

flutter 学习



1. 课前准备

1. 环境搭建 [flutter中文网](#)
2. 开发工具 前端开发软件: [Visual Studio Code](#) 移动端开发软件: [Xcode](#)、[Android Studio](#)

2. 课堂目标

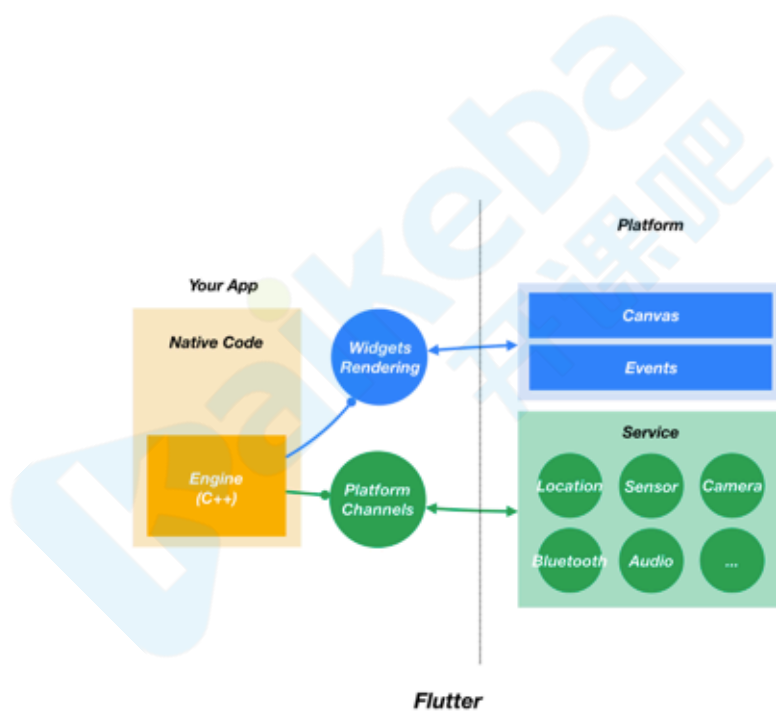
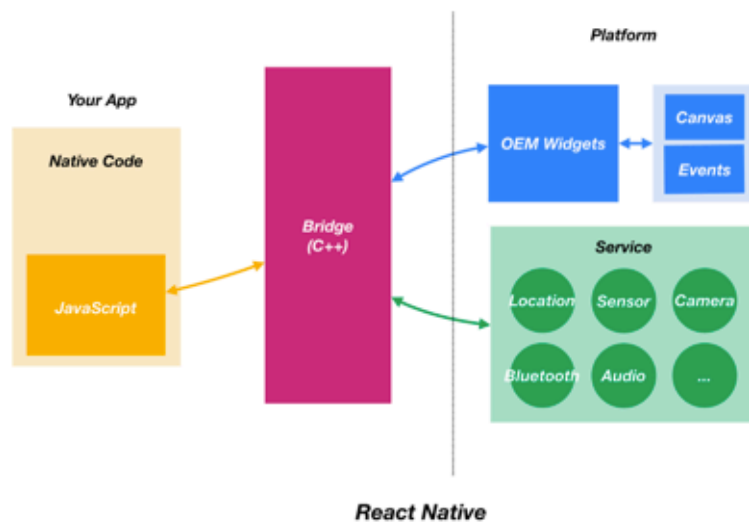
熟练flutter相关语法及基础控件的使用

3. 知识点

1. flutter简介

- Flutter是Google在2015年推出的移动UI框架，可快速在iOS和Android上构建高质量的用户界面
- Flutter使用语言是Dart（Google于2011年推出的新的计算机编程语言）
- 精美 (Beautiful)、极速 (Fast)、高效 (Productive)、开放 (Open)
- 实现真正意义的一套代码两端运行

2. **flutter原理** 与React Native不一样，flutter的界面显示只是依赖native侧（iOS or Andriod）的一个view即可，页面内容渲染交由Flutter层自身来完成，所以其相对React Native等框架性能更好。其显示原理如下：



从上图可以看出，RN 其实是通过 Bridge，把 js 代码传递到 native，最终是还转换成原生的 View 进行绘制。这就会导致我们经常会出现同一套 UI 在 iOS 和 Android 平台上存在各种不同的兼容性问题。而 Flutter 就完全不一样，它采用的是 Google 的 Skia 引擎，大家可以把它理解成一套全新的跨平台 UI 框架。

开发方式	性能	热更新	技术成熟度
Naitve	好	不支持	成熟
React Native	支持	中等文本	较成熟
Flutter	较好	iOS不支持	不成熟

3. 环境搭建

Mac安装方式[参考文章](#):

- 硬件环境: MacOS (64-bit) 、磁盘空间700MB (不包括Xcode或Android Studio的磁盘空间) 、
- Step1: 使用镜像

```
export PUB_HOSTED_URL=https://pub.flutter-io.cn
export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
```

- Step2: 配置Flutter SDK
 - 前往官网获取SDK, [这是传送门](#)
 - 解压安装包到想安装的目录(这个目录以后不需要动, 故不建议放在deskTop)

```
cd ~/development
unzip ~/Downloads/flutter_macos_v0.5.1-beta.zip
```

- 添加 flutter 相关工具到path中

```
export PATH=`pwd`/flutter/bin:$PATH
```

- 运行 flutter doctor查看安装结果如何 (注: 如果是想在iOS上运行, 则不需要安卓相关也是可以的, 反之一样)

Window安装方式请移步查看教学视频

4. 构建项目

- VS需要添加flutter和dart code两个插件, 安装完成重启编译器即可
- 创建flutter项目

- 方式一: 命令行

```
flutter create flutterProject
```

- 方式二: Android Studio

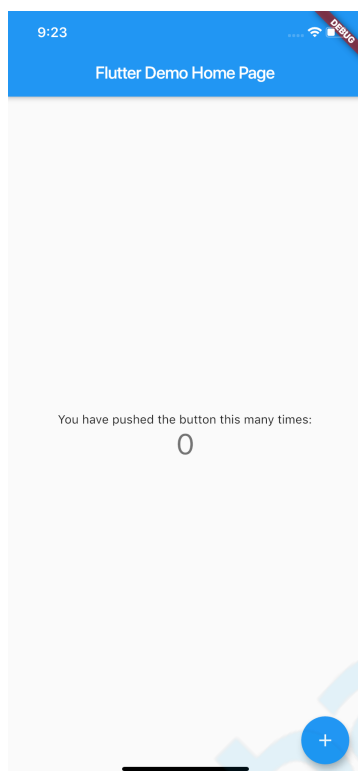
File>New Flutter Project

- 方式三: VSCode

①**View>Command Palette...**

②输入 flutter, 然后选择 **Flutter: New Project**

- 如果正常，运行效果如下：



5. 组件讲解

stateful和**stateless**：实现Flutter app时，我们用widgets来构建app的UI。这些widgets有两种类型——stateful（有状态）和 stateless（无状态）

- stateless：当创建的widget不需要管理任何形式的内部state时，则使用StatelessWidgets。eg: Text

```
void main() => runApp(MyStatelessWidget(text: "StatelessWidget
Example"));

class MyStatelessWidget extends StatelessWidget {
  final String text;
  MyStatelessWidget({Key key, this.text}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(
        text,
        textDirection: TextDirection.ltr,
      ),
    );
  }
}
```

- stateful：当创建一个能随时间动态改变的widget，并且不依赖于其初始化状态。eg: Image

```

class FavoriteWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() => new _FavoriteWidgetState();
}

class _FavoriteWidgetState extends State {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return null;
  }
}

```

注意：

- ① 创建一个Stateful Widget需要两个类，分别继承自StateFulWidget和State；
- ② state对象包含了widget的state和widget的build()方法；
- ③ 当widget的state改变了的时候，state对象会调用setState()方法，告诉框架去重绘widget；

Container:flutter开发中使用频率最高，它组合的widget，内部有绘制widget、定位widget、尺寸widget。

示例代码：

```

class Container_Property2 extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Row(
        children: <Widget>[
          Container(
            margin: const EdgeInsets.all(20),
            color: Colors.red,
            width: 50,
            height: 50,
          ),
          Container(
            margin: const EdgeInsets.all(30),
            color: Colors.blue,
            child: Column(
              children: <Widget>[
                Text( '显示1' ),
                Text( '显示2' ),
                Text( '显示3' ),
              ]
            ),
          ),
        ],
      ),
    );
  }
}

```

```

        Container(
            margin: const EdgeInsets.all(20),
            color: Colors.yellow,
            width: 50,
            height: 50,
        ),
    ],
),
);
}
}

```

- color:

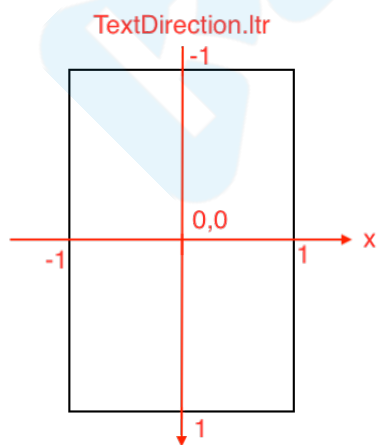
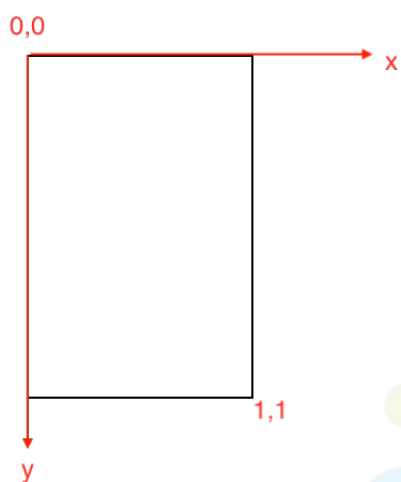
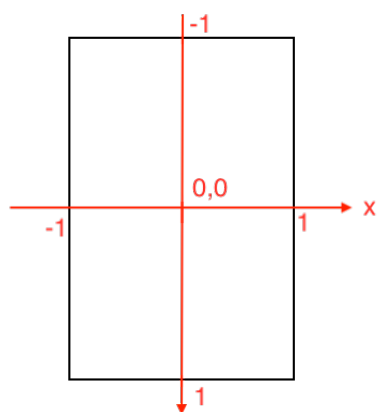
Colors.red:取得正常颜色 **Colors.red[300]**:代码.3透明度的颜色，等同于**Colors.red.shade300**
Color(0X11111111)、**Color.fromARGB(a, r, g, b)**、**Color.fromRGBO(r, g, b, opacity)**

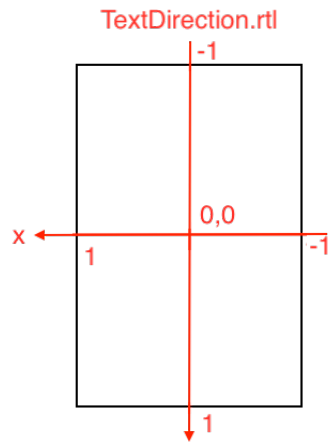
- child:子组件（Container里面没有子控件的时候它会填充整个屏幕，如果有子控件，Container就会自动适应子控件大小，并且一个Container只能容纳一个子控件，如果想容纳多个，需要设置Row、Column、Stack）

方式	排列方式
Row	水平方向排列子控件
Column	垂直方向排列子控件
Stack	堆叠的方式排列子控件
IndexStack	堆叠的方式排列子控件,通过index控制器显示哪个子控件
warp	可以让子控件自动换行的控件

- Alignment：对Container内部的子控件布局（Alignment、FractionalOffset、AlignmentDirectional） 注：x轴从左往右递增、y轴从上往下递增

方式	排列方式
Alignment	x、y轴 (-1, 1)
FractionalOffset	x、y轴 (0, 1)
AlignmentDirectional	与TextDirection有关





- constraints: 布局属性, 主要讲的是怎么确定控件的大小, 其中经常使用的就是BoxConstraint

```
class Container_Property4 extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return Center(  
      child: Container(  
        color: Colors.red,  
        alignment: Alignment.center,  
        child: Container(  
          color: Colors.blue,  
          child: Text('This is text'),  
          constraints: BoxConstraints(  
            maxHeight: 200,  
            maxWidth: 150,  
          ),  
        ),  
      ),  
    );  
  }  
}
```

注: BoxConstraints.expand()的作用是让子控件完全平铺显示在父控件上

- Margin: 其使用和RN类似, 其使用在相邻的控件之间, 主要用EdgeInsets

方式	排列方式
EdgeInsets.all()	设置上下左右的间距, 距离相同
EdgeInsets.symmetric()	设置垂直和水平方向上的距离
EdgeInsets.fromLTRB()	设置left, top, right,bottom边距
EdgeInsets.only()	用于设置哪些是非零的, 不设置默认是零

- padding: 用于设置主控件与内部空间之间的间距, 和Margin一样主要用EdgeInsets
- decoration: 其主要用于设置边框、圆角、阴影等, 主要用BoxDecoration (注: 如果设置了decoration, 则color不可以使用)

```
class Container_Property5 extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        decoration: BoxDecoration(
          color: Colors.purple,
          // 设置阴影 要在裁剪之外添加一个Containter里面处理, 否则无效
          boxShadow: [
            BoxShadow(
              color: Colors.red,
              offset: Offset(10, 10),
              blurRadius: 5),
          ],
          // 设置局部圆角
          borderRadius: BorderRadius.all(Radius.circular(20)),
          //渐变色
          gradient: LinearGradient(
            colors: [Colors.red, Colors.cyan],
          )
        ),
      child: FlutterLogo(
        size: 200.0,
      ),
    ),
  );
}
```

- Transform:设置控件显示的变换

```
class Container_Property6 extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        child: FlutterLogo(
          size: 200.0,
        ),
        transform: Matrix4.rotationZ(0.5),
      ),
    );
  }
}
```

Image:显示图片的widget。

- Image.asset: 获取本地图片, 注意yml文件的设置 ([点这里](#))
- Image.file: 加载一个本地图片, 比如相册中的图片
- Image.network: 获取网络图片

注: 如果需要设置placeholder, 则使用FadeInImage

```
return Container(  
  color: Colors.blue,  
  child: Column(  
    children: <Widget>[  
      new Image.asset('images/author.png'),  
      // new Image.file(file)  
      new Image.network(  
        imageUrl,  
        width: 150,  
        height: 150,  
      ),  
      new FadeInImage.assetNetwork(  
        placeholder: 'images/author.png',  
        image: imageUrl,  
        width: 120,  
        fit: BoxFit.fitWidth,  
      )  
    ],  
  ),  
);
```

ListView: 滚动列表控件

- 常用属性
 - scrollDirection: 滚动方向, `horizontal` 和 `vertical`, 默认是垂直方向上滚动;
 - physics: 列表滚动至边缘后继续拖动的物理效果, 安卓是一个波纹状 (对应 `ClampingScrollPhysics`), iOS是有一个回弹的弹性效果 (对应 `BouncingScrollPhysics`), 如果想不同的平台上呈现各自的效果可以使用 `AlwaysScrollableScrollPhysics`, 它会根据不同平台自动选用各自的物理效果。如果你想禁用边缘的拖动效果, 可以使用 `NeverScrollableScrollPhysics`;
 - shrinkWrap: 该属性将决定列表的长度是否仅包裹其内容的长度。当 `ListView` 嵌在一个无限长的容器组件中时, `shrinkWrap` 必须为true, 否则会给出警告;
 - padding: 内间距
 - itemExtent: 子元素长度。当列表中的每一项长度是固定的情况下可以指定该值, 有助于提高列表的性能 (因为它可以帮助 `ListView` 在未实际渲染子元素之前就计算出每一项元素的位置)
 - cacheExtent: 预渲染区域长度, `ListView` 会在其可视区域的两边留一个 `cacheExtent` 长

度的区域作为预渲染区域（对于 `ListView.build` 或 `ListView.separated` 构造函数创建的列表，不在可视区域和预渲染区域内的子元素不会被创建或会被销毁）

- children: 容纳子元素的组件数组
- 创建listView的方式
 - 方式一: `ListView()` (特点: 代码简洁, 对于小批量固定数据可以考虑使用, 但是数据量大则性能不好, 因为这种方式创建类似于RN中的scrollView, 即使还没有出现在屏幕中但仍然会被ListView所创建, 这将是一项较大的开销, 使用不当可能引起性能问题甚至卡顿)
 - 方式二: `ListView.build()`: 绝大多数列表类的需求都可以用 `ListView.build` 构造函数来解决问题
 - `itemCount`: 列表中元素的数量
 - `itemBuilder`: 子元素的渲染方法, 允许自定义子元素组件 (等同于 `rn` 中 `FlatList` 组件的 `renderItem` 属性)
 - 方式三: `ListView.separated()`: 列表子项之间需要 分割线, 此时可以考虑用此方法
 - `separatorBuilder`: 构造分割线

网络加载:

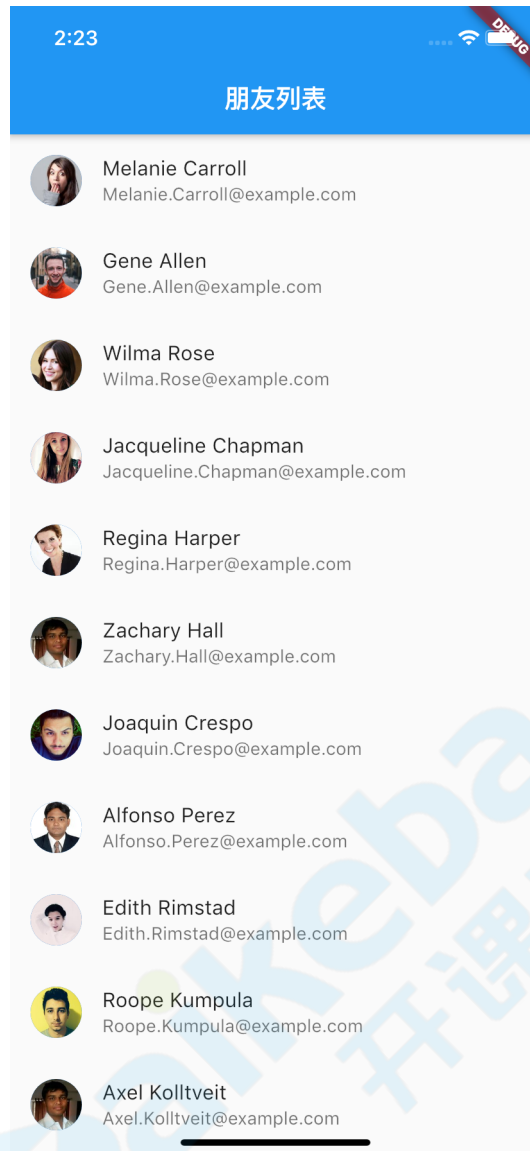
http支持位于 `dart:io`, 所以要创建一个HTTP client, 需要添加一个导入:

```
import 'dart:io';
var httpClient = new HttpClient();

var url = 'https://httpbin.org/ip';
String result;
var request = await httpClient.getUrl(Uri.parse(url));
var response = await request.close();
if (response.statusCode == HttpStatus.OK) {
  var json = await response.transform(utf8.decoder).join();
  var data = jsonDecode(json);
  result = data['origin'];
} else {
  result =
    'Error getting IP address:\nHttp status ${response.statusCode}';
}
```

作业:

- (1) 自学其余三个常用控件: `button`、输入框、`Text`
- (2) 完成如下效果 (接口: <https://randomuser.me/api/?results=30>)



4. 总结

虽然flutter和RN不属于同一种开发语言并且原理也大相径庭，但是flutter为了让开发者能够加载快速的入手，好多开发的思想甚至在api上和RN都有一定的相似度，不妨建议大家对比学习。

flutter中的控件很多，没必要每个都学，用到现看即可。

5. 作业 && 答疑

使用今天学习的内容完成列表页面

6. 下节课内容

- MaterialApp、Scaffold学习
- 项目开发