Flutter（十八）主题风格

AlanGe

在Flutter开发中，我们可以通过定义 `Theme`，复用颜色和字体样式，从而让整个app的设计看起来更一致。

一. Theme主题的使用

Theme分为：全局Theme和局部Theme

主题有两个作用：

- 设置了主题之后，某些Widget会自动使用主题的样式（比如AppBar的颜色）
- 将某些样式放到主题中统一管理，在应用程序的其它地方直接引用

1.1. 全局Theme

全局Theme会影响整个app的颜色和字体样式。

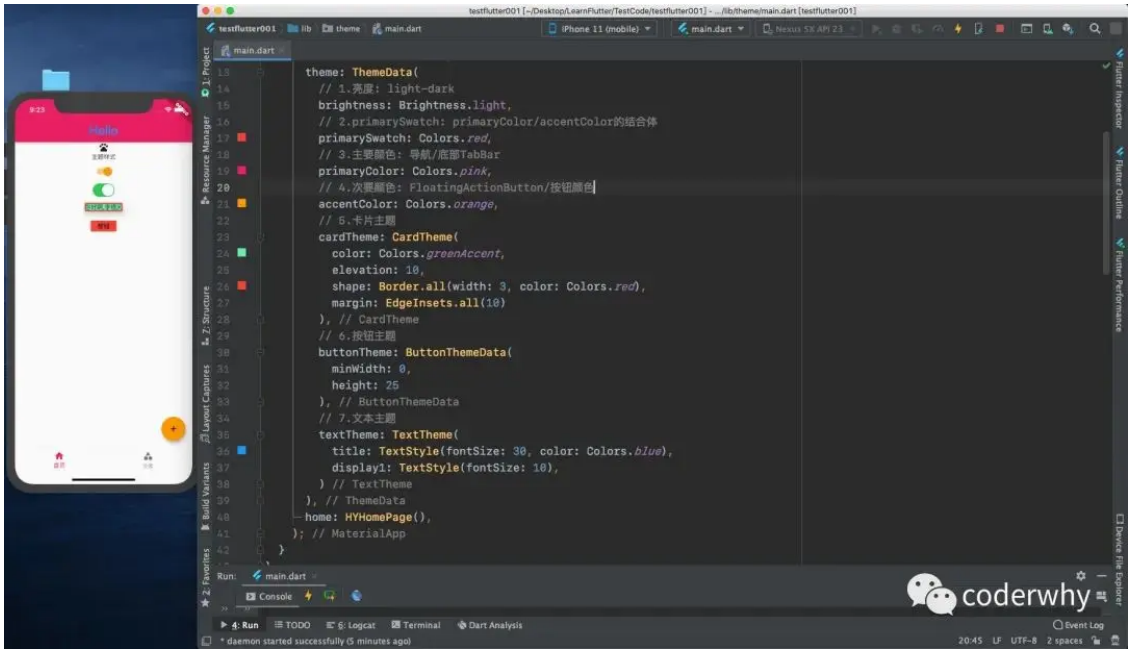
使用起来非常简单，只需要向MaterialApp构造器传入 `ThemeData` 即可。

- 如果没有设置Theme，Flutter将会使用预设的样式。
- 当然，我们可以对它进行定制。

全局主题

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // 1.亮度: light-dark
        brightness: Brightness.light,
        // 2.primarySwatch: primaryColor/accentColor的结合体
        primarySwatch: Colors.red,
        // 3.主要颜色: 导航/底部TabBar
        primaryColor: Colors.pink,
        // 4.次要颜色: FloatingActionButton/按钮颜色
        accentColor: Colors.orange,
        // 5.卡片主题
        cardTheme: CardTheme(
          color: Colors.greenAccent,
          elevation: 10,
          shape: Border.all(width: 3, color: Colors.red),
          margin: EdgeInsets.all(10)
        ),
        // 6.按钮主题
        buttonTheme: ButtonThemeData(
          minWidth: 0,
          height: 25
        ),
        // 7.文本主题
        textTheme: TextTheme(
          title: TextStyle(fontSize: 30, color:
Colors.blue),
          display1: TextStyle(fontSize: 10),
        )
      ),
      home: HYHomePage(),
```

```
    );
  }
}
```



图片

1.2. 局部 Theme

如果某个具体的 Widget 不希望直接使用全局的 Theme，而希望自己来定义，应该如何做呢?

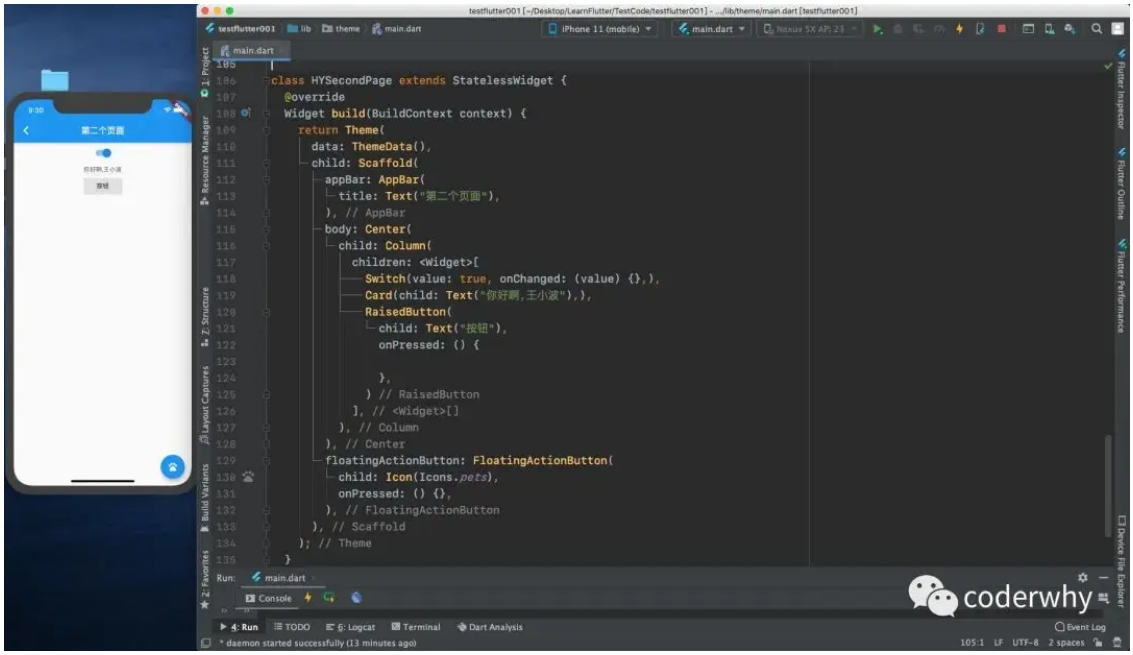
- 非常简单，只需要在 Widget 的父节点包裹一下 Theme 即可

创建另外一个新的页面，页面中使用新的主题：

- 在新的页面的 Scaffold 外，包裹了一个 Theme，并且设置 data 为一个新的 ThemeData

局部主题

```
class HYSecondPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Theme(
      data: ThemeData(),
      child: Scaffold(
      ),
    );
  }
}
```



图片

但是，我们很多时候并不是想完全使用一个新的主题，而且在之前的主题基础之上进行修改：

```
class HYSecondPage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Theme(
```

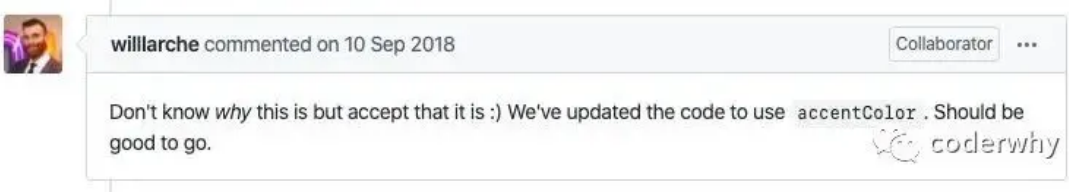
```
      data: Theme.of(context).copyWith(
        primaryColor: Colors.greenAccent
      ),
      child: Scaffold(
      ),
    );
  }
}
```

1.3. Flutter中文网错误

但是这里有一个注意事项：accentColor在这里并不会被覆盖。

为什么不能覆盖呢？https://github.com/material-components/material-components-flutter-codelabs/issues/106

我摘抄一点官方人员的回复：



图片

其实官网文档中之前也出现了类似的错误，比如Flutter中文网之前是翻译官方文档的

- https://flutterchina.club/cookbook/design/themes/其

中就有该错误

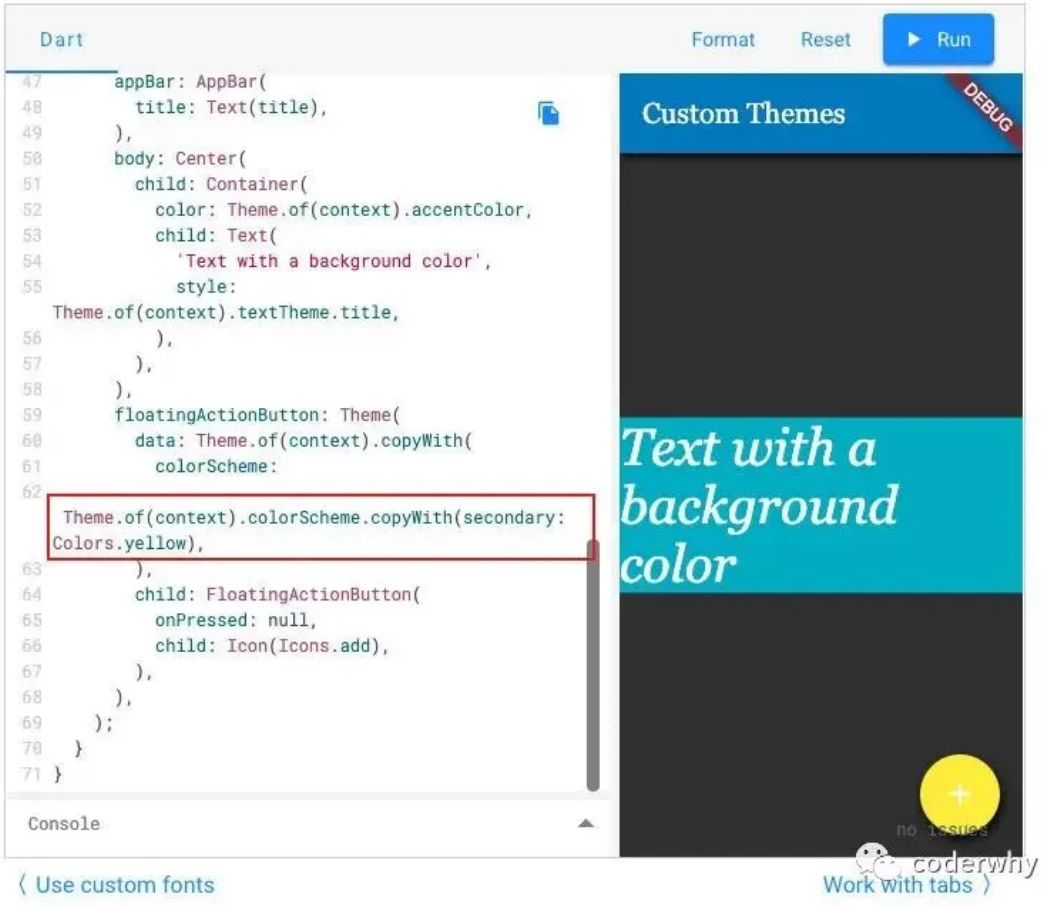
```
class MyHomePage extends StatelessWidget {
  final String title;

  MyHomePage({Key key, @required this.title}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text(title),
      ),
      body: new Center(
        child: new Container(
          color: Theme.of(context).accentColor,
          child: new Text(
            'Text with a background color',
            style: Theme.of(context).textTheme.title,
          ),
        ),
      ),
      floatingActionButton: new Theme(
        data: Theme.of(context).copyWith(accentColor: Colors.yellow),
        child: new FloatingActionButton(
          onPressed: null,
          child: new Icon(Icons.add),
        ),
      ),
    );
  }
}
```


图片

后来官网文档中对这个问题进行了修正：



图片

二. 暗黑 Theme 适配

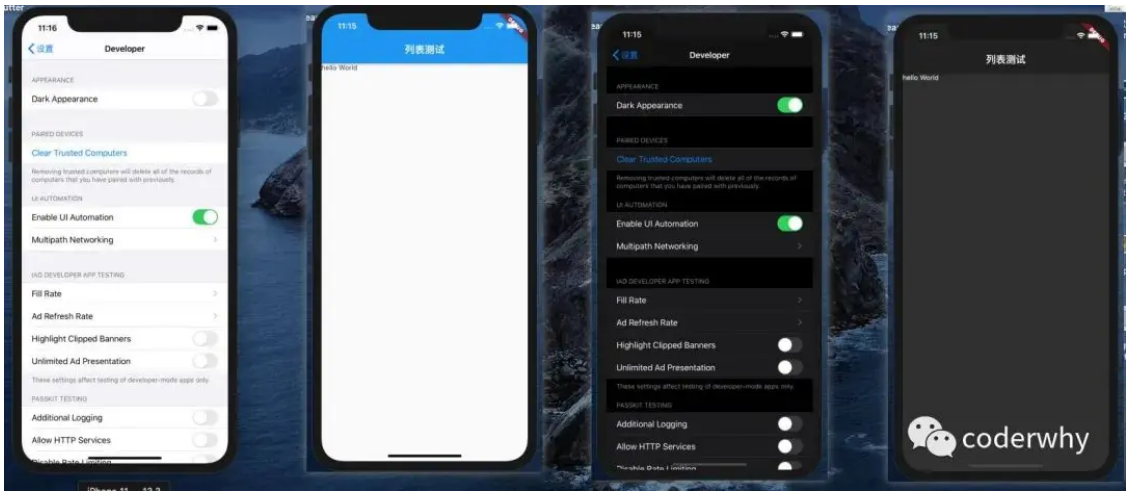
2.1. darkTheme

目前很多应用程序都需要适配暗黑模式，Flutter 中如何做到暗黑模式的适配呢?

事实上，MaterialApp 中有 theme 和 dartTheme 两个参数：

- 按照下面的写法，我们已经默认适配了暗黑主题

```dart
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      home: HYHomePage(),
    );
  }
}
```



图片

2.2. 开发中适配

在开发中，为了能适配两种主题（设置是更多的主题），我们可以封装一个 AppTheme

- 1.公共的样式抽取成常量
- 2.封装一个亮色主题

- 3.封装一个暗黑主题

```dart
import 'package:flutter/material.dart';

class AppTheme {
  // 1.抽取相同的样式
  static const double _titleFontSize = 20;

  // 2.亮色主题
  static final ThemeData lightTheme = ThemeData(
    primarySwatch: Colors.pink,
    primaryTextTheme: TextTheme(
      title: TextStyle(
        color: Colors.yellow,
        fontSize: _titleFontSize
      )
    ),
    textTheme: TextTheme(
      body1: TextStyle(color: Colors.red)
    )
  );

  // 3.暗黑主题
  static final ThemeData darkTheme = ThemeData(
    primaryColor: Colors.grey,
    primaryTextTheme: TextTheme(
      title: TextStyle(
        color: Colors.white,
        fontSize: _titleFontSize
      )
    ),
    textTheme: TextTheme(
      title: TextStyle(color: Colors.white),
      body1: TextStyle(color: Colors.white70)
```

pink 粉红色的，swatch样品，样本

```
    )
  );
}
```

在MaterialApp中，可以决定使用哪一个主题：

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: AppTheme.lightTheme,
      darkTheme: AppTheme.darkTheme,
      home: HYHomePage(),
    );
  }
}
```

参考：小码哥 Flutter