

# Flutter 写 UI 的方式 —— 声明式

Flutter 采用了一种新的方式来写 UI，这种方式就是声明式，刚开始你可能会不适应，但用过之后你就会发现它非常好用。

## Android 、iOS 的 UI 布局的写法

在讲如何写 Flutter UI 之前，首先我们回顾一下 Android 和 iOS 的是如何布局的：

- Android：XML

Android 中，通过 XML 来写布局。

- Android：代码

代码以 命令式 的方法来写布局

- iOS：Storyboard

使用 Storyboard 来组织视图和设置约束。

- iOS：代码

可以在 ViewController 中以 命令式 的编程方式设置约束。

在 Android 中通常使用 XML 来写 UI，XML 也是一种声明式，但使用 XML 写 UI，是不能在代码里直接操作 UI，还需要一步 `findViewById` 的步骤；在 iOS 中通常使用 命令式 的代码来写 UI，命令式 可以直接在代码里操作 UI，但是要写复杂的界面的时候，就会很痛苦。

# Flutter UI 布局的写法

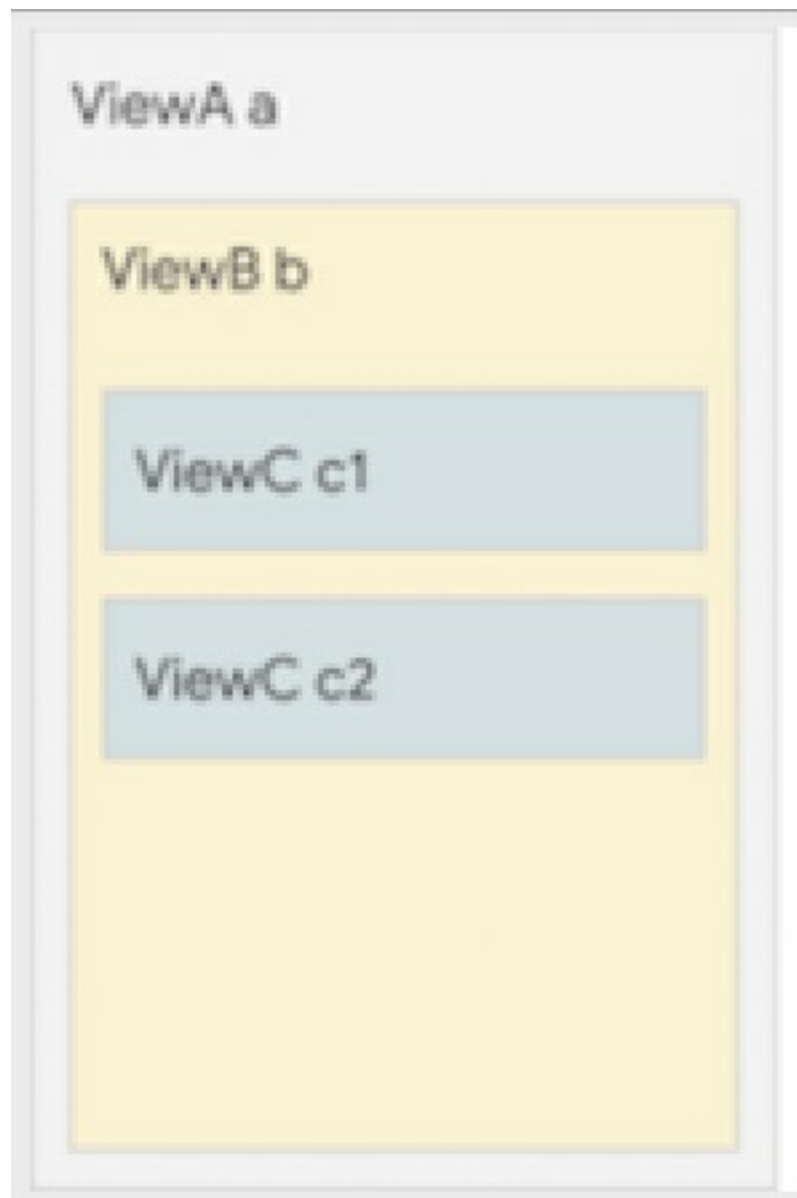
Flutter 写 UI 布局：

1. 只能用代码来写 UI
2. 而且 Flutter 用代码写 UI 的方式是声明式的，和其他平台用代码来写 UI 的命令式不同。

为了看出 Flutter 声明式和其他平台命令式的不同，接下来举一个例子说明。

## 命令式 VS 声明式

假设要实现一个下面的界面：



## 使用命令式实现

如果用命令式的代码来实现，那么代码就会是这样子的（）：

```
// 分别实现各个view
ViewA a = new ViewA(...)
ViewB b = new ViewB(...)
ViewC c1 = new ViewC(...)
ViewC c2 = New ViewC(...)
// 然后将子view 添加到容器中
a.add(b)
b.add(c1)
b.add(c2)
```

你首先要通过 View 的构造函数创建 View 的实例，然后在组织 View 的层级。

## 使用声明式实现

用 Flutter 声明式 的代码实现就是这样的：

```
Widget build(BuildContext context) {
  return ViewA{
    child: ViewB{
      color: yellow,
      children: [
        ViewC(...),
        ViewC(...)
      ]
    }
  }
}
```

这种方式写 UI，有点类似于 Android 的 XML，声明了 UI 元素之间的关系和嵌套层次，但却是用代码实现的，所以少了 XML 解析的步骤，也使 Flutter UI 构建的效率更高。

# Flutter 写 UI

在看之前创建的 Flutter 工程里，用来写 UI 的代码：

```
Widget build(BuildContext context) {  
  return MaterialApp(  
    title: 'Flutter Demo',  
    theme: ThemeData(  
      primarySwatch: Colors.blue,  
    ),  
    home: Scaffold(  
      body: Center(  
        child: GestureDetector(  
          child: Text(widget.content),  
          onTap: increment,  
        ),  
      ),  
    ),  
  );  
}
```

用代码来写 UI，同时也表现出了 UI 的层级：

