flutter 学习一

1. 课前准备

1. 查阅网站 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

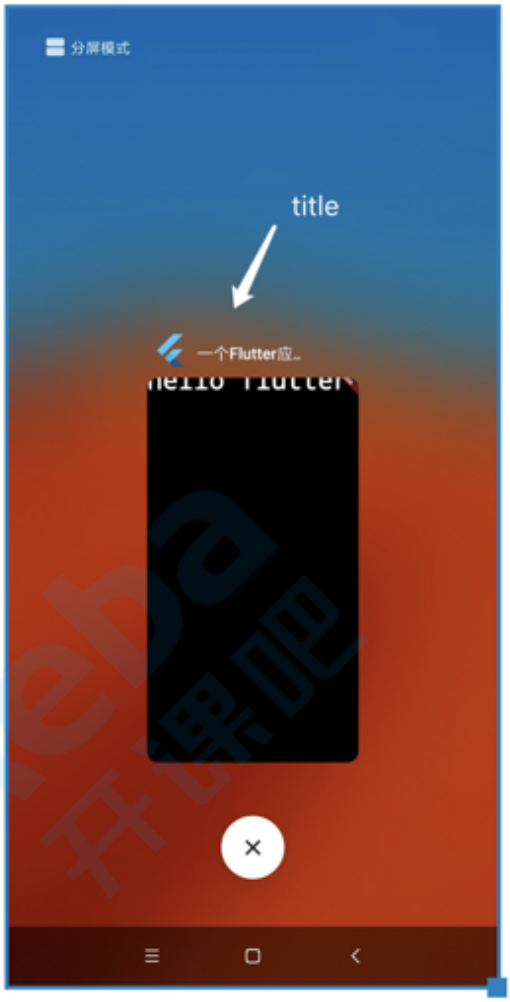
2. 课堂目标

- App整体结构
  - MaterialApp应用组件
  - Scaffold脚手架组件
  - Flutter主题
- 常用组件的使用
  - Image
  - Text
  - 按钮相关组件（MaterialButton、RaiseButton、FlatButton、DropdownButton、悬浮按钮、IconButton）
  - 单组件容器布局组件（Container、Padding、Center、Align）
  - 多组件容器布局组件（Row、Column、Stack、Wrap）

3. 知识点

App结构和导航组件

1. **MaterialApp**：封装了应用程序实现Material Design所需要的一些Widget，实际是一种设计风格，里面会有已有的一些组件（eg：theme）
    - title: 该属性会在`Android`应用管理器的App上方显示，对于`ios`设备是没有效果的



    - home: `Widget`类型，这是在应用程序正常启动时首先显示的Widget，除非指定了`initialRoute`。如果`initialRoute`显示失败，也该显示该Widget。
    - theme:`ThemeData`类型，定义应用所使用的主题颜色，可以指定一个主题中每个控件的颜色
    - routes: `Map<String, WidgetBuilder>`类型，是应用的顶级路由表，当使用`Navigator.pushNamed`进行命名路由的跳转时，会在此路表中进行查找并跳转
    - initialRoute:`String`类型，初始化路由
    - onGenerateRoute:`RouteFactory`类型，路由回调函数。当通过Navigator.of(context).pushNamed跳转的时候，如果routes查找不到会调用这个方法

2. **Scaffold**： 实现了基本的 Material Design 布局结构

- appBar：显示在界面顶部的一个 AppBar
- body：当前界面所显示的主要内容 Widget
    - drawer：抽屉菜单控件
    - bottomNavigationBar：显示在页面底部的导航栏，`items`必须大于2个

```
class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(
          title: Text('首页'),
          // centerTitle: false,
        ),
        drawer: Drawer(
          child: Column(
            children: <Widget>[
              DrawerItem(1, '选项1'),
              DrawerItem(2, '选项2'),
              DrawerItem(3, '选项3'),
              DrawerItem(4, '选项4'),
              DrawerItem(5, '选项5')
            ],
          ),
        ),
        bottomNavigationBar: BottomNavigationBar(
          type: BottomNavigationBarType.fixed,
          currentIndex: 1,
          items: [
            new BottomNavigationBarItem(
                  icon: Icon(Icons.account_balance), title: Text('标题一')),
            new BottomNavigationBarItem(
                icon: Icon(Icons.contacts), title: Text('标题二')),
            new BottomNavigationBarItem(
                icon: Icon(Icons.library_music), title: Text('标题三'))
          ],
        ),
        body: Center(
          child: Text('THIS IS BODY'),
        ),
      ),
    );
  }
}
```

3. **Flutter主题**： 使用主题可以在App里面共享颜色和字体样式。在Flutter里面有两种方式来使用主题，一种是全局范围的、一种是使用`Theme` Widget， `Theme` Widget可以在App的某个部分使用主题。全局的主题其实也就是`MaterialApp`将 `Theme`做为根widget了。

```
ThemeData({
  Brightness brightness, //深色还是浅色
  MaterialColor primarySwatch, //主题颜色样本
  Color primaryColor, //主色，决定导航栏颜色
  Color accentColor, //次级色，决定大多数Widget的颜色，如进度条、开关等。
  Color cardColor, //卡片颜色
  Color dividerColor, //分割线颜色
  ButtonThemeData buttonTheme, //按钮主题
  Color cursorColor, //输入框光标颜色
  Color dialogBackgroundColor,//对话框背景颜色
  String fontFamily, //文字字体
  TextTheme textTheme,// 字体主题，包括标题、body等文字样式
  IconThemeData iconTheme, // Icon的默认样式
  TargetPlatform platform, //指定平台，应用特定平台控件风格
  ...
})
```

- 创建全局主题：`MaterialApp` 接收一个theme的参数，类型为 `ThemeData`，为App提供统一的颜色和字体。支持的参数可以在这里查看

```
new MaterialApp(
    title: title,
    theme: new ThemeData(
        brightness: Brightness.dark,
        primaryColor: Colors.lightBlue[800],
    ),
);
```

- 创建局部主题：如果想为某个页面使用不同于App的风格，可以使用 `Theme` 来覆盖App的主题

```
new Theme(
      data: new ThemeData(
          accentColor: Colors.yellow,
      ),
      child: new Text('Hello World'),
  );
```

- 覆盖（扩展）主题：如果不想覆盖所有的样式，可以继承App的主题，只覆盖部分样式，使用 `copyWith` 方法。

```
new Theme(
    data: Theme.of(context).copyWith(accentColor: Colors.yellow),
    child: new Text('use copyWith method'),
);
```

状态

stateful和stateless：实现Flutter app时，我们用widgets来构建app的UI。这些widgets有大体有两种类型——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StatelessWidget。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；

常用组件

图片组件Image

- new Image：从ImageProvider获取图片
- new Image.asset：加载资源图片
- new Image.file：加载本地图片文件
- new Image.network：加载网络图片
- new Image.memory：加载Uint8List资源图片

```
// 资源图片
new Image.asset('imgs/logoName.jpeg'),
//网络图片
new Image.network('https://flutter.io/images/homepage/header-
illustration.png'),
// 本地文件图片
new Image.file(new File("/Users/source/Downloads/imageName.jpeg")),
// Uint8List图片
new Image.memory(bytes),
//使用ImageProvider加载图片
new Image(image: new
NetworkImage("https://flutter.io/images/homepage/screenshot-2.png"))
```

> 如果需要在加载网络图片的时候显示一个占位图或者图片加载出错时显示某张特定的图片，此时可以使用FadeInImage

```
//加载一个本地的占位图
new FadeInImage.assetNetwork(
placeholder: 'images/logo.png',
image: imageUrl,
width: 120,
fit: BoxFit.fitWidth,
)

//加载一个透明的占位图，但这个组件是不可以设置加载出错显示的图片
//需要引入transparent_image: ^0.1.0
new FadeInImage.memoryNetwork(
placeholder: kTransparentImage,
image: imageUrl,
width: 120,
fit: BoxFit.fitWidth,
)

//可以设置holder和error显示，CachedNetworkImage三方需要在package配置
new CachedNetworkImage(
width: 120,
fit: BoxFit.fitWidth,
placeholder: new CircularProgressIndicator(),
imageUrl: imageUrl,
errorWidget: new Icon(Icons.error),
```

```
)
```

Image组件属性及描述

API名称	描述
width & height	用来指定显示图片区域的宽高（并非图片的宽高）
fit	设置图片填充，类似于Android中的ScaleType
alignment	用来控制图片摆放的位置
repeat	用来设置图片重复显示（repeat-x水平重复，repeat-y垂直重复，repeat两个方向都重复，no-repeat默认情况不重复）
centerSlice	设置图片内部拉伸，相当于在图片内部设置了一个.9图，但是需要注意的是，要在显示图片的大小大于原图的情况下才可以使用这个属性，要不然会报错
matchTextDirection	需要配合Directionality进行使用
gaplessPlayback	当ImageProvider发生变化的时候，重新加载图片的过程中国，原图片的展示是否保留，true：保留，false不保留，直接空白等待下一张图片加载

BoxFit介绍

取值	描述
BoxFit.fill	全图显示，显示可能拉伸，充满
BoxFit.contain	全图显示，显示原比例，不需要充满
BoxFit.cover	显示可能拉伸，可能裁剪，充满
BoxFit.fitWidth	显示可能拉伸，可能裁剪，宽度充满
BoxFit.fitHeight	显示可能拉伸，可能裁剪，高度充满
BoxFit.none	原始大小显示
BoxFit.scaleDown	效果和contain差不多， 但是只能缩小图片，不能放大图片

> 如果需要显示圆角图片可以使用CircleAvatar或者通过Container包裹

```
new CircleAvatar(
 backgroundImage: new NetworkImage(url),
 radius: 100.0,       // --> 半径越大，图片越大
),
```

```
new Container(
 width: 200.0,
 height: 200.0,
 margin: const EdgeInsets.all(20.0),
 decoration: new BoxDecoration(
     color: Colors.white,
     image: new DecorationImage(image: new NetworkImage(this.imgsrc), fit:
BoxFit.cover),
     shape: BoxShape.rectangle,              // <-- 这里需要设置为 rectangle
     borderRadius: new BorderRadius.all(
         const Radius.circular(20.0),        // <-- rectangle 时,
BorderRadius 才有效
     ),
 ),
),
```

文本组件Text

Text组件属性及描述

属性	描述
textAlign	对齐：left、start、right、end、center、justify
textDirection	TextDirection.ltr:从左到右、TextDirection.rtl：从右到左
softWrap	是否自动换行
overflow	截取部分展示：clip：直接截取 fade：渐隐 ellipsis：省略号，省略的部分是以单词为单位，而不是字母，比如 hello worddfsa fdafafsasfs ，显示hello …
textScaleFactor	字体缩放
maxLines	显示到最大行数

TextStyle属性及描述

属性	描述
inherit	是否继承父组件的属性，默认true，这个属性极少需要设置为false，设置为false字体默认为白色、10pixels
color	字体颜色
fontSize	字体大小
fontWeight	字体粗细 一般使用的属性：FontWeight normal（默认） 、FontWeight bold（粗体）
fontStyle	字体：normal和italic
fontFamily	设置字体，注意和 fontStyle的区别
letterSpacing	字母间距，默认0，负数间距越小，正数 间距越大
wordSpacing	单词 间距，默认0，负数间距越小，正数 间距越大，注意和letterSpacing的区别，比如hello，h、e、l、l、o各是一个字母，hello是一个单词
textBaseline	字体基线
height	乘以fontSize做为行高
shadows	阴影

> 注：如果是想设置富文本，使用*RichText*
>
> **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)**

按钮相关组件

MaterialButton

有主题的button，不推荐使用，推荐使用它的子类（RaisedButton、FlatButton、OutlineButton），默认大小是 88 * 36 的大小，有圆角、阴影，点击的时候高亮，有 onpress 属性的一个控件，可以响应用户点击事件。

属性	描述
onPressed	按下之后松开的回调
onHighlightChanged	onPressed!=null 的时候可以看出 相当于用户按下时（高亮） 或者 松开时（不高亮）的监听
textColor	里面文本的颜色
disabledTextColor	当状态为 disable的时候 文本的颜色，onpress=null 为disable （子类有效）
color	当 enable （onpress != null） 背景色
disabledColor	onpress = null 的时候的颜色（子类有效）
highlightColor	当用户 按下高亮时 的颜色
elevation	Z轴阴影 默认是2 ， 当 enable 的时候的阴影
highlightElevation	高亮时的阴影 默认是 8 被按下的时候
minWidth	最小的宽度默认是88。 在 ButtonTheme 规定的
height	高度， 默认是 36 也是在 ButtonTheme 规定的
child	子控件
shape	边框样式

RaiseButton：凸起的材质矩形按钮，按下会有一个触摸效果

```
RaisedButton(
  child: Text('按钮标题'),
  onPressed: (()=>{
    print('点击事件')
  }),
)
```

FlatButton:扁平按钮，一般没有阴影，没有边框，且大部分用在 toolbar，dialogs

DropdownButton:菜单组按钮

```
DropdownButton(
    items: getListData(),
    hint:new Text('下拉选择你的性别'),//当没有默认值的时候可以设置的提示
    value: value,//下拉菜单选择完之后显示给用户的值
    onChanged: (T){//下拉菜单item点击之后的回调
      setState(() {
        value=T;
      });
    },
```

```
    elevation: 24,//设置阴影的高度
    style: new TextStyle(//设置文本框里面文字的样式
      color: Colors.red
    ),
    iconSize: 50.0,//设置三角标icon的大小
  )


  List<DropdownMenuItem> getListData(){
    List<DropdownMenuItem> items=new List();
    DropdownMenuItem dropdownMenuItem1=new DropdownMenuItem(
      child:new Text('男'),
      value: '男',
    );
    items.add(dropdownMenuItem1);
    DropdownMenuItem dropdownMenuItem2=new DropdownMenuItem(
      child:new Text('女'),
      value: '女',
    );
    items.add(dropdownMenuItem2);
    return items;
  }
```

悬浮按钮

FloatingActionButton 简称 FAB ,可以实现浮动按钮。

```
floatingActionButton: FloatingActionButton(
    child: Icon(Icons.add, color: Colors.black, size: 40),
),
floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
```

> 通过FloatingActionButton与bottomNavigationBar的结合，可以实现如下效果
>
> 
>

IconButton

可交互的Icon

```
IconButton(
  icon: Icon(Icons.access_time),
  iconSize: 30,
  onPressed:(()=>{
    print('onPressed')
  })
)
```

单组件容器布局组件

Container(容器布局):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

```
class Container_Property2 extends StatelessWidget {
  @override
   Widget build(BuildContext context) {
     return Center(
       child: Container(
         margin: const EdgeInsets.all(20),
         color: Colors.red,
         width: 150,
         height: 150,
       ),
     );
   }
}
```

Padding(填充布局):用于处理容器和其子元素之间的间距，与padding对应的是margin，margin处理容器与其他组件之间的间距。

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

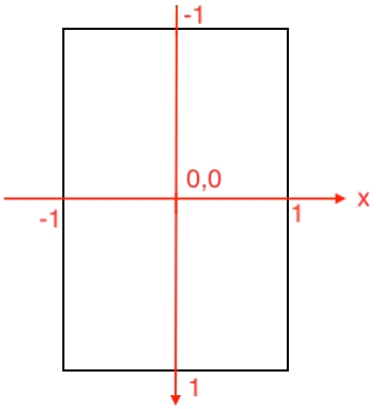
```
new Padding(
  padding: const EdgeInsets.only(
    top: 10.0,
    left: 15.0,
    right: 15.0,
    bottom: 0.0
  ),
  child: new Image.network(
    company.logo,
    width: 50.0,
    height: 50.0,
  ),
)
```

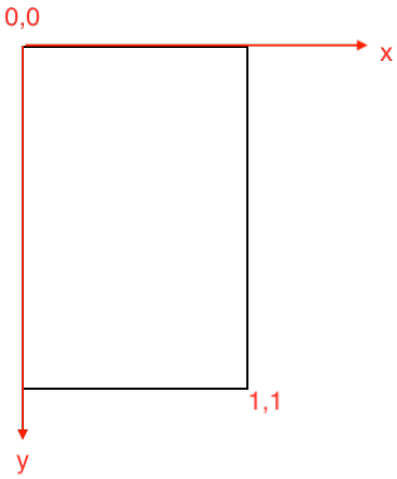
Center(居中布局):子元素处于水平和垂直方向的中间位置

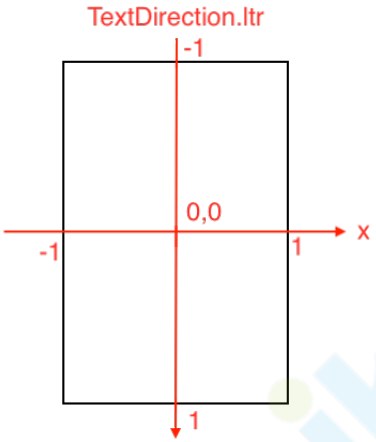
```
Center(
  child: Text('center')
)
```

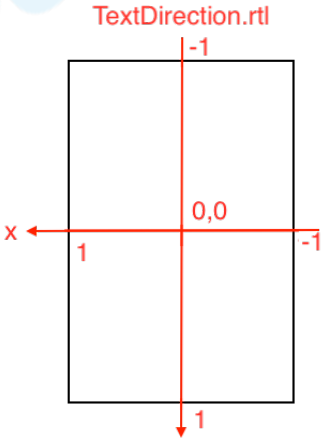
Align(对齐布局):可以将子组件按照指定的方式对齐，并根据子组件的大小调整自己的大小

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









多组件容器布局组件

Row(水平布局)

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

多组件容器都是通过children设置显示的内容

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。

> 注意：在项目目录下可以使用flutter create .生成代码依赖项

5. 暗号

初见Flutter

6. 作业 && 答疑

完成如下效果

6. 下节课内容

- 手势
- 可滚动组件
  - ListView
  - GridView
  - SingleChildScrollView
  - Table