简介

本章将通过一些简单的示例来一步步介绍Flutter的开发流程.

本章目录

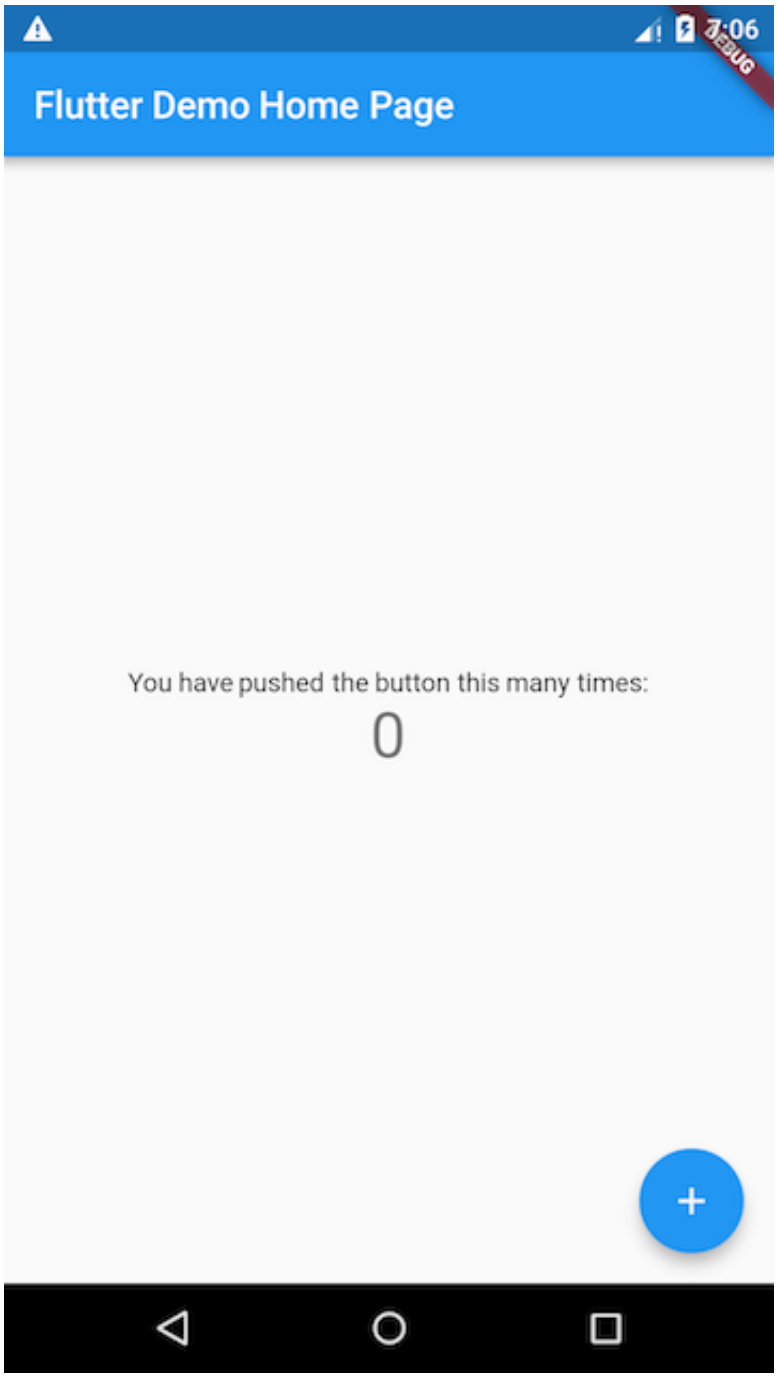
计数器应用示例

用Android Studio和VS Code创建的Flutter应用模板是一个简单的计数器示例，本节先仔细讲解一下这个计数器Demo的源码，让读者对Flutter应用程序结构有个基本了解，在随后小节中，将会基于此示例，一步一步添加一些新的功能来介绍Flutter应用的其它概念与技术。对于接下来的示例，希望读者可以跟着笔者实际动手来写一下，这样不仅可以加深印象，而且也会对介绍的概念与技术有一个真切的体会。如果你还不是很熟悉Dart或者没有移动开发经验，不用担心，只要你熟悉面向对象和基本编程概念（如变量、循环和条件控制），则可以完成本示例。

通过Android Studio和VS Code根据前面“编辑器配置与使用”一章中介绍的创建Flutter工程的方法创建一个新的Flutter工程，命名为"first_flutter_app"。创建好后，就会得到一个计数器应用的Demo。

> 注意，默认Demo示例可能随着编辑器Flutter插件版本变化而变化，本例中会介绍计数器示例的全部代码，所以不会对本示例产生影响。

我们先运行此示例，效果图如下：



该计数器示例中，每点击一次右下角带“ ➕ ”号的悬浮按钮，屏幕中央的数字就会加1。

在这个示例中，主要Dart代码是在 **lib/main.dart** 文件中，下面我们看看该示例的源码：

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

```
void main() => runApp(new MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Flutter Demo',
      theme: new ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: new MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

  @override
  _MyHomePageState createState() => new _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text(widget.title),
      ),
      body: new Center(
        child: new Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            new Text(
              'You have pushed the button this many times:',
            ),
            new Text(
              '$_counter',
```

```
              style: Theme.of(context).textTheme.display1,
            ),
          ],
        ),
      ),
      floatingActionButton: new FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: new Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for
build methods.
    );
  }
}
```

分析

1. 导入包。

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

   此行代码作用是导入了Material UI组件库。Material是一种标准的移动端和web端的视觉设计语言， Flutter默认提供了一套丰富的Material风格的UI组件。

2. 应用入口。

   ```
   void main() => runApp(new MyApp());
   ```

   - 与C/C++、Java类似，Flutter 应用中`main`函数为应用程序的入口，main函数中调用了， `runApp` 方法，它的功能是启动Flutter应用，它接受一个`Widget`参数，在本示例中它是`MyApp`类的一个实例，该参数代表Flutter应用。
   - main函数使用了(`=>`)符号，这是Dart中单行函数或方法的简写。

3. 应用结构。

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      //应用名称
      title: 'Flutter Demo',
      theme: new ThemeData(
        //蓝色主题
        primarySwatch: Colors.blue,
      ),
      //应用首页路由
      home: new MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

- `MyApp` 类代表Flutter应用，它继承了 `StatelessWidget` 类，这也就意味着应用本身也是一个widget。
- 在Flutter中，大多数东西都是widget，包括对齐(alignment)、填充(padding)和布局(layout)。
- Flutter在构建页面时，会调用组件的 `build` 方法，widget的主要工作是提供一个build()方法来描述如何构建UI界面（通常是通过组合、拼装其它基础widget）。
- `MaterialApp` 是Material库中提供的Flutter APP框架，通过它可以设置应用的名称、主题、语言、首页及路由列表等。`MaterialApp` 也是一个widget。
- `Scaffold` 是Material库中提供的页面脚手架，它包含导航栏和Body以及FloatingActionButton（如果需要的化）。
- `home` 为Flutter应用的首页，它也是一个widget。

4. 首页

```
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;
  @override
  _MyHomePageState createState() => new _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
 ...
}
```

`MyHomePage` 是应用的首页，它继承自`StatefulWidget`类，表示它是一个有状态的widget（Stateful widget）。现在，我们可以简单认为Stateful widget 和 Stateless widget有两点不同：

1. Stateful widget可以拥有状态，这些状态在widget生命周期中是可以变的，而Stateless widget是不可变的。
2. Stateful widget至少由两个类组成：
   - 一个`StatefulWidget`类。
   - 一个 State类； `StatefulWidget` 类本身是不变的，但是 State类中持有的状态在widget生命周期中可能会发生变化。

   `_MyHomePageState` 类是 `MyHomePage` 类对应的状态类。看到这里，细心的读者可能已经发现，和 `MyApp` 类不同， `MyHomePage` 类中并没有 `build` 方法，取而代之的是， `build` 方法被挪到了 `_MyHomePageState` 方法中，至于为什么这么做，先留个疑问，在分析完完整代码后再来解答。

接下来，我们看看`_MyHomePageState`中都包含哪些东西：

1. 状态。

   ```
   int _counter = 0;
   ```

   `_counter` 为保存屏幕右下角带“➕”号按钮点击次数的状态。
2. 设置状态的自增函数。

```
void _incrementCounter() {
  setState(() {
     _counter++;
  });
}
```

当按钮点击时，会调用此函数，该函数的作用是先自增`_counter`，然后调用`setState`方法。`setState`方法的作用是通知Flutter框架，有状态发生了改变，Flutter框架收到通知后，会执行`build`方法来根据新的状态重新构建界面， Flutter 对此方法做了优化，使重新执行变的很快，所以你可以重新构建任何需要更新的东西，而无需分别去修改各个widget。

3. 构建UI界面

   构建UI界面的逻辑在`build`方法中，当`MyHomePage`第一次创建时，`_MyHomePageState`类会被创建，当初始化完成后，Flutter框架会调用Widget的`build`方法来构建widget树，最终将widget树渲染到设备屏幕上。所以，我们看看`_MyHomePageState`的`build`方法中都干了什么事：

```
Widget build(BuildContext context) {
  return new Scaffold(
    appBar: new AppBar(
      title: new Text(widget.title),
    ),
    body: new Center(
      child: new Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          new Text(
            'You have pushed the button this many times:',
          ),
          new Text(
            '$_counter',
            style: Theme.of(context).textTheme.display1,
          ),
        ],
      ),
    ),
    floatingActionButton: new FloatingActionButton(
      onPressed: _incrementCounter,
      tooltip: 'Increment',
      child: new Icon(Icons.add),
    ),
  );
}
```

- Scaffold 是 Material库中提供的一个widget, 它提供了默认的导航栏、标题和包含主屏幕widget树的body属性。widget树可以很复杂。
- body的widget树中包含了一个 `Center` widget， `Center` 可以将其子widget树对其到屏幕中心， `Center` 子widget是一个 `Column` widget， `Column` 的作用是将其所有子widget沿屏幕垂直方向依次排列，此例中 `Column` 包含两个 `Text` 子widget，第一个 `Text` widget显示固定文本 “You have pushed the button this many times:”，第二个 `Text` widget显示 `_counter` 状态的数值。
- floatingActionButton是页面右下角的带“ ➕ ”的悬浮按钮，它的 `onPressed` 属性接受一个回调函数，代表它本点击后的处理器，本例中直接将 `_incrementCounter` 作为其处理函数。

现在，我们将整个流程串起来：当右下角的floatingActionButton按钮被点击之后，会调用`_incrementCounter`，在`_incrementCounter`中，首先会自增`_counter`计数器（状态），然后`setState`会通知Flutter框架状态发生变化，接着，Flutter会调用`build`方法以新的状态重新构建UI，最终显示在设备屏幕上。

为什么要将build方法放在State中，而不是放在StatefulWidget中?

现在，我们回答之前提出的问题，为什么`build()`方法在State（而不是StatefulWidget）中？这主要是为了开发的灵活性。如果将`build()`方法在StatefulWidget中则会有两个问题：

- 状态访问不便

  试想一下，如果我们的Stateful widget 有很多状态，而每次状态改变都要调用`build`方法，由于状态是保存在State中的，如果将`build`方法放在StatefulWidget中，那么构建时读取状态将会很不方便，试想一下，如果真的将`build`方法放在StatefulWidget中的话，由于构建用户界面过程需要依赖State，所以`build`方法将必须加一个`State`参数，大概是下面这样：

  ```
  Widget build(BuildContext context, State state){
      //state.counter
      ...
  }
  ```

  这样的话就只能将State的所有状态声明为公开的状态，这样才能在State类外部访问状态，但将状态设置为公开后，状态将不再具有私密性，这样依赖，对状态的修改将会变的不可控。将`build()`方法放在State中的话，构建过程则可以直接访问状态，这样会很方便。

- 继承StatefulWidget不便

  例如，Flutter中有一个动画widget的基类`AnimatedWidget`，它继承自`StatefulWidget`类。`AnimatedWidget`中引入了一个抽象方法`build(BuildContext context)`，继承自`AnimatedWidget`的动画widget都要实现这个`build`方法。现在设想一下，如果`StatefulWidget`类中已经有了一个`build`方法，正如上面所述，此时`build`方法需要接收一个state对象，这就意味着`AnimatedWidget`必须将自己的State对象(记为_animatedWidgetState)提供给其子类，因为子类需要在其`build`方法中调用

父类的`build`方法，代码可能如下：

```
class MyAnimationWidget extends AnimatedWidget{
    @override
    Widget build(BuildContext context, State state){
      //由于子类要用到AnimatedWidget的状态对象_animatedWidgetState,
      //所以AnimatedWidget必须通过某种方式将其状态对象
_animatedWidgetState
      //暴露给其子类
      super.build(context, _animatedWidgetState)
    }
}
```

这样很显然是不合理的，因为

i. `AnimatedWidget`的状态对象是`AnimatedWidget`内部实现细节，不应该暴露给外部。
ii. 如果要将父类状态暴露给子类，那么必须得有一种传递机制，而做这一套传递机制是无意义的，因为父子类之间状态的传递和子类本身逻辑是无关的。

综上所述，可以发现，对于StatefulWidget，将`build`方法放在State中，可以给开发带来很大的灵活性。

路由管理

路由(Route)在移动开发中通常指页面（Page），这跟web开发中单页应用的Route概念意义是相同的，Route在Android中通常指一个Activity，在iOS中指一个ViewController。所谓路由管理，就是管理页面之间如何跳转，通常也可被称为导航管理。这和原生开发类似，无论是Android还是iOS，导航管理都会维护一个路由栈，路由入栈(push)操作对应打开一个新页面，路由出栈(pop)操作对应页面关闭操作，而路由管理主要是指如何来管理路由栈。

示例

我们在上一节“计数器”示例的基础上，做如下修改：

1. 创建一个新路由，命名“NewRoute”

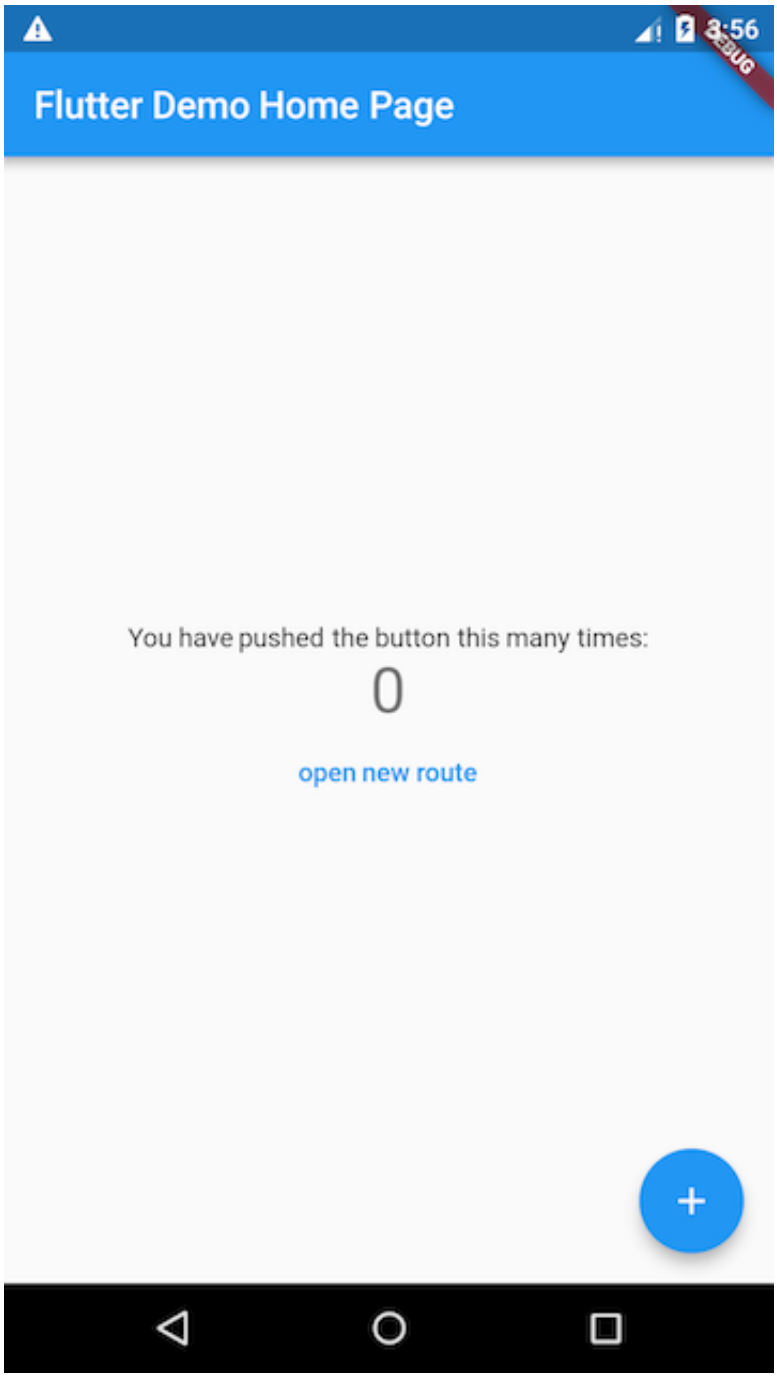
```
class NewRoute extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("New route"),
      ),
      body: Center(
        child: Text("This is new route"),
      ),
    );
  }
}
```

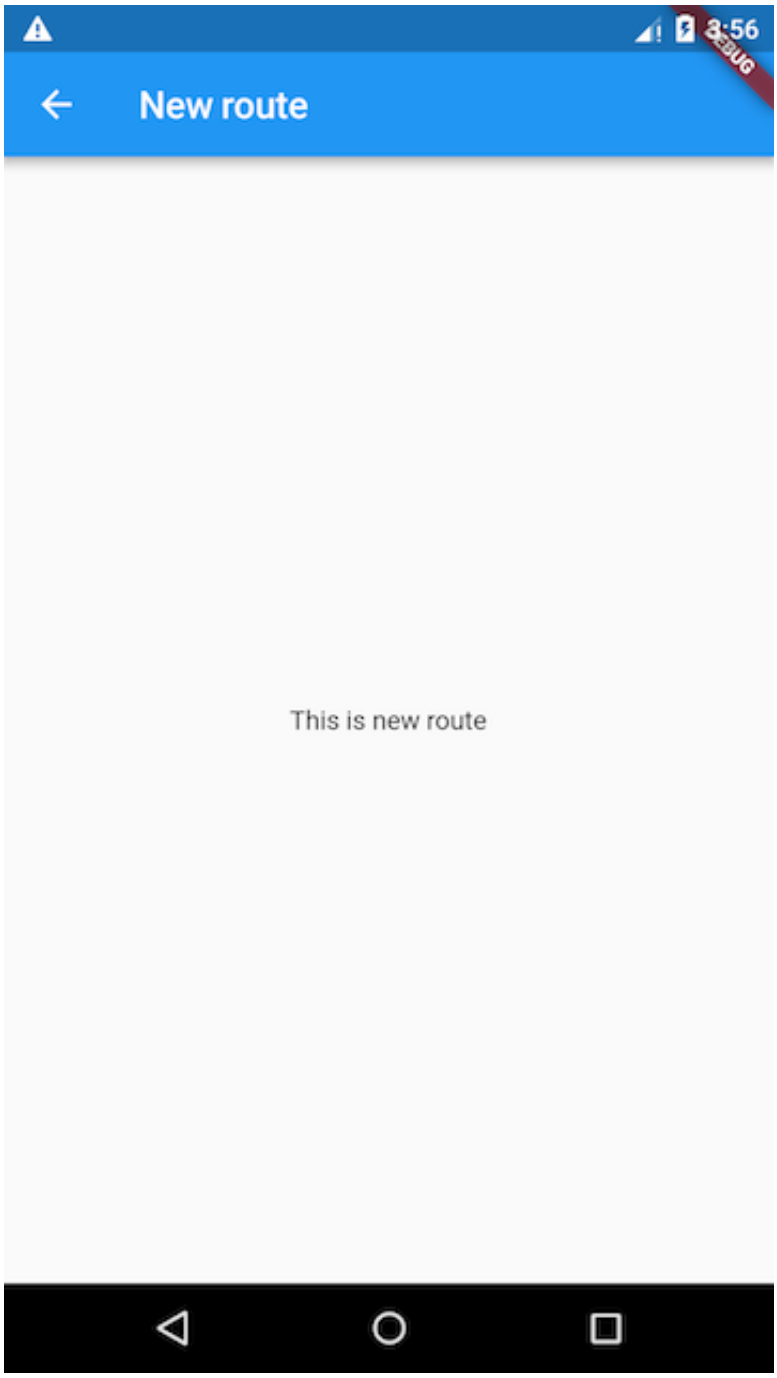
新路由继承自`StatelessWidget`，界面很简单，在页面中间显示一句"This is new route"。

2. 在`_MyHomePageState.build`方法中的`Column`的子widget中添加一个按钮（`FlatButton`）：

```
Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
      ... //省略无关代码
      FlatButton(
         child: Text("open new route"),
         textColor: Colors.blue,
         onPressed: () {
          //导航到新路由
          Navigator.push( context,
           new MaterialPageRoute(builder: (context) {
                  return new NewRoute();
              }));
          },
         ),
       ],
 )
```

我们添加了一个打开新路由的按钮，并将按钮文字颜色设置为蓝色，点击该按钮后就会打开新的路由页面。


8:56
DEBUG
Flutter Demo Home Page
You have pushed the button this many times:
0
open new route
+



MaterialPageRoute

`MaterialPageRoute`继承自`PageRoute`类，`PageRoute`类是一个抽象类，表示占有整个屏幕空间的一个模态路由页面，它还定义了路由构建及切换时过渡动画的相关接口及属性。`MaterialPageRoute`是Material组件库的一个Widget，它可以针对不同平台，实现与平台页面切换动画风格一致的路由切换动画：

- 对于Android，当打开新页面时，新的页面会从屏幕底部滑动到屏幕顶部；当关闭页面时，当前页面会从屏幕顶部滑动到屏幕底部后消失，同时上一个页面会显示到屏幕上。
- 对于iOS，当打开页面时，新的页面会从屏幕右侧边缘一致滑动到屏幕左边，直到新页面全部显示到屏幕上，而上一个页面则会从当前屏幕滑动到屏幕左侧而消失；当关闭页面时，正好相反，当前页面会从屏幕右侧滑出，同时上一个页面会从屏幕左侧滑入。

下面我们介绍一下`MaterialPageRoute`构造函数的各个参数的意义：

```
MaterialPageRoute({
  WidgetBuilder builder,
  RouteSettings settings,
  bool maintainState = true,
  bool fullscreenDialog = false,
})
```

- `builder`是一个WidgetBuilder类型的回调函数，它的作用是构建路由页面的具体内容，返回值是一个widget。我们通常要实现此回调，返回新路由的实例。
- `settings`包含路由的配置信息，如路由名称、是否初始路由（首页）。
- `maintainState`：默认情况下，当入栈一个新路由时，原来的路由仍然会被保存在内存中，如果想在路由没用的时候释放其所占用的所有资源，可以设置`maintainState`为false。
- `fullscreenDialog`表示新的路由页面是否是一个全屏的模态对话框，在iOS中，如果`fullscreenDialog`为`true`，新页面将会从屏幕底部滑入（而不是水平方向）。

> 如果想自定义路由切换动画，可以自己继承PageRoute来实现，我们将在后面介绍动画时，实现一个自定义的路由Widget。

Navigator

`Navigator`是一个路由管理的widget，它通过一个栈来管理一个路由widget集合。通常当前屏幕显示的页面就是栈顶的路由。`Navigator`提供了一系列方法来管理路由栈，在此我们只介绍其最常用的两个方法：

Future push(BuildContext context, Route route)

将给定的路由入栈（即打开新的页面），返回值是一个`Future`对象，用以接收新路由出栈（及关闭）时的返回数据。

bool pop(BuildContext context, [result])

将栈顶路由出栈，`result`为页面关闭时返回给上一个页面的数据。

`Navigator`还有很多其它方法，
如`Navigator.replace`、`Navigator.popUntil`等，详情请参考API文档或SDK源码注释，在此不再赘述。下面我们还需要介绍一下路由相关的另一个概念“命名路由”。

命名路由

所谓命名路由（Named Route）即给路由起一个名字，然后可以通过路由名字直接打开新的路由。这为路由管理带来了一种直观、简单的方式。

路由表

要想使用命名路由，我们必须先提供并注册一个路由表（routing table），这样应用程序才知道哪个名称与哪个路由Widget对应。路由表的定义如下：

```
Map<String, WidgetBuilder> routes;
```

它是一个`Map`， key 为路由的名称，是个字符串；value是个builder回调函数，用于生成相应的路由Widget。我们在通过路由名称入栈新路由时，应用会根据路由名称在路由表中找到对应的WidgetBuilder回调函数，然后调用该回调函数生成路由widget并返回。

注册路由表

我们需要先注册路由表后，我们的Flutter应用才能正确处理命名路由的跳转。注册方式很简单，我们回到之前“计数器”的示例，然后在`MyApp`类的`build`方法中找到`MaterialApp`，添加`routes`属性，代码如下：

```
return new MaterialApp(
  title: 'Flutter Demo',
  theme: new ThemeData(
    primarySwatch: Colors.blue,
  ),
  //注册路由表
  routes:{
   "new_page":(context)=>NewRoute(),
  } ,
  home: new MyHomePage(title: 'Flutter Demo Home Page'),
);
```

现在我们就完成了路由表的注册。

通过路由名打开新路由页

要通过路由名称来打开新路由，可以使用：

```
Future pushNamed(BuildContext context, String routeName)
```

`Navigator`除了`pushNamed`方法，还有`pushReplacementNamed`等其他管理命名路由的方法，读者可以自行查看API文档。

接下来我们通过路由名来打开新的路由页，修改`FlatButton`的`onPressed`回调代码，改为：

```
onPressed: () {
  Navigator.pushNamed(context, "new_page");
  //Navigator.push(context,
  //  new MaterialPageRoute(builder: (context) {
  //  return new NewRoute();
  //}));
},
```

热重载应用，再次点击“open new route”按钮，依然可以打开新的路由页。

命名路由的优缺点

命名路由的最大优点是直观，我们可以通过语义化的字符串来管理路由。但其有一个明显的缺点：不能直接传递路由参数。举个例子，假设有一个新路由EchoRoute，它的功能是接受一个字符串参数`tip`，然后再在屏幕中心将`tip`的内容显示出来，代码如下：

```
class EchoRoute extends StatelessWidget {
  EchoRoute(this.tip);
  final String tip;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Echo route"),
      ),
      body: Center(
        //回显tip内容
        child: Text(tip),
      ),
    );
  }
}
```

如果我们使用命名参数，就必须将路由提前注册到路由表中，所以就无法动态修改`tip`参数，如：

```
{
    "tip_widgets":(context)=>EchoRoute("内容固定")
}
```

综上所述，我们可以看到当路由需要参数时，使用命名路由则不够灵活。

包管理

一个完整的应用程序往往会依赖很多第三方包，正如在原生开发中，Android使用Gradle来管理依赖，iOS用Cocoapods或Carthage来管理依赖，而Flutter也有自己的依赖管理工具，本节我们主要介绍一下flutter如何使用配置文件`pubspec.yaml`（位于项目根目录）来管理第三方依赖包。

YAML是一种直观、可读性高并且容易被人类阅读的文件格式，它和xml或Json相比，它语法简单并非常容易解析，所以YAML常用于配置文件，Flutter也是用yaml文件作为其配置文件，Flutter项目默认的配置文件是`pubspec.yaml`，我们看一个简单的示例：

```yaml
name: flutter_in_action
description: First Flutter application.

version: 1.0.0+1

dependencies:
  flutter:
    sdk: flutter
  cupertino_icons: ^0.1.2

dev_dependencies:
  flutter_test:
    sdk: flutter

flutter:
  uses-material-design: true
```

下面，我们逐一解释一下各个字段的意义：

- name：应用或包名称。
- description: 应用或包的描述、简介。
- version：应用或包的版本号。
- dependencies：应用或包依赖的其它包或插件。
- dev_dependencies：开发环境依赖的工具包（而不是flutter应用本身依赖的包）。
- flutter：flutter相关的配置选项。

如果我们的Flutter应用本身依赖某个包，我们需要将所依赖的包添加到`dependencies`下，接下来我们通过一个例子来演示一下如何依赖、下载并使用第三方包。

Pub仓库

Pub（https://pub.dartlang.org/ ）是Google官方的Dart Packages仓库，类似于node中的npm仓库，android中的jcenter，我们可以在上面查找我们需要的包和插件，也可以向pub发布我们的包和插件。我们将在后面的章节中介绍如何向pub发布我们的包和插件。

示例

接下来，我们实现一个显示随机字符串的widget。有一个名为“english_words”的开源软件包，其中包含数千个常用的英文单词以及一些实用功能。我们首先在pub上找到english_words这个包，确定其最新的版本号和是否支持Flutter。

我们看到“english_words”包最新的版本是3.1.3，并且支持flutter，接下来：

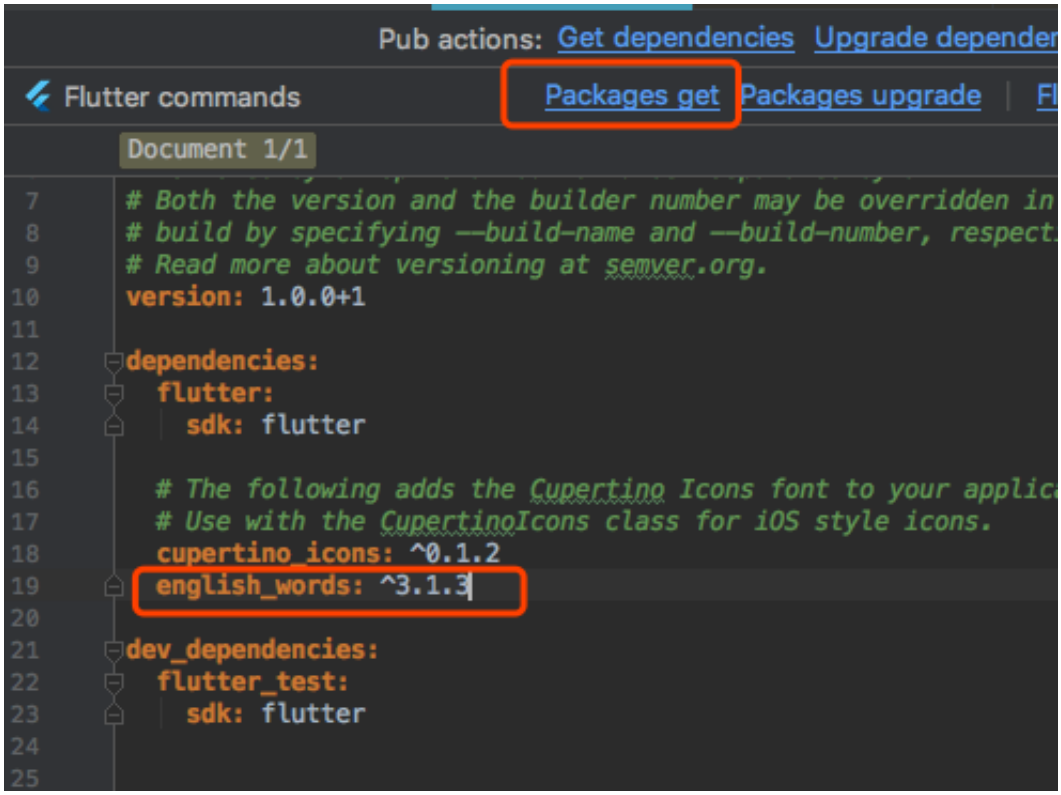
1. 将english_words（3.1.3版本）添加到依赖项列表，如下：

```
dependencies:
  flutter:
    sdk: flutter

  cupertino_icons: ^0.1.0
  # 新添加的依赖
  english_words: ^3.1.3
```

2. 下载包

   在Android Studio的编辑器视图中查看pubspec.yaml时，单击右上角的 **Packages get** 。



这会将依赖包安装到您的项目。您可以在控制台中看到以下内容：

```
flutter packages get
Running "flutter packages get" in flutter_in_action...
Process finished with exit code 0
```

你也可以在控制台，定位到当前工程目录，然后手动运行`flutter packages get`命令来下载依赖包。

3. 引入`english_words`包。

   ```
   import 'package:english_words/english_words.dart';
   ```

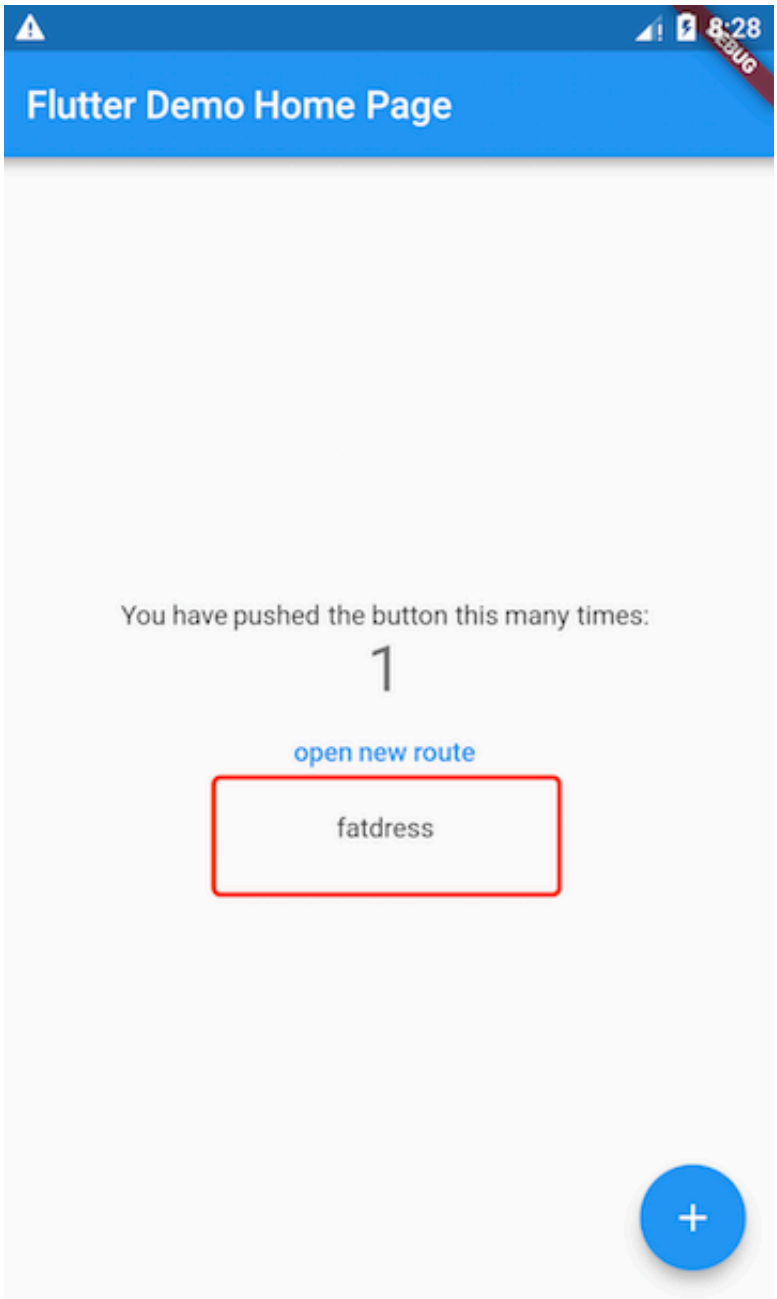
   在输入时，Android Studio会自动提供有关库导入的建议选项。导入后该行代码将会显示为灰色，表示导入的库尚未使用。

4. 使用`english_words`包来生成随机字符串。

   ```
   class RandomWordsWidget extends StatelessWidget {
     @override
     Widget build(BuildContext context) {
      // 生成随机字符串
       final wordPair = new WordPair.random();
       return Padding(
         padding: const EdgeInsets.all(8.0),
         child: new Text(wordPair.toString()),
       );
     }
   }
   ```

   我们将`RandomWordsWidget`添加到"计数器"示例的首页`MyHomePage`的`Column`的子widget中。

5. 如果应用程序正在运行，请使用热重载按钮 () 更新正在运行的应用程序。每次单击热重载或保存项目时，都会在正在运行的应用程序中随机选择不同的单词对。 这是因为单词对是在`build`方法内部生成的。每次热更新时，`build`方法都会被执行。



其它依赖方式

上文所述的依赖方式是依赖pub仓库的。但我们还可以依赖本地包和git仓库。

- 依赖本地包

  如果我们正在本地开发一个包，包名为pkg1，我们可以通过下面方式依赖：

  ```
  dependencies:
      pkg1:
          path: ../../code/pkg1
  ```

  路径可以是相对的，也可以是绝对的。

- 依赖Git：你也可以依赖存储在Git仓库中的包。如果软件包位于仓库的根目录中，请使用以下语法

  ```
  dependencies:
    pkg1:
      git:
        url: git://github.com/xxx/pkg1.git
  ```

  上面假定包位于Git存储库的根目录中。如果不是这种情况，可以使用path参数指定相对位置，例如：

  ```
  dependencies:
    package1:
      git:
        url: git://github.com/flutter/packages.git
        path: packages/package1
  ```

总结

本节介绍了引用、下载、使用一个包的整体流程，并在后面介绍了多种不同的引入方式，这些是Flutter开发中常用的，但Dart的dependencies还有一些其它依赖方式，完整的内容读者可以自行查看：
https://www.dartlang.org/tools/pub/dependencies 。

资源管理

Flutter应用程序可以包含代码和 assets（有时称为资源）。assets是会打包到程序安装包中的，可在运行时访问。常见类型的assets包括静态数据（例如JSON文件）、配置文件、图标和图片（JPEG，WebP，GIF，动画WebP / GIF，PNG，BMP和WBMP）等。

指定 assets

和包管理一样，Flutter也使用`pubspec.yaml`文件来管理应用程序所需的资源。举一个例子:

```
flutter:
  assets:
    - assets/my_icon.png
    - assets/background.png
```

`assets`指定应包含在应用程序中的文件。 每个asset都通过相对于`pubspec.yaml`文件所在位置的显式路径进行标识。asset的声明顺序是无关紧要的。asset的实际目录可以是任意文件夹（在本示例中是assets）。

在构建期间，Flutter将asset放置到称为 *asset bundle* 的特殊存档中，应用程序可以在运行时读取它们（但不能修改）。

Asset 变体（variant）

构建过程支持asset变体的概念：不同版本的asset可能会显示在不同的上下文中。在`pubspec.yaml`的assets部分中指定asset路径时，构建过程中，会在相邻子目录中查找具有相同名称的任何文件。这些文件随后会与指定的asset一起被包含在asset bundle中。

例如，如果应用程序目录中有以下文件:

- .../pubspec.yaml
- .../graphics/my_icon.png
- .../graphics/background.png
- .../graphics/dark/background.png
- ...etc.

然后`pubspec.yaml`文件中只需包含:

```
flutter:
  assets:
    - graphics/background.png
```

那么这两个`graphics/background.png`和`graphics/dark/background.png`都将包含在您的asset bundle中。前者被认为是_main asset_ （主资源），后者被认为是一种变体（variant）。

在选择匹配当前设备分辨率的图片时，Flutter会使用到asset变体（见下文），将来，Flutter可能会将这种机制扩展到本地化、阅读提示等方面。

加载 assets

您的应用可以通过`AssetBundle`对象访问其asset 。有两种主要方法允许从Asset bundle中加载字符串或图片(二进制)文件。

加载文本assets

- 通过`rootBundle`对象加载：每个Flutter应用程序都有一个`rootBundle`对象， 通过它可以轻松访问主资源包，直接使用`package:flutter/services.dart`中全局静态的`rootBundle`对象来加载asset即可。
- 通过`DefaultAssetBundle`加载：建议使用`DefaultAssetBundle`来获取当前BuildContext的AssetBundle。 这种方法不是使用应用程序构建的默认asset bundle，而是使父级widget在运行时动态替换的不同的AssetBundle，这对于本地化或测试场景很有用。

通常，可以使用`DefaultAssetBundle.of()`在应用运行时来间接加载asset（例如JSON文件），而在widget上下文之外，或其它`AssetBundle`句柄不可用时，可以使用`rootBundle`直接加载这些asset，例如：

```dart
import 'dart:async' show Future;
import 'package:flutter/services.dart' show rootBundle;

Future<String> loadAsset() async {
  return await rootBundle.loadString('assets/config.json');
}
```

加载图片

类似于原生开发，Flutter也可以为当前设备加载适合其分辨率的图像。

声明分辨率相关的图片 assets

`AssetImage` 可以将asset的请求逻辑映射到最接近当前设备像素比例(dpi)的asset。为了使这种映射起作用，必须根据特定的目录结构来保存asset：

- .../image.png
- .../**M**x/image.png
- .../**N**x/image.png
- ...etc.

其中M和N是数字标识符，对应于其中包含的图像的分辨率，也就是说，它们指定不同设备像素比例的图片。

主资源默认对应于1.0倍的分辨率图片。看一个例子：

- .../my_icon.png
- .../2.0x/my_icon.png
- .../3.0x/my_icon.png

在设备像素比率为1.8的设备上，`.../2.0x/my_icon.png` 将被选择。对于2.7的设备像素比率，`.../3.0x/my_icon.png` 将被选择。

如果未在 `Image` widget上指定渲染图像的宽度和高度，那么 `Image` widget将占用与主资源相同的屏幕空间大小。 也就是说，如果 `.../my_icon.png` 是72px乘72px，那么 `.../3.0x/my_icon.png` 应该是216px乘216px; 但如果未指定宽度和高度，它们都将渲染为72像素×72像素（以逻辑像素为单位）。

`pubspec.yaml`中asset部分中的每一项都应与实际文件相对应，但主资源项除外。当主资源缺少某个资源时，会按分辨率从低到高的顺序去选择，也就是说1x中没有的话会在2x中找，2x中还没有的话就在3x中找。

加载图片

要加载图片，可以使用`AssetImage`类。例如，我们可以从上面的asset声明中加载背景图片：

```
Widget build(BuildContext context) {
  return new DecoratedBox(
    decoration: new BoxDecoration(
      image: new DecorationImage(
        image: new AssetImage('graphics/background.png'),
      ),
    ),
  );
}
```

注意，`AssetImage`并非是一个widget，它实际上是一个`ImageProvider`，有些时候你可能期望直接得到一个显示图片的widget，那么你可以使用`Image.asset()`方法，如：

```
Widget build(BuildContext context) {
  return Image.asset('graphics/background.png');
}
```

使用默认的 asset bundle 加载资源时，内部会自动处理分辨率等，这些处理对开发者来说是无感知的。（如果使用一些更低级别的类，
如`ImageStream`或`ImageCache`时你会注意到有与缩放相关的参数）

依赖包中的资源图片

要加载依赖包中的图像，必须给`AssetImage`提供`package`参数。

例如，假设您的应用程序依赖于一个名为“my_icons”的包，它具有如下目录结构：

- .../pubspec.yaml
- .../icons/heart.png
- .../icons/1.5x/heart.png
- .../icons/2.0x/heart.png
- ...etc.

然后加载图像，使用：

```
 new AssetImage('icons/heart.png', package: 'my_icons')
```

或

```
new Image.asset('icons/heart.png', package: 'my_icons')
```

注意：包在使用本身的资源时也应该加上`package`参数来获取。

打包包中的 assets

如果在`pubspec.yaml`文件中声明了期望的资源，它将会打包到相应的package中。特别是，包本身使用的资源必须在`pubspec.yaml`中指定。

包也可以选择在其`lib/`文件夹中包含未在其`pubspec.yaml`文件中声明的资源。在这种情况下，对于要打包的图片，应用程序必须在`pubspec.yaml`中指定包含哪些图像。 例如，一个名为“fancy_backgrounds”的包，可能包含以下文件：

- .../lib/backgrounds/background1.png
- .../lib/backgrounds/background2.png
- .../lib/backgrounds/background3.png

要包含第一张图像，必须在`pubspec.yaml`的assets部分中声明它：

```
flutter:
  assets:
    - packages/fancy_backgrounds/backgrounds/background1.png
```

`lib/`是隐含的，所以它不应该包含在资产路径中。

特定平台 assets

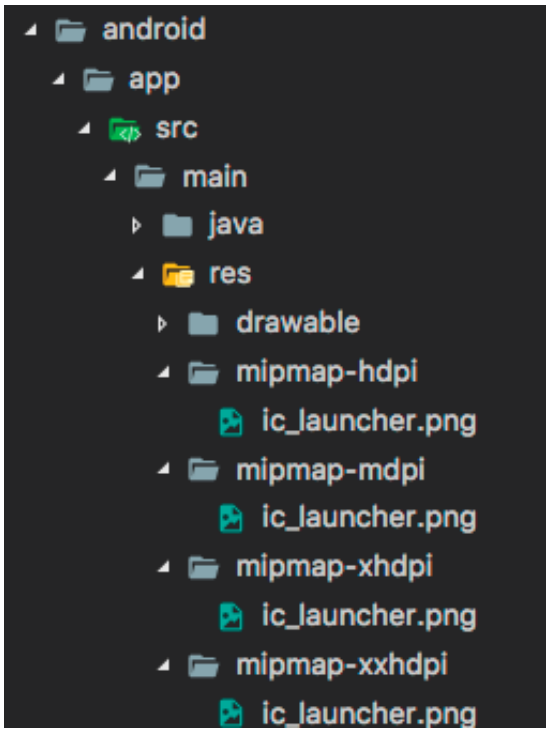
上面的资源都是flutter应用中的，这些资源只有在Flutter框架运行之后才能使用，如果要给我们的应用设置APP图标或者添加启动图，那我们必须使用特定平台的assets。

设置APP图标

更新Flutter应用程序启动图标的方式与在本机Android或iOS应用程序中更新启动图标的方式相同。

- Android

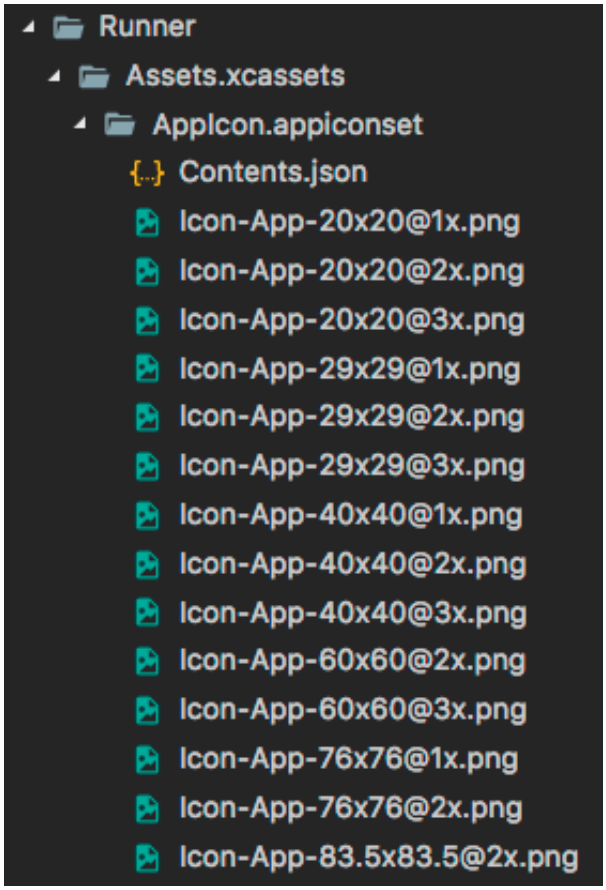
  在Flutter项目的根目录中，导航到`.../android/app/src/main/res`目录，里面包含了各种资源文件夹（如`mipmap-hdpi`已包含占位符图像”ic_launcher.png”）。 只需按照Android开发人员指南中的说明， 将其替换为所需的资源，并遵守每种屏幕密度（dpi）的建议图标大小标准。



  > **注意:** 如果您重命名.png文件，则还必须在您`AndroidManifest.xml`的`<application>`标签的`android:icon`属性中更新名称。

- iOS

在Flutter项目的根目录中，导航到`.../ios/Runner`。该目录中`Assets.xcassets/AppIcon.appiconset`已经包含占位符图片。 只需将它们替换为适当大小的图片。保留原始文件名称。



更新启动页

在Flutter框架加载时，Flutter会使用本地平台机制绘制启动页。此启动页将持续到Flutter渲染应用程序的第一帧时。

> **注意:** 这意味着如果您不在应用程序的`main()`方法中调用runApp 函数 （或者更具体地说，如果您不调用`window.render`去响应`window.onDrawFrame`）的话， 启动屏幕将永远持续显示。

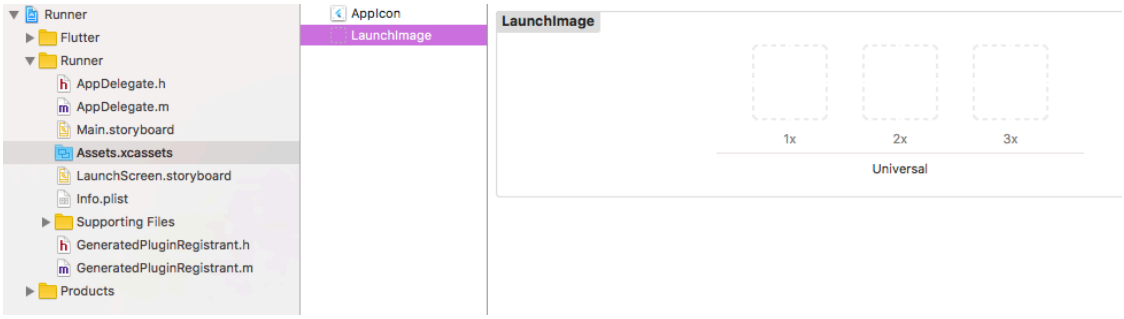
Android

要将启动屏幕（splash screen）添加到您的Flutter应用程序， 请导航至`.../android/app/src/main`。在`res/drawable/launch_background.xml`，通过自定义drawable来实现自定义启动界面（你也可以直接换一张图片）。

iOS

要将图片添加到启动屏幕（splash screen）的中心，请导航至`.../ios/Runner`。在`Assets.xcassets/LaunchImage.imageset`， 拖入图片，并命名为`LaunchImage.png`、`LaunchImage@2x.png`、`LaunchImage@3x.png`。 如果你使用不同的文件名，那您还必须更新同一目录中的`Contents.json`文件，图片的具体尺寸可以查看苹果官方的标准。

您也可以通过打开Xcode完全自定义storyboard。在Project Navigator中导航到`Runner/Runner`然后通过打开`Assets.xcassets`拖入图片，或者通过在LaunchScreen.storyboard中使用Interface Builder进行自定义。



调试Flutter应用

有各种各样的工具和功能来帮助调试Flutter应用程序。

Dart 分析器

在运行应用程序前，请运行`flutter analyze`测试你的代码。这个工具（它是`dartanalyzer`工具的一个包装）将分析你的代码并帮助你发现可能的错误。 如果你使用IntelliJ的Flutter插件，那么已经自动启用了。

Dart分析器大量使用了代码中的类型注释来帮助追踪问题。我们鼓励您在任何地方使用它们（避免var、无类型的参数、无类型的列表文字等），因为这是追踪问题的最快的方式。

Dart Observatory（语句级的单步调试和分析器）

如果您使用`flutter run`启动应用程序，那么当它运行时，您可以打开Observatory URL的Web页面（例如Observatory监听http://127.0.0.1:8100/）， 直接使用语句级单步调试器连接到您的应用程序。如果您使用的是IntelliJ，则还可以使用其内置的调试器来调试您的应用程序。

Observatory 同时支持分析、检查堆等。有关Observatory的更多信息请参考 Observatory 文档.

如果您使用Observatory进行分析，请确保通过`--profile`选项来运行`flutter run`命令来运行应用程序。 否则，配置文件中将出现的主要问题将是调试断言，以验证框架的各种不变量（请参阅下面的“调试模式断言”）。

`debugger()`声明

当使用Dart Observatory（或另一个Dart调试器，例如IntelliJ IDE中的调试器）时，可以使用该`debugger()`语句插入编程式断点。要使用这个，你必须添加`import 'dart:developer';`到相关文件顶部。

`debugger()`语句采用一个可选`when`参数，您可以指定该参数仅在特定条件为真时中断，如下所示：

```
void someFunction(double offset) {
  debugger(when: offset > 30.0);
  // ...
}
```

`print`、`debugPrint`、`flutter logs`

Dart `print()`功能将输出到系统控制台，您可以使用`flutter logs`了查看它（基本上是一个包装`adb logcat`）。

如果你一次输出太多，那么Android有时会丢弃一些日志行。为了避免这种情况，您可以使用Flutter的`foundation`库中的`debugPrint()`。 这是一个封装print，它将输出限制在一个级别，避免被Android内核丢弃。

Flutter框架中的许多类都有`toString`实现。按照惯例，这些输出通常包括对象的`runtimeType`单行输出，通常在表单中ClassName(more information about this instance...)。 树中使用的一些类也具有`toStringDeep`，从该点返回整个子树的多行描述。已一些具有详细信息`toString`的类会实现一个`toStringShort`，它只返回对象的类型或其他非常简短的（一个或两个单词）描述。

调试模式断言

在开发过程中，强烈建议您使用Flutter的“调试”模式，有时也称为“checked”模式（注意：Dart2.0后“checked”被废除，可以使用“strong” mode）。 如果您使用`flutter run`运行程序。在这种模式下，Dart assert语句被启用，并且Flutter框架使用它来执行许多运行时检查来验证是否违反一些不可变的规则。

当一个不可变的规则被违反时，它被报告给控制台，并带有一些上下文信息来帮助追踪问题的根源。

要关闭调试模式并使用发布模式，请使用`flutter run --release`运行您的应用程序。 这也关闭了Observatory调试器。一个中间模式可以关闭除Observatory之外所有调试辅助工具的，称为“profile mode”，用`--profile`替代`--release`即可。

调试应用程序层

Flutter框架的每一层都提供了将其当前状态或事件转储(dump)到控制台（使用`debugPrint`）的功能。

Widget 层

要转储Widgets库的状态，请调用`debugDumpApp()`。 只要应用程序已经构建了至少一次（即在调用`build()`之后的任何时间），您可以在应用程序未处于构建阶段（即，不在`build()`方法内调用 ）的任何时间调用此方法（在调用`runApp()`之后）。

如, 这个应用程序:

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

void main() {
  runApp(
    new MaterialApp(
      home: new AppHome(),
    ),
  );
}

class AppHome extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new Material(
      child: new Center(
        child: new FlatButton(
          onPressed: () {
            debugDumpApp();
          },
          child: new Text('Dump App'),
        ),
      ),
    );
  }
}
```

...会输出这样的内容（精确的细节会根据框架的版本、设备的大小等等而变化）：

```
I/flutter ( 6559): WidgetsFlutterBinding - CHECKED MODE
I/flutter ( 6559): RenderObjectToWidgetAdapter<RenderBox>
([GlobalObjectKey RenderView(497039273)]; renderObject: RenderView)
I/flutter ( 6559):  └MaterialApp(state:
_MaterialAppState(1009803148))
I/flutter ( 6559):   └ScrollConfiguration()
I/flutter ( 6559):    └AnimatedTheme(duration: 200ms; state:
_AnimatedThemeState(543295893; ticker inactive;
ThemeDataTween(ThemeData(Brightness.light Color(0xff2196f3) etc...)
→ null)))
I/flutter ( 6559):     └Theme(ThemeData(Brightness.light
Color(0xff2196f3) etc...))
I/flutter ( 6559):      └WidgetsApp([GlobalObjectKey
_MaterialAppState(1009803148)]; state: _WidgetsAppState(552902158))
I/flutter ( 6559):       └CheckedModeBanner()
I/flutter ( 6559):        └Banner()
I/flutter ( 6559):         └CustomPaint(renderObject:
```

```
RenderCustomPaint)
I/flutter ( 6559):          └DefaultTextStyle(inherit: true; color:
Color(0xd0ff0000); family: "monospace"; size: 48.0; weight: 900;
decoration: double Color(0xffffff00) TextDecoration.underline)
I/flutter ( 6559):           └MediaQuery(MediaQueryData(size:
Size(411.4, 683.4), devicePixelRatio: 2.625, textScaleFactor: 1.0,
padding: EdgeInsets(0.0, 24.0, 0.0, 0.0)))
I/flutter ( 6559):            └LocaleQuery(null)
I/flutter ( 6559):             └Title(color: Color(0xff2196f3))
I/flutter ( 6559):
 └Navigator([GlobalObjectKey<NavigatorState>
_WidgetsAppState(552902158)]; state: NavigatorState(240327618;
tracking 1 ticker))
I/flutter ( 6559):              └Listener(listeners: down, up,
cancel; behavior: defer-to-child; renderObject:
RenderPointerListener)
I/flutter ( 6559):               └AbsorbPointer(renderObject:
RenderAbsorbPointer)
I/flutter ( 6559):                └Focus([GlobalKey 489139594];
state: _FocusState(739584448))
I/flutter ( 6559):                 └Semantics(container: true;
renderObject: RenderSemanticsAnnotations)
I/flutter ( 6559):                  └_FocusScope(this scope has
focus; focused subscope: [GlobalObjectKey MaterialPageRoute<Null>
(875520219)])
I/flutter ( 6559):                   └Overlay([GlobalKey
199833992]; state: OverlayState(619367313; entries:
[OverlayEntry@248818791(opaque: false; maintainState: false),
OverlayEntry@837336156(opaque: false; maintainState: true)]))
I/flutter ( 6559):                    └_Theatre(renderObject:
_RenderTheatre)
I/flutter ( 6559):                     └Stack(renderObject:
RenderStack)
I/flutter ( 6559):                      ├_OverlayEntry([GlobalKey
612888877]; state: _OverlayEntryState(739137453))
I/flutter ( 6559):                      | └IgnorePointer(ignoring:
false; renderObject: RenderIgnorePointer)
I/flutter ( 6559):                      |  └ModalBarrier()
I/flutter ( 6559):                      |   └Semantics(container:
true; renderObject: RenderSemanticsAnnotations)
I/flutter ( 6559):                      |    └GestureDetector()
I/flutter ( 6559):                      |
 └RawGestureDetector(state: RawGestureDetectorState(39068508;
gestures: tap; behavior: opaque))
I/flutter ( 6559):                      |
 └_GestureSemantics(renderObject: RenderSemanticsGestureHandler)
I/flutter ( 6559):                      |
```

```
 └Listener(listeners: down; behavior: opaque; renderObject:
RenderPointerListener)
I/flutter ( 6559):                        |
 └ConstrainedBox(BoxConstraints(biggest); renderObject:
RenderConstrainedBox)
I/flutter ( 6559):                        └_OverlayEntry([GlobalKey
727622716]; state: _OverlayEntryState(279971240))
I/flutter ( 6559):                         └_ModalScope([GlobalKey
816151164]; state: _ModalScopeState(875510645))
I/flutter ( 6559):                          └Focus([GlobalObjectKey
MaterialPageRoute<Null>(875520219)]; state: _FocusState(331487674))
I/flutter ( 6559):                           └Semantics(container:
true; renderObject: RenderSemanticsAnnotations)
I/flutter ( 6559):                            └_FocusScope(this scope
has focus)
I/flutter ( 6559):                             └Offstage(offstage:
false; renderObject: RenderOffstage)
I/flutter ( 6559):
 └IgnorePointer(ignoring: false; renderObject: RenderIgnorePointer)
I/flutter ( 6559):
 └_MountainViewPageTransition(animation: AnimationController(⏭
1.000; paused; for MaterialPageRoute<Null>
(/))➩ProxyAnimation➩Cubic(0.40, 0.00, 0.20, 1.00)➩Tween<Offset>
(Offset(0.0, 1.0) → Offset(0.0, 0.0))➩Offset(0.0, 0.0); state:
_AnimatedState(552160732))
I/flutter ( 6559):
 └SlideTransition(animation: AnimationController(⏭ 1.000; paused;
for MaterialPageRoute<Null>(/))➩ProxyAnimation➩Cubic(0.40, 0.00,
0.20, 1.00)➩Tween<Offset>(Offset(0.0, 1.0) → Offset(0.0,
0.0))➩Offset(0.0, 0.0); state: _AnimatedState(714726495))
I/flutter ( 6559):
 └FractionalTranslation(renderObject: RenderFractionalTranslation)
I/flutter ( 6559):
 └RepaintBoundary(renderObject: RenderRepaintBoundary)
I/flutter ( 6559):
 └PageStorage([GlobalKey 619728754])
I/flutter ( 6559):
 └_ModalScopeStatus(active)
I/flutter ( 6559):                                          └AppHome()
I/flutter ( 6559):
 └Material(MaterialType.canvas; elevation: 0; state:
_MaterialState(780114997))
I/flutter ( 6559):
 └AnimatedContainer(duration: 200ms; has background; state:
_AnimatedContainerState(616063822; ticker inactive; has
background))
I/flutter ( 6559):
```

```
 └Container(bg: BoxDecoration())
I/flutter ( 6559):
 └DecoratedBox(renderObject: RenderDecoratedBox)
I/flutter ( 6559):
 └Container(bg: BoxDecoration(backgroundColor: Color(0xfffafafa)))
I/flutter ( 6559):
 └DecoratedBox(renderObject: RenderDecoratedBox)
I/flutter ( 6559):
 └NotificationListener<LayoutChangedNotification>()
I/flutter ( 6559):
 └_InkFeature([GlobalKey ink renderer]; renderObject:
_RenderInkFeatures)
I/flutter ( 6559):
 └AnimatedDefaultTextStyle(duration: 200ms; inherit: false; color:
Color(0xdd000000); family: "Roboto"; size: 14.0; weight: 400;
baseline: alphabetic; state:
_AnimatedDefaultTextStyleState(427742350; ticker inactive))
I/flutter ( 6559):
 └DefaultTextStyle(inherit: false; color: Color(0xdd000000); family:
"Roboto"; size: 14.0; weight: 400; baseline: alphabetic)
I/flutter ( 6559):
 └Center(alignment: Alignment.center; renderObject:
RenderPositionedBox)
I/flutter ( 6559):
 └FlatButton()
I/flutter ( 6559):
 └MaterialButton(state: _MaterialButtonState(398724090))
I/flutter ( 6559):
 └ConstrainedBox(BoxConstraints(88.0<=w<=Infinity, h=36.0);
renderObject: RenderConstrainedBox relayoutBoundary=up1)
I/flutter ( 6559):
 └AnimatedDefaultTextStyle(duration: 200ms; inherit: false; color:
Color(0xdd000000); family: "Roboto"; size: 14.0; weight: 500;
baseline: alphabetic; state:
_AnimatedDefaultTextStyleState(315134664; ticker inactive))
I/flutter ( 6559):
 └DefaultTextStyle(inherit: false; color: Color(0xdd000000); family:
"Roboto"; size: 14.0; weight: 500; baseline: alphabetic)
I/flutter ( 6559):
 └IconTheme(color: Color(0xdd000000))
I/flutter ( 6559):
 └InkWell(state: _InkResponseState<InkResponse>(369160267))
I/flutter ( 6559):
 └GestureDetector()
I/flutter ( 6559):
 └RawGestureDetector(state: RawGestureDetectorState(175370983;
gestures: tap; behavior: opaque))
```

```
I/flutter ( 6559):
 └_GestureSemantics(renderObject: RenderSemanticsGestureHandler
relayoutBoundary=up2)
I/flutter ( 6559):
 └Listener(listeners: down; behavior: opaque; renderObject:
RenderPointerListener relayoutBoundary=up3)
I/flutter ( 6559):
 └Container(padding: EdgeInsets(16.0, 0.0, 16.0, 0.0))
I/flutter ( 6559):
 └Padding(renderObject: RenderPadding relayoutBoundary=up4)
I/flutter ( 6559):
 └Center(alignment: Alignment.center; widthFactor: 1.0;
renderObject: RenderPositionedBox relayoutBoundary=up5)
I/flutter ( 6559):
 └Text("Dump App")
I/flutter ( 6559):
 └RichText(renderObject: RenderParagraph relayoutBoundary=up6)
```

这是一个“扁平化”的树，显示了通过各种构建函数投影的所有widget（如果你在widget树的根中调用`toStringDeepwidget`，这是你获得的树）。 你会看到很多在你的应用源代码中没有出现的widget，因为它们是被框架中widget的`build()`函数插入的。例如，`InkFeature`是Material widget的一个实现细节 。

当按钮从被按下变为被释放时debugDumpApp()被调用，FlatButton对象同时调用`setState()`，并将自己标记为"dirty"。 这就是为什么如果你看转储，你会看到特定的对象标记为“dirty”。您还可以查看已注册了哪些手势监听器; 在这种情况下，一个单一的GestureDetector被列出，并且监听“tap”手势（“tap”是`TapGestureDetector`的`toStringShort`函数输出的）

如果您编写自己的widget，则可以通过覆盖`debugFillProperties()`来添加信息。将DiagnosticsProperty对象作为方法参数，并调用父类方法。 该函数是该`toString`方法用来填充小部件描述信息的。

渲染层

如果您尝试调试布局问题，那么Widgets层的树可能不够详细。在这种情况下，您可以通过调用`debugDumpRenderTree()`转储渲染树。 正如`debugDumpApp()`，除布局或绘制阶段外，您可以随时调用此函数。作为一般规则，从frame 回调 或事件处理器中调用它是最佳解决方案。

要调用`debugDumpRenderTree()`，您需要添加`import'package:flutter/rendering.dart';`到您的源文件。

上面这个小例子的输出结果如下所示：

```
I/flutter ( 6559): RenderView
I/flutter ( 6559):   | debug mode enabled - android
I/flutter ( 6559):   | window size: Size(1080.0, 1794.0) (in
physical pixels)
I/flutter ( 6559):   | device pixel ratio: 2.625 (physical pixels
per logical pixel)
I/flutter ( 6559):   | configuration: Size(411.4, 683.4) at 2.625x
(in logical pixels)
I/flutter ( 6559):   |
I/flutter ( 6559):   └─child: RenderCustomPaint
I/flutter ( 6559):     | creator: CustomPaint ← Banner ←
CheckedModeBanner ←
I/flutter ( 6559):     |   WidgetsApp-[GlobalObjectKey
_MaterialAppState(1009803148)] ←
I/flutter ( 6559):     |   Theme ← AnimatedTheme ←
ScrollConfiguration ← MaterialApp ←
I/flutter ( 6559):     |   [root]
I/flutter ( 6559):     | parentData: <none>
I/flutter ( 6559):     | constraints: BoxConstraints(w=411.4,
h=683.4)
I/flutter ( 6559):     | size: Size(411.4, 683.4)
I/flutter ( 6559):     |
I/flutter ( 6559):     └─child: RenderPointerListener
I/flutter ( 6559):       | creator: Listener ← Navigator-
[GlobalObjectKey<NavigatorState>
I/flutter ( 6559):       |   _WidgetsAppState(552902158)] ← Title
← LocaleQuery ← MediaQuery
I/flutter ( 6559):       |   ← DefaultTextStyle ← CustomPaint ←
Banner ← CheckedModeBanner ←
I/flutter ( 6559):       |   WidgetsApp-[GlobalObjectKey
_MaterialAppState(1009803148)] ←
I/flutter ( 6559):       |   Theme ← AnimatedTheme ← …
I/flutter ( 6559):       | parentData: <none>
I/flutter ( 6559):       | constraints: BoxConstraints(w=411.4,
h=683.4)
I/flutter ( 6559):       | size: Size(411.4, 683.4)
I/flutter ( 6559):       | behavior: defer-to-child
I/flutter ( 6559):       | listeners: down, up, cancel
I/flutter ( 6559):       |
I/flutter ( 6559):       └─child: RenderAbsorbPointer
I/flutter ( 6559):         | creator: AbsorbPointer ← Listener ←
I/flutter ( 6559):         |   Navigator-
[GlobalObjectKey<NavigatorState>
```

```
I/flutter ( 6559):        |   _WidgetsAppState(552902158)] ← Title
← LocaleQuery ← MediaQuery
I/flutter ( 6559):        |   ← DefaultTextStyle ← CustomPaint ←
Banner ← CheckedModeBanner ←
I/flutter ( 6559):        |   WidgetsApp-[GlobalObjectKey
_MaterialAppState(1009803148)] ←
I/flutter ( 6559):        |   Theme ← …
I/flutter ( 6559):        | parentData: <none>
I/flutter ( 6559):        | constraints: BoxConstraints(w=411.4,
h=683.4)
I/flutter ( 6559):        | size: Size(411.4, 683.4)
I/flutter ( 6559):        | absorbing: false
I/flutter ( 6559):        |
I/flutter ( 6559):        └──child: RenderSemanticsAnnotations
I/flutter ( 6559):          | creator: Semantics ← Focus-
[GlobalKey 489139594] ← AbsorbPointer
I/flutter ( 6559):          |   ← Listener ← Navigator-
[GlobalObjectKey<NavigatorState>
I/flutter ( 6559):          |   _WidgetsAppState(552902158)] ←
Title ← LocaleQuery ← MediaQuery
I/flutter ( 6559):          |   ← DefaultTextStyle ← CustomPaint
← Banner ← CheckedModeBanner ←
I/flutter ( 6559):          |   …
I/flutter ( 6559):          | parentData: <none>
I/flutter ( 6559):          | constraints: BoxConstraints(w=411.4,
h=683.4)
I/flutter ( 6559):          | size: Size(411.4, 683.4)
I/flutter ( 6559):          |
I/flutter ( 6559):          └──child: _RenderTheatre
I/flutter ( 6559):            | creator: _Theatre ← Overlay-
[GlobalKey 199833992] ← _FocusScope ←
I/flutter ( 6559):            |   Semantics ← Focus-[GlobalKey
489139594] ← AbsorbPointer ←
I/flutter ( 6559):            |   Listener ← Navigator-
[GlobalObjectKey<NavigatorState>
I/flutter ( 6559):            |   _WidgetsAppState(552902158)] ←
Title ← LocaleQuery ← MediaQuery
I/flutter ( 6559):            |   ← DefaultTextStyle ← …
I/flutter ( 6559):            | parentData: <none>
I/flutter ( 6559):            | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            | size: Size(411.4, 683.4)
I/flutter ( 6559):            |
I/flutter ( 6559):            ├──onstage: RenderStack
I/flutter ( 6559):            ┆ | creator: Stack ← _Theatre ←
Overlay-[GlobalKey 199833992] ←
```

```
I/flutter ( 6559):            ┆ |   _FocusScope ← Semantics ←
Focus-[GlobalKey 489139594] ←
I/flutter ( 6559):            ┆ |   AbsorbPointer ← Listener ←
I/flutter ( 6559):            ┆ |   Navigator-
[GlobalObjectKey<NavigatorState>
I/flutter ( 6559):            ┆ |   _WidgetsAppState(552902158)] ←
Title ← LocaleQuery ← MediaQuery
I/flutter ( 6559):            ┆ |   ← …
I/flutter ( 6559):            ┆ | parentData: not positioned;
offset=Offset(0.0, 0.0)
I/flutter ( 6559):            ┆ | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆ | size: Size(411.4, 683.4)
I/flutter ( 6559):            ┆ |
I/flutter ( 6559):            ┆ ├─child 1: RenderIgnorePointer
I/flutter ( 6559):            ┆ | | creator: IgnorePointer ←
_OverlayEntry-[GlobalKey 612888877] ←
I/flutter ( 6559):            ┆ | |   Stack ← _Theatre ← Overlay-
[GlobalKey 199833992] ← _FocusScope
I/flutter ( 6559):            ┆ | |   ← Semantics ← Focus-
[GlobalKey 489139594] ← AbsorbPointer ←
I/flutter ( 6559):            ┆ | |   Listener ← Navigator-
[GlobalObjectKey<NavigatorState>
I/flutter ( 6559):            ┆ | |   _WidgetsAppState(552902158)]
← Title ← …
I/flutter ( 6559):            ┆ | | parentData: not positioned;
offset=Offset(0.0, 0.0)
I/flutter ( 6559):            ┆ | | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆ | | size: Size(411.4, 683.4)
I/flutter ( 6559):            ┆ | | ignoring: false
I/flutter ( 6559):            ┆ | | ignoringSemantics: implicitly
false
I/flutter ( 6559):            ┆ | |
I/flutter ( 6559):            ┆ | └─child:
RenderSemanticsAnnotations
I/flutter ( 6559):            ┆ |   | creator: Semantics ←
ModalBarrier ← IgnorePointer ←
I/flutter ( 6559):            ┆ |   |   _OverlayEntry-[GlobalKey
612888877] ← Stack ← _Theatre ←
I/flutter ( 6559):            ┆ |   |   Overlay-[GlobalKey
199833992] ← _FocusScope ← Semantics ←
I/flutter ( 6559):            ┆ |   |   Focus-[GlobalKey
489139594] ← AbsorbPointer ← Listener ← …
I/flutter ( 6559):            ┆ |   | parentData: <none>
I/flutter ( 6559):            ┆ |   | constraints:
```

```
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ╎ │    │ size: Size(411.4, 683.4)
I/flutter ( 6559):            ╎ │    │
I/flutter ( 6559):            ╎ │    └──child:
RenderSemanticsGestureHandler
I/flutter ( 6559):            ╎ │      │ creator: _GestureSemantics
← RawGestureDetector ← GestureDetector
I/flutter ( 6559):            ╎ │      │   ← Semantics ←
ModalBarrier ← IgnorePointer ←
I/flutter ( 6559):            ╎ │      │   _OverlayEntry-[GlobalKey
612888877] ← Stack ← _Theatre ←
I/flutter ( 6559):            ╎ │      │   Overlay-[GlobalKey
199833992] ← _FocusScope ← Semantics ← …
I/flutter ( 6559):            ╎ │      │ parentData: <none>
I/flutter ( 6559):            ╎ │      │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ╎ │      │ size: Size(411.4, 683.4)
I/flutter ( 6559):            ╎ │      │
I/flutter ( 6559):            ╎ │      └──child:
RenderPointerListener
I/flutter ( 6559):            ╎ │        │ creator: Listener ←
_GestureSemantics ← RawGestureDetector ←
I/flutter ( 6559):            ╎ │        │   GestureDetector ←
Semantics ← ModalBarrier ← IgnorePointer ←
I/flutter ( 6559):            ╎ │        │   _OverlayEntry-
[GlobalKey 612888877] ← Stack ← _Theatre ←
I/flutter ( 6559):            ╎ │        │   Overlay-[GlobalKey
199833992] ← _FocusScope ← …
I/flutter ( 6559):            ╎ │        │ parentData: <none>
I/flutter ( 6559):            ╎ │        │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ╎ │        │ size: Size(411.4, 683.4)
I/flutter ( 6559):            ╎ │        │ behavior: opaque
I/flutter ( 6559):            ╎ │        │ listeners: down
I/flutter ( 6559):            ╎ │        │
I/flutter ( 6559):            ╎ │        └──child:
RenderConstrainedBox
I/flutter ( 6559):            ╎ │          creator: ConstrainedBox
← Listener ← _GestureSemantics ←
I/flutter ( 6559):            ╎ │            RawGestureDetector ←
GestureDetector ← Semantics ← ModalBarrier
I/flutter ( 6559):            ╎ │            ← IgnorePointer ←
_OverlayEntry-[GlobalKey 612888877] ← Stack ←
I/flutter ( 6559):            ╎ │            _Theatre ← Overlay-
[GlobalKey 199833992] ← …
I/flutter ( 6559):            ╎ │          parentData: <none>
```

```
I/flutter ( 6559): ┆ │           constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ┆ │           size: Size(411.4,
683.4)
I/flutter ( 6559): ┆ │           additionalConstraints:
BoxConstraints(biggest)
I/flutter ( 6559): ┆ │
I/flutter ( 6559): ┆ └──child 2:
RenderSemanticsAnnotations
I/flutter ( 6559): ┆   │ creator: Semantics ← Focus-
[GlobalObjectKey
I/flutter ( 6559): ┆   │   MaterialPageRoute<Null>
(875520219)] ← _ModalScope-[GlobalKey
I/flutter ( 6559): ┆   │   816151164] ← _OverlayEntry-
[GlobalKey 727622716] ← Stack ←
I/flutter ( 6559): ┆   │   _Theatre ← Overlay-
[GlobalKey 199833992] ← _FocusScope ←
I/flutter ( 6559): ┆   │   Semantics ← Focus-[GlobalKey
489139594] ← AbsorbPointer ←
I/flutter ( 6559): ┆   │   Listener ← …
I/flutter ( 6559): ┆   │ parentData: not positioned;
offset=Offset(0.0, 0.0)
I/flutter ( 6559): ┆   │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ┆   │ size: Size(411.4, 683.4)
I/flutter ( 6559): ┆   │
I/flutter ( 6559): ┆   └──child: RenderOffstage
I/flutter ( 6559): ┆     │ creator: Offstage ←
_FocusScope ← Semantics ←
I/flutter ( 6559): ┆     │   Focus-[GlobalObjectKey
MaterialPageRoute<Null>(875520219)] ←
I/flutter ( 6559): ┆     │   _ModalScope-[GlobalKey
816151164] ← _OverlayEntry-[GlobalKey
I/flutter ( 6559): ┆     │   727622716] ← Stack ←
_Theatre ← Overlay-[GlobalKey 199833992] ←
I/flutter ( 6559): ┆     │   _FocusScope ← Semantics ←
Focus-[GlobalKey 489139594] ← …
I/flutter ( 6559): ┆     │ parentData: <none>
I/flutter ( 6559): ┆     │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ┆     │ size: Size(411.4, 683.4)
I/flutter ( 6559): ┆     │ offstage: false
I/flutter ( 6559): ┆     │
I/flutter ( 6559): ┆     └──child: RenderIgnorePointer
I/flutter ( 6559): ┆       │ creator: IgnorePointer ←
Offstage ← _FocusScope ← Semantics ←
```

```
I/flutter ( 6559): ├─       │   Focus-[GlobalObjectKey
MaterialPageRoute<Null>(875520219)] ←
I/flutter ( 6559): ├─       │   _ModalScope-[GlobalKey
816151164] ← _OverlayEntry-[GlobalKey
I/flutter ( 6559): ├─       │   727622716] ← Stack ←
_Theatre ← Overlay-[GlobalKey 199833992] ←
I/flutter ( 6559): ├─       │   _FocusScope ← Semantics
← …
I/flutter ( 6559): ├─       │ parentData: <none>
I/flutter ( 6559): ├─       │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ├─       │ size: Size(411.4, 683.4)
I/flutter ( 6559): ├─       │ ignoring: false
I/flutter ( 6559): ├─       │ ignoringSemantics:
implicitly false
I/flutter ( 6559): ├─       │
I/flutter ( 6559): ├─       └──child:
RenderFractionalTranslation
I/flutter ( 6559): ├─         │ creator:
FractionalTranslation ← SlideTransition ←
I/flutter ( 6559): ├─         │
_MountainViewPageTransition ← IgnorePointer ← Offstage ←
I/flutter ( 6559): ├─         │   _FocusScope ←
Semantics ← Focus-[GlobalObjectKey
I/flutter ( 6559): ├─         │   MaterialPageRoute<Null>
(875520219)] ← _ModalScope-[GlobalKey
I/flutter ( 6559): ├─         │   816151164] ←
_OverlayEntry-[GlobalKey 727622716] ← Stack ←
I/flutter ( 6559): ├─         │   _Theatre ← …
I/flutter ( 6559): ├─         │ parentData: <none>
I/flutter ( 6559): ├─         │ constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ├─         │ size: Size(411.4, 683.4)
I/flutter ( 6559): ├─         │ translation: Offset(0.0,
0.0)
I/flutter ( 6559): ├─         │ transformHitTests: true
I/flutter ( 6559): ├─         │
I/flutter ( 6559): ├─         └──child:
RenderRepaintBoundary
I/flutter ( 6559): ├─           │ creator:
RepaintBoundary ← FractionalTranslation ←
I/flutter ( 6559): ├─           │   SlideTransition ←
_MountainViewPageTransition ← IgnorePointer ←
I/flutter ( 6559): ├─           │   Offstage ←
_FocusScope ← Semantics ← Focus-[GlobalObjectKey
I/flutter ( 6559): ├─           │
```

```
MaterialPageRoute<Null>(875520219)] ← _ModalScope-[GlobalKey
I/flutter ( 6559):            ┆            |   816151164] ←
_OverlayEntry-[GlobalKey 727622716] ← Stack ← …
I/flutter ( 6559):            ┆            | parentData: <none>
I/flutter ( 6559):            ┆            | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆            | size: Size(411.4,
683.4)
I/flutter ( 6559):            ┆            | metrics: 83.3% useful
(1 bad vs 5 good)
I/flutter ( 6559):            ┆            | diagnosis: this is a
useful repaint boundary and should be kept
I/flutter ( 6559):            ┆            |
I/flutter ( 6559):            ┆            └──child:
RenderDecoratedBox
I/flutter ( 6559):            ┆              | creator: DecoratedBox
← Container ← AnimatedContainer ← Material
I/flutter ( 6559):            ┆              |   ← AppHome ←
_ModalScopeStatus ← PageStorage-[GlobalKey
I/flutter ( 6559):            ┆              |   619728754] ←
RepaintBoundary ← FractionalTranslation ←
I/flutter ( 6559):            ┆              |   SlideTransition ←
_MountainViewPageTransition ← IgnorePointer ←
I/flutter ( 6559):            ┆              |   …
I/flutter ( 6559):            ┆              | parentData: <none>
I/flutter ( 6559):            ┆              | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆              | size: Size(411.4,
683.4)
I/flutter ( 6559):            ┆              | decoration:
I/flutter ( 6559):            ┆              |   <no decorations
specified>
I/flutter ( 6559):            ┆              | configuration:
ImageConfiguration(bundle:
I/flutter ( 6559):            ┆              |
PlatformAssetBundle@367106502(), devicePixelRatio: 2.625,
I/flutter ( 6559):            ┆              |   platform: android)
I/flutter ( 6559):            ┆              |
I/flutter ( 6559):            ┆              └──child:
RenderDecoratedBox
I/flutter ( 6559):            ┆                | creator:
DecoratedBox ← Container ← DecoratedBox ← Container ←
I/flutter ( 6559):            ┆                |   AnimatedContainer
← Material ← AppHome ← _ModalScopeStatus ←
I/flutter ( 6559):            ┆                |   PageStorage-
[GlobalKey 619728754] ← RepaintBoundary ←
```

```
I/flutter ( 6559):            ┆                 |
FractionalTranslation ← SlideTransition ← …
I/flutter ( 6559):            ┆                 | parentData: <none>
I/flutter ( 6559):            ┆                 | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆                 | size: Size(411.4,
683.4)
I/flutter ( 6559):            ┆                 | decoration:
I/flutter ( 6559):            ┆                 |   backgroundColor:
Color(0xfffafafa)
I/flutter ( 6559):            ┆                 | configuration:
ImageConfiguration(bundle:
I/flutter ( 6559):            ┆                 |
PlatformAssetBundle@367106502(), devicePixelRatio: 2.625,
I/flutter ( 6559):            ┆                 |   platform:
android)
I/flutter ( 6559):            ┆                 |
I/flutter ( 6559):            ┆                 └──child:
_RenderInkFeatures
I/flutter ( 6559):            ┆                   | creator:
_InkFeature-[GlobalKey ink renderer] ←
I/flutter ( 6559):            ┆                   |
NotificationListener<LayoutChangedNotification> ← DecoratedBox
I/flutter ( 6559):            ┆                   |   ← Container ←
DecoratedBox ← Container ← AnimatedContainer ←
I/flutter ( 6559):            ┆                   |   Material ←
AppHome ← _ModalScopeStatus ← PageStorage-[GlobalKey
I/flutter ( 6559):            ┆                   |   619728754] ←
RepaintBoundary ← …
I/flutter ( 6559):            ┆                   | parentData:
<none>
I/flutter ( 6559):            ┆                   | constraints:
BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559):            ┆                   | size: Size(411.4,
683.4)
I/flutter ( 6559):            ┆                   |
I/flutter ( 6559):            ┆                   └──child:
RenderPositionedBox
I/flutter ( 6559):            ┆                     | creator: Center
← DefaultTextStyle ← AnimatedDefaultTextStyle ←
I/flutter ( 6559):            ┆                     |   _InkFeature-
[GlobalKey ink renderer] ←
I/flutter ( 6559):            ┆                     |
NotificationListener<LayoutChangedNotification> ← DecoratedBox
I/flutter ( 6559):            ┆                     |   ← Container
← DecoratedBox ← Container ← AnimatedContainer ←
```

```
I/flutter ( 6559): ┆                    │   Material ← AppHome ← …
I/flutter ( 6559): ┆                    │ parentData: <none>
I/flutter ( 6559): ┆                    │ constraints: BoxConstraints(w=411.4, h=683.4)
I/flutter ( 6559): ┆                    │ size: Size(411.4, 683.4)
I/flutter ( 6559): ┆                    │ alignment: Alignment.center
I/flutter ( 6559): ┆                    │ widthFactor: expand
I/flutter ( 6559): ┆                    │ heightFactor: expand
I/flutter ( 6559): ┆                    │
I/flutter ( 6559): ┆                    └─child: RenderConstrainedBox relayoutBoundary=up1
I/flutter ( 6559): ┆                      │ creator: ConstrainedBox ← MaterialButton ← FlatButton ← Center ←
I/flutter ( 6559): ┆                      │ DefaultTextStyle ← AnimatedDefaultTextStyle ←
I/flutter ( 6559): ┆                      │ _InkFeature-[GlobalKey ink renderer] ←
I/flutter ( 6559): ┆                      │ NotificationListener<LayoutChangedNotification> ← DecoratedBox
I/flutter ( 6559): ┆                      │   ← Container ← DecoratedBox ← Container ← …
I/flutter ( 6559): ┆                      │ parentData: offset=Offset(156.7, 323.7)
I/flutter ( 6559): ┆                      │ constraints: BoxConstraints(0.0<=w<=411.4, 0.0<=h<=683.4)
I/flutter ( 6559): ┆                      │ size: Size(98.0, 36.0)
I/flutter ( 6559): ┆                      │ additionalConstraints: BoxConstraints(88.0<=w<=Infinity, h=36.0)
I/flutter ( 6559): ┆                      │
I/flutter ( 6559): ┆                      └─child: RenderSemanticsGestureHandler relayoutBoundary=up2
I/flutter ( 6559): ┆                        │ creator: _GestureSemantics ← RawGestureDetector ← GestureDetector
I/flutter ( 6559): ┆                        │   ← InkWell ← IconTheme ← DefaultTextStyle ←
I/flutter ( 6559): ┆                        │ AnimatedDefaultTextStyle ← ConstrainedBox ← MaterialButton ←
I/flutter ( 6559): ┆                        │ FlatButton ← Center ← DefaultTextStyle ← …
```

```
I/flutter ( 6559):            ┆                          │ parentData:
<none>
I/flutter ( 6559):            ┆                          │
constraints: BoxConstraints(88.0<=w<=411.4, h=36.0)
I/flutter ( 6559):            ┆                          │ size:
Size(98.0, 36.0)
I/flutter ( 6559):            ┆                          │
I/flutter ( 6559):            ┆                          └──child:
RenderPointerListener relayoutBoundary=up3
I/flutter ( 6559):            ┆                            │ creator:
Listener ← _GestureSemantics ← RawGestureDetector ←
I/flutter ( 6559):            ┆                            │
GestureDetector ← InkWell ← IconTheme ← DefaultTextStyle ←
I/flutter ( 6559):            ┆                            │
AnimatedDefaultTextStyle ← ConstrainedBox ← MaterialButton ←
I/flutter ( 6559):            ┆                            │
FlatButton ← Center ← ⋯
I/flutter ( 6559):            ┆                            │
parentData: <none>
I/flutter ( 6559):            ┆                            │
constraints: BoxConstraints(88.0<=w<=411.4, h=36.0)
I/flutter ( 6559):            ┆                            │ size:
Size(98.0, 36.0)
I/flutter ( 6559):            ┆                            │ behavior:
opaque
I/flutter ( 6559):            ┆                            │
listeners: down
I/flutter ( 6559):            ┆                            │
I/flutter ( 6559):            ┆                            └──child:
RenderPadding relayoutBoundary=up4
I/flutter ( 6559):            ┆                              │
creator: Padding ← Container ← Listener ← _GestureSemantics ←
I/flutter ( 6559):            ┆                              │
RawGestureDetector ← GestureDetector ← InkWell ← IconTheme ←
I/flutter ( 6559):            ┆                              │
DefaultTextStyle ← AnimatedDefaultTextStyle ← ConstrainedBox ←
I/flutter ( 6559):            ┆                              │
MaterialButton ← ⋯
I/flutter ( 6559):            ┆                              │
parentData: <none>
I/flutter ( 6559):            ┆                              │
constraints: BoxConstraints(88.0<=w<=411.4, h=36.0)
I/flutter ( 6559):            ┆                              │ size:
Size(98.0, 36.0)
I/flutter ( 6559):            ┆                              │
padding: EdgeInsets(16.0, 0.0, 16.0, 0.0)
```

```
I/flutter ( 6559):            ├                                │
I/flutter ( 6559):            ├                                └──child:
RenderPositionedBox relayoutBoundary=up5
I/flutter ( 6559):            ├                                  │
creator: Center ← Padding ← Container ← Listener ←
I/flutter ( 6559):            ├                                  │
_GestureSemantics ← RawGestureDetector ← GestureDetector ←
I/flutter ( 6559):            ├                                  │
InkWell ← IconTheme ← DefaultTextStyle ←
I/flutter ( 6559):            ├                                  │
AnimatedDefaultTextStyle ← ConstrainedBox ← …
I/flutter ( 6559):            ├                                  │
parentData: offset=Offset(16.0, 0.0)
I/flutter ( 6559):            ├                                  │
constraints: BoxConstraints(56.0<=w<=379.4, h=36.0)
I/flutter ( 6559):            ├                                  │ size:
Size(66.0, 36.0)
I/flutter ( 6559):            ├                                  │
alignment: Alignment.center
I/flutter ( 6559):            ├                                  │
widthFactor: 1.0
I/flutter ( 6559):            ├                                  │
heightFactor: expand
I/flutter ( 6559):            ├                                  │
I/flutter ( 6559):            ├
└──child: RenderParagraph relayoutBoundary=up6
I/flutter ( 6559):            ├                                    │
creator: RichText ← Text ← Center ← Padding ← Container ←
I/flutter ( 6559):            ├                                    │
Listener ← _GestureSemantics ← RawGestureDetector ←
I/flutter ( 6559):            ├                                    │
GestureDetector ← InkWell ← IconTheme ← DefaultTextStyle ← …
I/flutter ( 6559):            ├                                    │
parentData: offset=Offset(0.0, 10.0)
I/flutter ( 6559):            ├                                    │
constraints: BoxConstraints(0.0<=w<=379.4, 0.0<=h<=36.0)
I/flutter ( 6559):            ├                                    │
size: Size(66.0, 16.0)
I/flutter ( 6559):            ├
└─┬──────── text ═══
I/flutter ( 6559):            ├                                      ╟
TextSpan:
I/flutter ( 6559):            ├                                      ╟
inherit: false
I/flutter ( 6559):            ├                                      ╟
color: Color(0xdd000000)
```

```
I/flutter ( 6559):            ┆                                   ┆
family: "Roboto"
I/flutter ( 6559):            ┆                                   ┆
size: 14.0
I/flutter ( 6559):            ┆                                   ┆
weight: 500
I/flutter ( 6559):            ┆                                   ┆
baseline: alphabetic
I/flutter ( 6559):            ┆                                   ┆
"Dump App"
I/flutter ( 6559):            ┆
 ╚═══════════════════
I/flutter ( 6559):            ┆
I/flutter ( 6559):             └--no offstage children
```

这是根`RenderObject`对象的`toStringDeep`函数的输出。

当调试布局问题时，关键要看的是`size`和`constraints`字段。约束沿着树向下传递，尺寸向上传递。

例如，在上面的转储中，您可以看到窗口大小，Size(411.4, 683.4)，它用于强制`RenderPositionedBox`下的所有渲染框到屏幕的大小， 约束条件为BoxConstraints(w=411.4, h=683.4)。从`RenderPositionedBox`的转储中看到是由Center widget创建的（如creator字段所描述的）， 设置其孩子的约束为：BoxConstraints(0.0<=w<=411.4,0.0<=h<=683.4)。一个子widget RenderPadding进一步插入这些约束以添加填充空间，padding值为EdgeInsets(16.0, 0.0, 16.0, 0.0)，因此RenderConstrainedBox具有约束BoxConstraints(0.0<=w<=395.4, 0.0<=h<=667.4)。该creator字段告诉我们的这个对象可能是其`FlatButton`定义的一部分，它在其内容上设置最小宽度为88像素，并且设置高度为36.0像素（这是Material Design设计规范中`FlatButton`类的尺寸标准）。

最内部RenderPositionedBox再次松开约束，这次是将按钮中的文本居中。在`RenderParagraph`中基于它的内容来决定其大小。 如果您现在按照size链继续往下查看，您会看到文本的大小是如何影响其按钮的框的宽度的，它们都是根据孩子的尺寸自行调整大小。

另一种需要注意的是每个盒子描述的”relayoutSubtreeRoot”部分，它告诉你有多少祖先以某种方式依赖于这个元素的大小。 因此，`RenderParagraph`有一个`relayoutSubtreeRoot=up8`，这意味着当它`RenderParagraph`被标及为”dirty”时，它的八个祖先也必须被标记为”dirty”，因为它们可能受到新尺寸的影响。

如果您编写自己的渲染对象，则可以通过覆盖`debugFillProperties()`将信息添加到转储。 将DiagnosticsProperty对象作为方法的参数，并调用父类方法。

层

如果您尝试调试合成问题，则可以使用`debugDumpLayerTree()`。对于上面的例子，它会输出：

```
I/flutter :  TransformLayer
I/flutter :  │ creator: [root]
I/flutter :  │ offset: Offset(0.0, 0.0)
I/flutter :  │ transform:
I/flutter :  │   [0] 3.5,0.0,0.0,0.0
I/flutter :  │   [1] 0.0,3.5,0.0,0.0
I/flutter :  │   [2] 0.0,0.0,1.0,0.0
I/flutter :  │   [3] 0.0,0.0,0.0,1.0
I/flutter :  │
I/flutter :  ├─child 1: OffsetLayer
I/flutter :  │ │ creator: RepaintBoundary ← _FocusScope ←
Semantics ← Focus-[GlobalObjectKey MaterialPageRoute(560156430)] ←
_ModalScope-[GlobalKey 328026813] ← _OverlayEntry-[GlobalKey
388965355] ← Stack ← Overlay-[GlobalKey 625702218] ← Navigator-
[GlobalObjectKey _MaterialAppState(859106034)] ← Title ← ⋯
I/flutter :  │ │ offset: Offset(0.0, 0.0)
I/flutter :  │ │
I/flutter :  │ └─child 1: PictureLayer
I/flutter :  │
I/flutter :  └─child 2: PictureLayer
```

这是根`Layer`的`toStringDeep`输出的。

根部的变换是应用设备像素比的变换; 在这种情况下，每个逻辑像素代表3.5个设备像素。

`RepaintBoundary` widget在渲染树的层中创建了一个`RenderRepaintBoundary`。这用于减少需要重绘的需求量。

语义

您还可以调用`debugDumpSemanticsTree()`获取语义树（呈现给系统可访问性API的树）的转储。 要使用此功能，必须首先启用辅助功能，例如启用系统辅助工具或`SemanticsDebugger` （下面讨论）。

对于上面的例子，它会输出:

```
I/flutter : SemanticsNode(0; Rect.fromLTRB(0.0, 0.0, 411.4, 683.4))
I/flutter :  ├SemanticsNode(1; Rect.fromLTRB(0.0, 0.0, 411.4,
683.4))
I/flutter :  │ └SemanticsNode(2; Rect.fromLTRB(0.0, 0.0, 411.4,
683.4); canBeTapped)
I/flutter :  └SemanticsNode(3; Rect.fromLTRB(0.0, 0.0, 411.4,
683.4))
I/flutter :   └SemanticsNode(4; Rect.fromLTRB(0.0, 0.0, 82.0,
36.0); canBeTapped; "Dump App")
```

调度

要找出相对于帧的开始/结束事件发生的位置，可以切换`debugPrintBeginFrameBanner`和`debugPrintEndFrameBanner`布尔值以将帧的开始和结束打印到控制台。

例如:

```
I/flutter : ▄▄▄▄▄▄▄▄▄▄▄▄▄ Frame 12         30s 437.086ms
▄▄▄▄▄▄▄▄▄▄▄▄▄
I/flutter : Debug print: Am I performing this work more than once
per frame?
I/flutter : Debug print: Am I performing this work more than once
per frame?
I/flutter :
▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄
```

`debugPrintScheduleFrameStacks`还可以用来打印导致当前帧被调度的调用堆栈。

可视化调试

您也可以通过设置`debugPaintSizeEnabled`为`true`以可视方式调试布局问题。 这是来自`rendering`库的布尔值。它可以在任何时候启用，并在为true时影响绘制。设置它的最简单方法是在`void main()`的顶部设置。

当它被启用时，所有的盒子都会得到一个明亮的深青色边框，padding（来自widget如Padding）显示为浅蓝色，子widget周围有一个深蓝色框， 对齐方式（来自widget如Center和Align）显示为黄色箭头. 空白（如没有任何子节点的Container）以灰色显示。

`debugPaintBaselinesEnabled`做了类似的事情，但对于具有基线的对象，文字基线以绿色显示，表意(ideographic)基线以橙色显示。

`debugPaintPointersEnabled`标志打开一个特殊模式，任何正在点击的对象都会以深青色突出显示。 这可以帮助您确定某个对象是否以某种不正确地方式进行hit测试（Flutter检测点击的位置是否有能响应用户操作的widget）,例如，如果它实际上超出了其父项的范围，首先不会考虑通过hit测试。

如果您尝试调试合成图层，例如以确定是否以及在何处添加`RepaintBoundary` widget，则可以使用`debugPaintLayerBordersEnabled`标志， 该标志用橙色或轮廓线标出每个层的边界，或者使用`debugRepaintRainbowEnabled`标志， 只要他们重绘时，这会使该层被一组旋转色所覆盖。

所有这些标志只能在调试模式下工作。通常，Flutter框架中以“`debug...`”开头的任何内容都只能在调试模式下工作。

调试动画

调试动画最简单的方法是减慢它们的速度。为此，请将`timeDilation`变量（在scheduler库中）设置为大于1.0的数字，例如50.0。 最好在应用程序启动时只设置一次。如果您在运行中更改它，尤其是在动画运行时将其值减小，则框架的观察时可能会倒退，这可能会导致断言并且通常会干扰您的工作。

调试性能问题

要了解您的应用程序导致重新布局或重新绘制的原因，您可以分别设置`debugPrintMarkNeedsLayoutStacks`和`debugPrintMarkNeedsPaintStacks`标志。 每当渲染盒被要求重新布局和重新绘制时，这些都会将堆栈跟踪记录到控制台。如果这种方法对您有用，您可以使用`services`库中的`debugPrintStack()`方法按需打印堆栈痕迹。

衡量应用启动时间

要收集有关Flutter应用程序启动所需时间的详细信息，可以在运行`flutter run`时使用`trace-startup`和`profile`选项。

```
$ flutter run --trace-startup --profile
```

跟踪输出保存为`start_up_info.json`，在Flutter工程目录在build目录下。输出列出了从应用程序启动到这些跟踪事件（以微秒捕获）所用的时间：

- 进入Flutter引擎时.
- 展示应用第一帧时.
- 初始化Flutter框架时.
- 完成Flutter框架初始化时.

如：

```
{
  "engineEnterTimestampMicros": 96025565262,
  "timeToFirstFrameMicros": 2171978,
  "timeToFrameworkInitMicros": 514585,
  "timeAfterFrameworkInitMicros": 1657393
}
```

跟踪Dart代码性能

要执行自定义性能跟踪和测量Dart任意代码段的wall/CPU时间（类似于在Android上使用systrace）。 使用`dart:developer`的Timeline工具来包含你想测试的代码块，例如：

```
Timeline.startSync('interesting function');
// iWonderHowLongThisTakes();
Timeline.finishSync();
```

然后打开你应用程序的Observatory timeline页面，在”Recorded Streams”中选择’Dart’复选框，并执行你想测量的功能。

刷新页面将在Chrome的跟踪工具中显示应用按时间顺序排列的timeline记录。

请确保运行 `flutter run` 时带有 `--profile` 标志，以确保运行时性能特征与您的最终产品差异最小。

Performance Overlay

要获得应用程序性能图，请将 `MaterialApp` 构造函数的 `showPerformanceOverlay` 参数设置为true。 `WidgetsApp` 构造函数也有类似的参数（如果你没有使用 `MaterialApp` 或者 `WidgetsApp`，你可以通过将你的应用程序包装在一个stack中， 并将一个widget放在通过 `new PerformanceOverlay.allEnabled()` 创建的stack上来获得相同的效果）。

这将显示两个图表。第一个是GPU线程花费的时间，最后一个是CPU线程花费的时间。 图中的白线以16ms增量沿纵轴显示; 如果图中超过这三条线之一，那么您的运行频率低于60Hz。横轴代表帧。 该图仅在应用程序绘制时更新，因此如果它处于空闲状态，该图将停止移动。

这应该始终在发布模式（release mode）下测试，因为在调试模式下，故意牺牲性能来换取有助于开发调试的功能，如assert声明，这些都是非常耗时的，因此结果将会产生误导。

Material grid

在开发实现Material Design的应用程序时， 将Material Design基线网格覆盖在应用程序上可能有助于验证对齐。 为此， `MaterialApp` 构造函数 有一个 `debugShowMaterialGrid` 参数， 当在调试模式设置为true时，它将覆盖这样一个网格。

您也可以直接使用 `GridPaper` widget将这种网格覆盖在非Material应用程序上 。