**setOnApplyWindowInsetsListener**

Android cihazlarımızda saati, bildirim simgelerini, şarj durumunu vs görebildiğimiz alan status bar (durum çubuğu) ve geri dönmemizi, anasayfayı açmamızı ve açık uygulamalarımı görmemizi sağlayan navigation bar (gezinme çubuğu) vardır. Bunların ekranda kapladığı belli bir alan vardır. Bu fonksiyon ile ekranlarımızdaki bu boşlukları oluştururuz.

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  
**}**

Buradaki kodlamada systemBars nesnesi oluşturulmuş ve View sınıfının setPadding fonksiyonu ile en dıştaki Constraint Layout’a padding verilmiş. Bu padding ölçüleri ise systemBars.left / top / right / bottom ile dinamik hale getirilmiş. Burada top ile status bar, bottom ile navigation bar yükseklikleri temsil ediliyor. Sağ ve solda mevcutta sistem bar olmadığı için bir padding vermiyor. Üstten ve alttan sistem barlarının yüksekliği kadar padding veriyor. Bununla birlikte tasarımımıza eklediğimiz görünümlerin, sistem barlar ile üstüste binmesi engellenir.

Fonksiyonun kaynağına gittiğimizde;

public static void setOnApplyWindowInsetsListener(@NonNull final View v,  
 final @Nullable OnApplyWindowInsetsListener listener) {  
 if (Build.VERSION.*SDK\_INT* >= 21) {  
 Api21Impl.*setOnApplyWindowInsetsListener*(v, listener);  
 }  
}

OnApplyWindowInsetsListener interfaceini parametre olarak aldığını görüyoruz. Interface’deki açıklamada sistem barların özel olarak ayarlanması için kullanılacağını görebiliriz. Bu fonksiyonu kullandığımızda, view’ın default onApplyWindowInsets metodu yerine bizim view’a özel ayarlayacağımız onApplyWindowInsets metodunu dikkate alacaktır.

Böylecek sistem bar boyutlarını ve görünümlerini yönetebiliriz, tam ekran modunu yönetme imkanı buluruz.