluk değerlerinin uygulanmasını dinler ve bu değerlerin nasıl uygulanacağını belirler. Burdaki insets ise Uygulanacak pencere kenarlık değerlerini içeren WindowInsetsCompat nesnesidir. Bu nesne ana fonksiyonumuzda zaten bize dönüyor.

```
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
}
```

Peki tam olarak ne işe yarıyor? setOnApplyWindowInsetsListener fonksiyonu cihaza göre pencere boşluklarını ele almamızı sağlayan ve bunu istediğimiz herhangi bir view'a uygulamamızı sağlayan bir api'dir diyebiliriz. Android cihazlardaki üst taraftaki sistem özelliklerini bulunduran status bar veya alt taraftaki navigation bar gibi ekranın kenarlarına yerleşen sistem özelliklerinin boyutunu ve düzenini kullanmamızı sağlar. Fonksiyonun bize döndürdüğü v bizim verdiğimiz view, insets ise sistem pencere boşluk değerlerini içeren bir windowinsetscompat nesnesidir. Biz bu iki değeri kullanarak uygulamamızı istediğimiz şekilde düzenleyebilir ve çeşitlendirebiliriz.

SystemBars değişkeni ile cihazdaki sistem barlarının boyutunu ve düzenini alıyoruz. Ardından kullandığımız view üzerinden bize dönen v nesnesini kullanarak setPadding ile dört taraftan boşluklar verebiliyoruz.Default kısımda systemBars boyutuna göre view'a boşluklar veriyoruz.

Daha farklı neler yapabiliriz? Sistem çubuklarının görünürlüğünü ayarlayabilir, buna göre animasyonlar verebilir, view'ın sistem çubuğuna göre özelliklerini değiştirebilir, Bu değerlere göre View'in konumunu ve boyutunu ayarlayabiliyoruz. Bu şekilde Android bize sistem çubuklarının özelliklerine göre uygulamamızı kontrol edebilme özelliği veriyor.