**BÖLÜM 8**

# Widget'larınızı Şekillendirme

Widget'larınızı şekillendirmek sizin için tamamen yeni bir şey değil. Önceki bölümlerde bazı küçük stil özelliklerine değindik ve kod örneklerimizde stil tekniklerini gördünüz. Ancak bu bölüm, stil oluşturma konusunda derinlemesine bir dalış yapacağımız bölümdür. Nihayet! Widget'larımızın harika çalışmasının yanı harika görünmesini de sağlayabileceğiz.

Bölüm 6, "Widget'larınızı Yerleştirmek" kısmında düzenleri ele almıştık, dolayısıyla widget'larımızın konumlarını ve boyutlarını nasıl ayarlayacağımızı biliyoruz. Bu şekillendirme değildir. Stil, pencere öğelerinin görünümünü etkileyen diğer şeylerdir. Şunlar gibi şeyler:

* Renkler
* Metin görünümü (yazı tipleri, ağırlıklar, alt çizgiler, vb.)
* Kenarlıklar (kalınlıklar, desenler, köşe yarıçapları)
* Arka plan resimleri
* Konteynere şekil uygulama

Bunları, düzen ve stil arasında önemli ölçüde örtüşme olan birkaç Flutter widget'ı

ile birlikte tartışacağız: Kartlar ve Yığınlar.

Son olarak, Temaları kullanarak varsayılan stilleri toplu olarak uygulamanın en iyi

uygulamasından bahsedeceğiz.

Ama önce Flutter'daki stillerin felsefesi hakkında birkaç söz.

© Rap Pay

## FlutterTarzlarında Düşünme

Flutter'ın Android, iOS geliştirme ve web geliştirmeden, özellikle de React ve JavaScript'in dinamikliğinden en iyi fikirleri ödünç aldığını muhtemelen görmüşsünüzdür. Ancak onların tekniklerini birebir kopyalamıyor. Flutter her şeyi kendi yöntemiyle yapıyor ve web benzetmelerini çok ileri götürmek hata olur. Flutter'ın nasıl farklı olduğunu kavramak bizim yararımıza olacaktır.

Öncelikle Google, Material Design'ı[1](#_bookmark1) Android geliştirmede ve tüm Web'de çok popüler hale getirdi. Flutter'ın varsayılan görünümü Materyal Tasarım ile uyumlu olsa da, kimsenin size bunu kullanmak olduğunuzu söylemesine izin vermeyin. Bu popüler ama doğru olmayan bir yanılgıdır.

İkinci olarak, Flutter'ın stili CSS değildir. CSS, alt öğelerine aktarılan belirli özelliklere sahipken, Flutter stilleri miras alınmaz. Örneğin, özel widget'ınızda bir yazı tipi ailesi ayarlayamazsınız ve altındaki tüm Metinlerin, Metin Alanlarının ve düğmelerin aniden bu yazı tipiyle oluşturulmaya başlamasını sağlayamazsınız. Bu tür bir şey yapmak için

gerçekleşirse, bu bölümün sonunda nasıl yapılacağını göstereceğimiz Temaları

kullanmak mümkündür.

Son olarak, Flutter, Dart gibi çok ayrıntılıdır - basit olması gerektiğini düşündüğünüz bir şeyi ifade etmek için çok fazla kod gerekir. Ve şaşırtıcı olmayan bir şekilde, stil de farklı değildir. Flutter'ı çok zeki insanların yazdığından ve bu kadar ayrıntılı olmasının çok iyi nedenleri olduğundan emin olabilirsiniz; güvenlik ve bütünlük bunlardan sadece ikisi. Bu hayatın bir gerçeği. Biz sadece farkında olmanızı ve buna hazırlıklı olmanızı istiyoruz.

1 Materyal Tasarım, bir uygulamanın görünümü ve hissi için bir dizi kılavuzdur. Kısaca, kullanıcı arayüzünü fiziksel dünyaya, özellikle de üst üste dizilmiş kağıt öğelere benzetmeye çalışır. Bu konuda daha fazla bilgiyi <https://material.io/> adresinden edinebilirsiniz.

## Renkler hakkında bir kelime

Çoğu Flutter stili çok dar bir şekilde uygulanır; yalnızca sıkı bir şekilde tanımlanmış belirli durumlar için anlamlıdır. Öte yandan, renkler neredeyse her yere uygulanır (Şekil [8-1](#_bookmark2)).

Kenarlıklar, metinler, arka planlar, simgeler, düğmeler ve gölgelerin hepsinin renkleri vardır. Ve hepsi aynı şekilde belirtilir. Örneğin, burada sarı kenarlıklı kırmızı bir kapta beyaz Metin var ve bu widget'ların tümü "color: Colors. somethingOrOther" sözdizimiyle aynı şekilde renklendirilmiştir:

child: Container( child: Text( 'Colors!',

style: TextStyle(color: Colors.white,),

),

decoration: BoxDecoration( color: Colors.red, border: Border.all( color: Colors.yellow,

)),

),



***Şekil 8-1.*** *Flutter'da renkler her yerde*

Ve arka plandaki şu renkli blokları görüyor musunuz? Onlar bu şekilde oluşturuldu:

List<Widget> \_randomColors() {

Random rnd = Random(); return List.generate(25, (int i) => Container( color: Color.fromRGBO(

rnd.nextInt(255), rnd.nextInt(255), rnd.nextInt(255), 1.0),

));

}

Böylece Color. fromRGBO(red, green, blue, opacity) ile üç RGB renginin her birinin 0 ile 255 arasında bir sayı olduğu ve opaklığın tamamen opak için 1.0 ve tamamen saydam için 0.0 olduğu 16+ milyon renkten herhangi birini oluşturabilirsiniz.

Web geçmişiniz varsa, onaltılık sayıları kullanarak renk oluştururken daha rahat

olabilirsiniz. Bu da işe yarar:

color: Renk(0xFFFF7F00),

**Dikkat** Dikkatli olun. Bu onaltılık sayı aslında ilk iki onaltılık karakterin alfa kanalı olduğu "ARGB "dir. Bunu unutursanız, Color(0xFFF700) gibi, tam saydam boyamış olursunuz ve asla göremezsiniz. Renkleriniz görünmüyorsa, bu tipik web onaltılık numarasını alın ve önüne bir "FF" koyun.

## Metin Şekillendirme

Metnin görünümü ile ilgili iki konu vardır: TextStyle ve Özel Yazı Tipleri. Bunlarla bir ya da iki sayfa sonra ilgileneceğiz ancak bu özellikleri ayarlamanın daha iyi bir yolunu ele alacağımız bölümün sonuna kadar izlemeye devam edin - Temalar.

### Metin Stili

Metin widget'ları bir TextStyle nesnesi alan bir *stil* özelliğine sahiptir (Şekil [8-2](#_bookmark3)).

 ***Şekil 8-2.*** *Stil ile ve stil olmadan*

Basitçe *style* özelliğini bir TextStyle widget örneğine ayarlayacak ve özellikleri

ayarlayacaksınız. TextStyle yaklaşık 20 özelliği destekler. İşte en kullanışlı olanları:

* color - Geçerli 16+ milyon renkten herhangi biri
* decoration - TextDecoration.underline, overline, strikethrough, none
* fontSize - Bir çift. Karakterlerin kaç piksel uzunluğunda belirler.

Varsayılan boyut 14.0 piksel

* fontStyle - FontStyle.italic veya normal
* fontWeight - FontWeight.w100-w900. Veya kalın (w700 olan) veya normal (w400 olan)
* fontFamily - Bir dize

fontFamily daha büyük bir konudur. Courier, Times New Roman, serif ve daha fazlası gibi yerleşik bazı yazı tipleri vardır. Kaç tane daha var? Bu, uygulamanın üzerinde çalıştığı cihazın türüne bağlıdır. Kullanıcıların cihazları üzerinde kontrolümüz olmadığından, en iyi uygulama, özel bir yazı tipi yükleyip kullanmadığınız sürece varsayılan yazı tipi ailesine bağlı kalmanızdır. Bunun nasıl yapılacağı hakkında konuşalım.

### Özel yazı tipleri

Bazı tasarımcılar sahne tasarlarken özel yazı tiplerine ihtiyaç duyarlar. Flutter ile özel

fontları kolay olduğu ve platformlar arası çalıştıkları ortaya çıktı. Üç adımdan oluşuyor:

1. ttf, woff veya woff2 biçimindeki özel yazı tipi dosyalarını indirin. Bunlar geleneksel olarak fonts adlı kök düzey bir klasörde saklanır, ancak adı size bağlıdır (Şekil [8-3](#_bookmark4)).

metin, ekran görüntüsü, yazı tipi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

#### **Şekil 8-3.** Yazı tipleri genellikle fonts adlı bir klasörde saklanır

İ**pucu**Bazı mükemmel ve ücretsiz yazı tiplerini [http://fonts. google.com](http://fonts.google.com/) adresinde bulabilirsiniz. Türlerine göre arama yapabilir, örnekleri görebilir ve kolayca indirebilirsiniz.

1. Font dosyalarını flutter/fonts altındaki pubspec.yaml dosyasına ekleyin, böylece derleyiciye bunları kurulum dosyasında paketlemesi için bildirim gönderilir.

flutter:

fontlar

:

* aile: MrDafoe yazı tipleri:

- varlık: fonts/MrDafoe-Regular.ttf

* aile: NanumBrushScript yazı tipleri:

- varlık: fonts/NanumBrushScript-Regular.ttf

1. Önceki bölümde gibi TextStyle widget'ının fontFamily özelliğinde büyük/küçük harfe duyarlı olmayan font adını kullanın:

Text(loremIpsums[0]), // Biçimlendirilmemiş Text(loremIpsums[1], // Courier gibi bazıları

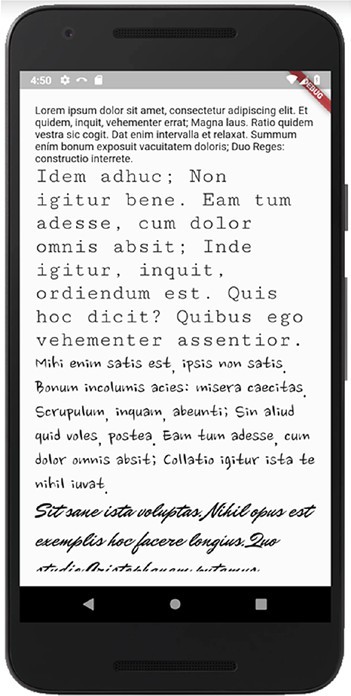
yerleşik

style: TextStyle(fontFamily: 'Courier'),), Text(loremIpsums[2],

style: TextStyle(fontFamily: 'NanumBrushScript'),), Text(loremIpsums[3],

style: TextStyle(fontFamily: 'MrDafoe'),),

Yukarıdaki örnek Şekil [8-4](#_bookmark5) gibi görünebilir.



***Şekil 8-4.*** *Kullanılabilir yazı tipleri*

## Konteyner Dekorasyonları

Metne nasıl kenarlık eklersiniz? Ekleyemezsiniz. Peki ya bir Simgeye arka plan? Hayır. Bu süslemelere sahip olma kapasiteleri yoktur. Ama ne var biliyor musunuz? Bir Konteyner. Bu gibi stil sorunlarınız olduğunda, cevap neredeyse her zaman widget'ları bir Konteynere sarmak ve Konteynere bir dekorasyon koymaktır.

Kapsayıcıların *dekorasyon* adı verilen her şeyi kapsayan bir stil özelliği vardır. İşte bir konteynere nasıl gölge koyulacağına dair bir örnek:

child: Container( width: 300.0, height: 300.0, decoration: BoxDecoration( color: Colors.purple, boxShadow: [ BoxShadow(

offset: Offset.fromDirection(0.25\*pi, 10.0), blurRadius: 10.0,

)

],

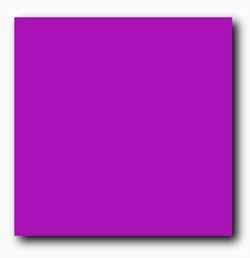
),

),

Şekil [8-5](#_bookmark6) ve [8-6](#_bookmark7) gölgesiz ve gölgeli kutuları göstermektedir.



***Şekil 8-5.*** *Kutu gölgesi olmadan*

**

***Şekil 8-6.*** *Bir kutu gölgesi ile*

Bu da Flutter'ın kelime dağarcığına müthiş bir örnek. Web'de bu 17 karakterle yapılabilirdi. Ancak Flutter'da boxShadow'un, her biri radyan cinsinden ifade edilen bir yön, piksel cinsinden ifade edilen bir boyut ve bulanıklık yarıçapını da piksel cinsinden alan bir ofsete sahip bir BoxShadows dizisi olduğunu hatırlamamız gerekir. Sheesh!

Bulanıklık yarıçapı ek açıklama gerektirebilir. Bulanıklık yarıçapı, gölgenin dağıldığı

mesafedir. Bir lambanın üzerine abajur koymak gibidir. Abajur olmadan ışık sert ve gölgeler nettir. Abajurla ışık daha yumuşak ve gölgeler de daha net olur. Bulanıklık yarıçapı ne kadar büyükse gölge o kadar yumuşak olur.

**Dikkat** Aynı zamanda bir BoxDecoration kullanıyorsanız, bir Container üzerinde doğrudan bir renk belirtemezsiniz. Ancak panik yapmayın; BoxDecoration'ın da bir renk özelliği vardır. Aynı etkiyi elde etmek için Container'ınızın renk özelliğini BoxDecoration'a taşımanız yeterlidir.

Kullanılabilecek başka süslemeler de vardır. Şimdi en kullanışlı olanlara, border, borderRadius ve BoxShape'e bakalım.

### Sınır

Tıpkı gölgeler için BoxDecoration kullandığınız gibi, bunları bir kaba kenarlık koymak için de kullanabilirsiniz. İşte dört farklı genişliğe sahip kırmızı bir kenarlık (Şekil [8-7)](#_bookmark8):

dekorasyon

decoration: BoxDecoration( color: Colors.purple, border: Border( top: BorderSide( width: 10, color: Colors.red,

),

right: BorderSide( width: 20, color: Colors.red,

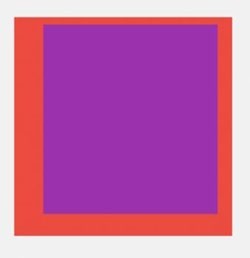
),

bottom: BorderSide( width: 30, color: Colors.red,

),

left: BorderSide( width: 40, color: Colors.red,

),



***Şekil 8-7.*** *Farklı genişliklere sahip kenarlıklar*

Flutter'ın farklı genişliklere ve hatta farklı renklerde kenarlıklara sahip olmamıza izin vermesi güzel olsa da, bunu ne sıklıkla kullanacaksınız? Genellikle dört taraf da tek tip olacaktır. Bu yüzden genellikle Border.all() kısaltmasını kullanırız:

decoration: BoxDecoration( color: Colors.purple, border: Border.all( width: 10, color: Colors.red,

),

),

Çok daha basit. Evet, hala ayrıntılı ama daha basit.

### Kenar Yarıçapı

Yuvarlatılmış köşeler favori bir görünümdür. Kenarlığı olmasa bile bir Konteyneri yuvarlak hale getirebilirsiniz (Şekil [8-8](#_bookmark9)). Bunu BorderRadius ile yaparsınız:

decoration: BoxDecoration( color: Colors.purple, borderRadius: BorderRadius.only( topLeft: Radius.circular(20.0), topRight: Radius.circular(60.0),

),

),



***Şekil 8-8.*** *İki köşede BorderRadius*

Sadece bir topLeft ve topRight yarıçapı verdik, ancak bir de bottomLeft ve bottomRight özelliği var. Esnekliği takdir etmemize rağmen, bunu kullanmak tipik değildir. Normalde dördünü de aynı şekilde belirtiriz (Şekil [8-9](#_bookmark10)):

decoration: BoxDecoration( color: Colors.purple, borderRadius: BorderRadius.all(

Radius.circular(20.0),

),

),



***Şekil 8-9.*** *Dört köşenin tamamında BorderRadius*

### Kutu Şekli

Kaplarınız her zaman dikdörtgen olmak zorunda değildir. Başka bir şekil olması gerektiğinde, BoxShape veya CustomPainter ile bunu yapabilirsiniz.

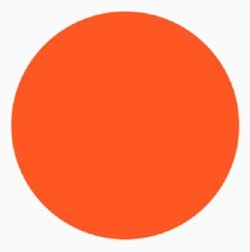
BoxShape'in kullanımı çok daha kolaydır, ancak Şekil [8-10](#_bookmark11)'da olduğu gibi yalnızca daireleri destekler (tabii ki varsayılan dikdörtgene ek olarak):

Container(

decoration: BoxDecoration( shape: BoxShape.circle, color: Colors.deepOrange,

),

),



***Şekil 8-10.*** *BoxShape.circle Konteynerinizi yuvarlak yapar*

CustomPainter çok daha karmaşıktır, ancak sonsuz şekillere izin verir.

CustomPainter'ın (Şekil [8-11](#_bookmark12)) ayrıntılarına çok fazla girmek dikkat dağıtıcı olacaktır,

ancak işte hızlı bir örnek, bir Superman kalkanı:

Container( child: CustomPaint( size: Size(200, 200),

painter: SupermanShieldPainter(),

),

)

class SupermanShieldPainter extends CustomPainter {

@override

void paint(Canvas canvas, Size size) { canvas.drawPath(Path()

..moveTo(25, 0)

..lineTo(125, 0)

..lineTo(150,25)

..lineTo(75, 125)

..lineTo(0,25)

..lineTo(25,0),

Paint()

..style=PaintingStyle.fill

..color = Colors.red

);

}



***Şekil 8-11.*** *Bir CustomPainter Kullanma*

Gördün mü? Biraz daha karmaşık. Kapsayıcınızın hala bir dikdörtgen olduğuna dikkat edin. Sadece arka plan farklıdır. CustomPainter'ı daha derinlemesine incelemek için [https://api.flutter.dev/flutter/ widgets/CustomPaint-](https://api.flutter.dev/flutter/widgets/CustomPaint-class.html)

[class.html](https://api.flutter.dev/flutter/widgets/CustomPaint-class.html) adresine bir göz atın.

**İpucu** Tüm bu süslemeler decoration özelliğine uygulanır, ancak adından da anlaşılacağı gibi, konteynerin üzerindeki bir katmana uygulanan foregroundDecoration adlı bir özelliğe de uygulanır. Aynı süslemeler oraya da uygulanır. Ancak diğer şeylerin çizildikleri için, bir değişikliği daha aklınızda tutmak isteyeceksiniz: opaklık. Renkler yarı saydam hale getirilebilir. Aşağıda,

%50 saydam olan bir kabın üstünde kırmızı bir katman oluşturulacaktır:

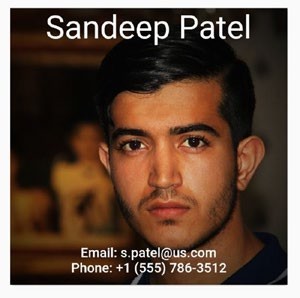
foregroundDecoration: BoxDecoration( color:Colors.red.withOpacity(0.5),

),

## Widget'ları istifleme

Ekranda iki veya daha fazla şeyin aynı x ve y konumunu işgal etmesini istiyorsanız, Stack widget'ına ulaşın. Yığın widget'ı bir widget'ı yerleştirmemizi ve ardından onun önüne başka bir widget'ı ve onun önüne başka bir widget'ı koymamızı sağlar. Açıkçası daha sonra eklenenler daha yüksek bir sahip olacak ve böylece arkasındakini "gizleyecektir" (gizleyecektir). Temel olarak, iç widget'ları ... iyi ... istiflemenizi sağlar.

Bir Yığın kullanarak gerçekten harika düzenler oluşturabilirsiniz. Aslında, Materyal Kartları, metin bulunan arka plan görüntülerini kucakladıkları için Yığınlara çok güvenir. Belki de bir kişinin profil fotoğrafı ile adının ve bilgilerinin üst üste bindirildiği bir kart istiyoruzdur (Şekil [8-12](#_bookmark13)).



#### **Şekil 8-12.** Yığın widget'ı sayesinde bir resmin üstünde metin bulunan bir Kart

*Şekil* [*8-12*](#_bookmark13) *ile* [*8-14*](#_bookmark16) *arasındaki görseller Hosein Hakimi'nin* [*Unsplash.com adresindeki*](http://unsplash.com/) *izniyle*

İşte bunu nasıl başarabileceğimiz:,

Card( child: Stack( children: <Widget>[

Image.asset("6.jpg"),

Column(

children: <Widget>[

Text(

"Sandeep Patel",

style: Theme.of(context).textTheme.display1

.copyWith(color: Colors.white),

),

Expanded(child: Container()), Text("Email: s.patel@us.com",

style: Theme.of(context)].textTheme.body2

.copyWith(color: Colors.white)),

Text("Phone: +1 (555) 786-3512",

style: Theme.of(context).textTheme.body2

.copyWith(color: Colors.white)),

],

),

],

),

),

Yığına önce bir resim yerleştirdik. Daha sonra bunun üzerine metin öğeleri içeren bir Sütun ekledik. Sütun görselden sonra eklendiği için görselin önünde görünüyor.

### Konumlandırılmış widget

Önceki örneğimizde, bir Sütun alt öğelerini ortaladığı ve Genişletilmiş de Metinleri en üste ve en alta ittiği için metinler düzgün bir şekilde yerleştirilmiştir. Ancak her şeyi doğrudan Yığın'a yerleştirseydik, Şekil [8-13](#_bookmark14)'teki gibi görünürdü.



#### **Şekil 8-13.** Konumlandırılmış bir widget olmadan her şey sol üstte toplanır

Bir Stack kullandığınızda, içindeki her widget sol üst köşede kalmaya çalışacaktır. Bu iç widget'ları bir Positioned widget'ına sararak bir Stack içinde istediğimiz yere yerleştirebiliriz.[2](#_bookmark15)

Card( child: Stack( children: <Widget>[

Image.asset("6.jpg"),

Positioned( top: 10, left: 10, child: Text("Sandeep Patel",

2 Yığın içinde konumlandırmak için Container, Align ve Padding gibi başka teknikler de vardır.

Ancak Position, Stack ile harika çalışır.

style: Theme.of(context).textTheme.display1

.copyWith(color: Colors.white),

),

),

Positioned( bottom: 30, right: 10, child: Text("Email: s.patel@us.com", style: Theme.of(context).textTheme.body2

.copyWith(color: Colors.white)),

),

Positioned( bottom: 10, right: 10, child: Text("Phone: +1 (555) 786-3512", style: Theme.of(context).textTheme.body2

.copyWith(color: Colors.white)),

),

Positioned(

bottom: 0, left: 0, height: 100, width: 100, child: FlutterLogo(),

)

],

),

)

İyi bir önlem için bir FlutterLogo ekledik. Şimdi Şekil [8-14](#_bookmark16)'e benziyor. Çok daha güzel!

metin, insan yüzü, adam, insan, ekran görüntüsü içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

#### **Şekil 8-14.** Konumlandırılmış widget ile çok daha güzel görünüm

Positioned widget'ı, üst, alt, sol ve/veya sağ konumları belirterek alt öğesini dört köşeden birinden sabit bir uzaklığa getirir.

## Kart widget'ı

Önceki örneğimizde bir Kart widget'ı kullandığımızı fark etmiş olabilirsiniz. Bir Kart bu durumda yapılması gereken doğru şey görünüyor, ancak hiçbir şekilde gerekli değildir.

Material Design görünüm ve hissini uygulamak için arka plan rengi için color, alt gölge boyutu için elevation, kenarlık için borderOnForeground ve etrafındaki boşluk için margin gibi özelliklere sahip bir Flutter Card widget'ı oluşturuldu. Kabul etmek gerekir ki, tüm bunlar bir Konteyner ile de gerçekleştirilebilir. Ancak bunu standart bir görünüm ve hisle yapmak istiyorsanız, bir Kart bunu kolaylaştırır:

Card( elevation: 20.0, child: Text("This is text in a card", style: Theme.of(context).textTheme.display3),

),

## Temalar

Sizi bilmem ama ben bir uygulamanın önceden iyi planlanmış, düşünülmüş ve tasarlanmış olmasını seviyorum. Aksi takdirde, italik, kalın ve altı çizili yazıların öngörülemeyen, dengesiz bir şekilde kullanıldığı çılgın bir renk ve yazı tipi yorganıyla karşılaşabilirsiniz. Kısacası, [http:// lingscars.com](http://lingscars.com/) adresini yeniden oluşturmak istemezsiniz.

Tutarlı bir stil kullanımı, kaliteyi yansıtan hoş bir uygulama yaratır.

Tutarlı kalmanın harika bir yolu da bir Temaya bağlı kalmaktır.

#### Flutter'daki bir Tema, birlikte uygulanabilen mantıksal olarak tanımlanmış

*gruplardaki stillerin bir gruplamasıdır.*

Bu şekilde, uygulamanızın her yerinde tutarlı bir görünüm ve his elde etmekle kalmaz, aynı zamanda temayı tek bir (1!) yerden, MaterialApp'ten kolayca değiştirebilirsiniz ve bu tüm çocuklara yayılır:

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) { return MaterialApp( title: 'Ch 8 - Styling Text', theme: ThemeData(primarySwatch: Colors.yellow), home: HomeWidget(),

);

}

}

Temalarla başka bir şey yapmıyorsanız, primarySwatch rengini ayarlamak isteyeceksiniz. Bunu yaparken aslında diğer tüm renkleri de ayarlamış olursunuz. PrimarySwatch ayarlandığında, bunların hepsi de otomatik olarak ayarlanır:

|  |  |  |
| --- | --- | --- |
| [accentColor](https://docs.flutter.io/flutter/material/ThemeData/accentColor.html) | [backgroundColor](https://docs.flutter.io/flutter/material/ThemeData/backgroundColor.html) | [bottomAppBarColor](https://docs.flutter.io/flutter/material/ThemeData/bottomAppBarColor.html) |
| [buttonColor](https://docs.flutter.io/flutter/material/ThemeData/buttonColor.html) | [canvasColor](https://docs.flutter.io/flutter/material/ThemeData/canvasColor.html) | [cardColor](https://docs.flutter.io/flutter/material/ThemeData/cardColor.html) |
| [colorScheme](https://docs.flutter.io/flutter/material/ThemeData/colorScheme.html) | [cursorColor](https://docs.flutter.io/flutter/material/ThemeData/cursorColor.html) | [dialogBackgroundColor](https://docs.flutter.io/flutter/material/ThemeData/dialogBackgroundColor.html) |
| [disabledColor](https://docs.flutter.io/flutter/material/ThemeData/disabledColor.html) | [dividerColor](https://docs.flutter.io/flutter/material/ThemeData/dividerColor.html) | [errorColor](https://docs.flutter.io/flutter/material/ThemeData/errorColor.html) |
| [highlightColor](https://docs.flutter.io/flutter/material/ThemeData/highlightColor.html) | [hintColor](https://docs.flutter.io/flutter/material/ThemeData/hintColor.html) | [indicatorColor](https://docs.flutter.io/flutter/material/ThemeData/indicatorColor.html) |
| [primaryColor](https://docs.flutter.io/flutter/material/ThemeData/primaryColor.html) | [primaryColorDark](https://docs.flutter.io/flutter/material/ThemeData/primaryColorDark.html) | [primaryColorLight](https://docs.flutter.io/flutter/material/ThemeData/primaryColorLight.html) |
| [scaffoldBackgroundColor](https://docs.flutter.io/flutter/material/ThemeData/scaffoldBackgroundColor.html) | [secondaryHeaderColor](https://docs.flutter.io/flutter/material/ThemeData/secondaryHeaderColor.html) | [selectedRowColor](https://docs.flutter.io/flutter/material/ThemeData/selectedRowColor.html) |
| [splashColor](https://docs.flutter.io/flutter/material/ThemeData/splashColor.html) | [textSelectionColor](https://docs.flutter.io/flutter/material/ThemeData/textSelectionColor.html) | [textSelectionHandleColor](https://docs.flutter.io/flutter/material/ThemeData/textSelectionHandleColor.html) |
| [toggleableActiveColor](https://docs.flutter.io/flutter/material/ThemeData/toggleableActiveColor.html) | [unselectedWidgetColor](https://docs.flutter.io/flutter/material/ThemeData/unselectedWidgetColor.html) |  |

Bu son derece verimlidir. Tek bir kod satırı tüm bu diğer şeyleri uygun değerlere ayarlar. Elbette, bu renklerden birinin değişmesini gerektiren özel gereksinimleriniz varsa (örneğin, kullanıcılarınız kırmızı renkten korkuyorsa), o zaman ihtiyacınız olan her şeyi değiştirin. Ancak genellikle varsayılanları kullanmak isteyeceksiniz çünkü her renk, primaryColor ve diğerlerinin yanında bir set olarak iyi görünecek şekilde tasarlanmıştır.

Temanızda renk dışında, kaydırıcıdaki onay işaretlerinin boyutu, sahneden sahneye geçne tür animasyonların gerçekleşeceği, modal iletişim kutularının keskin mi yoksa yuvarlak köşeli mi olacağı gibi pek çok ayar vardır. Tıpkı renklerde olduğu gibi, uygulamanız gerektiriyorsa bunları değiştirmekten çekinmeyin, ancak bu küçük ayrıntılarla uğraşmak zorunda kalmamamız ve harika bir uygulama oluşturmaya odaklanabilmemiz harika.

Temanızdaki tüm mülkleri buradan keşfedin[: https://docs.](https://docs.flutter.io/flutter/material/ThemeData-class.html) [flutter.io/flutter/material/ThemeData-class.html](https://docs.flutter.io/flutter/material/ThemeData-class.html). Bu biraz zaman alacaktır. Orada bir sürü seçenek var. Hepsini yönetmek zorunda kalmak yerine, primarySwatch'ı ayarlayıp işimizi halledebileceğimiz için minnettarım!

Bu listede bir textTheme, bir appBarTheme, bir InputDecorationTheme, bir sliderTheme ve çok daha fazlasının olduğunu fark edeceksiniz. Hadi konuşalım

bir saniyeliğine bu gruplamalar hakkında konuşalım. Uygulamanızdaki birçok şey *türü*, o

tür için stil özelliklerini toplayan varsayılan bir temaya sahiptir. Bu türde bir widget eklediğinizde, varsayılan olarak tema özelliklerini otomatik olarak alır.

### Tema özelliklerini uygulama

Unutmayın, temaları uygulamanızdaki hemen hemen her widget'ta kullanmak için hiçbir şey yapmanız gerekmez. Hiçbir şey. Aslında, bir Tema ve bazı temel özellikleri ayarlamanın tüm fikri budur; uygulamanız sadece işlendiklerinde onları emer. Tema onların varsayılan görünümü ve hissi haline gelir.

Peki ya açıkça bir tema uygulamak istiyorsanız? Örneğin, bir sahnenin en üstünde bir Metin widget'ınız var ve bunun bir sayfa başlığı olarak işlev görmesini istiyorsunuz. Ya da belki bunun altında bir yerde ikinci bir

seviye başlığı. Belki bir yerde bir alt başlık. Bu özel Metin widget'larına özel bir şekilde çizilmeleri gerektiğini nasıl söylersiniz? Stil özelliğini hatırlıyor musunuz?

Metin widget'larının TextStyle nesnesi alan bir stil özelliği vardır. Ancak bilinen metin stillerine temadan şu şekilde erişebilirsiniz:

Text('title', style: Theme.of(context).textTheme.title),

Text('subtitle', style: Theme.of(context).textTheme.subtitle),

Text('headline', style: Theme.of(context).textTheme.headline),

Text('subhead', style: Theme.of(context).textTheme.subhead),

Aralarından seçim yapabileceğiniz Tablo [8-1](#_bookmark17)'deki Malzeme metin Temalarına

sahipsiniz.

***Tablo 8-1.*** *Malzeme teması metin stilleri*

**Metin teması adı**

**Açıklama**

Body1 Gördüğünüz metnin çoğu. Açıkça bir stil uygulamazsanız bu varsayılan stil

olacaktır

Body2 Biraz daha kalın gövde metni

Buttone Düğmelerdeki varsayılan yazı tipi, genellikle büyük harflerle ve biraz yayılmış Caption Fotoğraf başlıkları için

Display1 En küçük başlık (diğer adıyla başlık4)

display2 3. en büyük manşet (diğer adıyla manşet3)

display3 2. en büyük manşeti (diğer adıyla manşet2)

display4 En büyük manşet (diğer adıyla manşet1)

Headline Alt başlıklar için. Genellikle bir başlığın hemen altında.

Subhead (diğer adıyla manşet6)

Tittle Alt başlıklar için. Genellikle bir başlık üst çizgisinin hemen overliine Nadiren kullanılır. Bir başlığı tanıtmak için

Şekil [8-15](#_bookmark18) bunların neye benzediğini göstermektedir.



***Şekil 8-15.*** *Materyal metin stillerinin bir cihazda nasıl göründüğü*

**İpucu** Bir temadan bir stil istediğinizde ancak birkaç özelliği değiştirmek istediğinizde copyWith() işlevini kullanın. İşte bir örnek:

Text('Foo', style:

Theme.of(context).textTheme.body2.copyWith(color:Colors.red),

),

Mevcut bir stile ulaşıyor ve onun bir kopyasını oluşturuyorsunuz, ancak bu örnek için bir veya iki özelliği değiştiriyorsunuz. Bu, orijinali değiştirmeden bırakırken biraz özelleştirilmiş stilinizi kullanmanıza olanak tanır.

## Sonuç

Böylece Flutter'da bir şeyleri şekillendirmek için seçeneklerin neredeyse sonsuz olduğunu görebilirsiniz. Flutter stilleri CSS'de gördüklerinize benzer, ancak hiçbir şekilde aynı değildir. İlk olarak, daha ayrıntılıdır. İkincisi, miras almaz.

Bazı insanlar bu özelliklerden rahatsız olabilir, ancak diğerleri yarattığı temizlikten hoşlanacaktır.

Bu konuda ne düşündüğünüzden bağımsız olarak, Flutter'ın sunduğu stil seçeneklerinden etkilenmeniz gerekir, özellikle de bunların Temalar'da nasıl düzenlendiğini düşündüğünüzde, uygulamamızın her yerinde tutarlı, profesyonel bir görünüm ve his sunabiliriz.

Şimdi, hepinizin beklediği an geldi... Durum bilgisine sahip widget'ları nasıl öğrenelim!

187