

# Flutter 布局 Widget —— 层叠布局

Flutter 的层叠布局是 [Stack](https://docs.flutter.io/flutter/widgets/Stack-class.html) (<https://docs.flutter.io/flutter/widgets/Stack-class.html>), 层叠布局允许 子Widget 堆叠（按照代码中声明的顺序）,同时 子Widget 可以根据到父容器四个边的位置来确定本身的位置。

## Stack

Stack 是层叠布局，其 子Widget 会按照添加顺序确定显示层级，后面添加的会覆盖在前面添加的 Widget 上面。

## 代码所在位置

`flutter_widget_demo/lib/stack/StackWidget.dart`

## Stack 的快速上手

给 Stack 的 子Widget 添加内容：

```
Stack(  
  children: <Widget>[  
    Image.asset(  
      "images/flutter.png",  
      width: 200,  
      fit: BoxFit.cover,  
    ),  
    Text('Hello Flutter', style:  
TextStyle(fontSize: 50.0),),  
  ],  
)
```

Stack 在一个页面使用的完整 Demo 为：

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

void main() => runApp(StackWidget());

class StackWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Demo",
      theme: ThemeData(
        primaryColor: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: Text("Flutter布局Widget -- 层叠布局")),
        body: Stack(
          children: <Widget>[
            Image.asset(
              "images/flutter.png",
              width: 200,
              fit: BoxFit.cover,
            ),
            Text('Hello Flutter', style:
TextStyle(fontSize: 50.0),),
          ],
        ),
      ),
    );
  }
}
```

运行效果为：



## Stack 的构造函数及参数说明

Stack 的构造函数为：

```
class Stack extends MultiChildRenderObjectWidget
{
  Stack({
    Key key,
    this.alignment =
AlignmentDirectional.topStart,
    this.textDirection,
    this.fit = StackFit.loose,
    this.overflow = Overflow.clip,
    List<Widget> children = const <Widget>[],
  }) : super(key: key, children: children);
  ...
}
```

参数名字

参数类型

意义

key	Key	Widget 的标识
		决定如何对齐 non-positioned 子Widget 和 部分positioned Widget, 默认值为 AlignmentDirectional.topStart
alignment	AlignmentDirectional	部分positioned子Widget, 在某一个轴上没有定义的, 这个时候就使用 alignment 的值, 比如 left、right 为横轴, left 和 right 都没有定义, 就是横轴没有定义, 只要这两个一个有值, 这个轴就算有值; top、bottom 为纵轴, 同理。
textDirection	TextDirection	用于确定 alignment 的对齐方向
fit	StackFit	此参数用于决定 non-positioned子Widget 如何去适应Stack的大小
overflow	Overflow	决定如何显示超出 Stack显示范围的 子widget
children	List< Widget>	Stack布局 里排列的内容

- alignment: 对齐方式

alignment 的类型是 AlignmentDirectional:

注意这里 start 和 end 指的是 **textDirection** 给定的方向

<b>AlignmentDirectional 的值</b>	<b>含义</b>
AlignmentDirectional.topStart	上边 start 对齐
AlignmentDirectional.topCenter	上边 居中 对齐
AlignmentDirectional.topEnd	上边 end 对齐
AlignmentDirectional.centerStart	中间 start 对齐
AlignmentDirectional.center	中间 对齐
AlignmentDirectional.centerEnd	中间 end 对齐
AlignmentDirectional.bottomStart	下边 start 对齐
AlignmentDirectional.bottomCenter	下边 居中 对齐
AlignmentDirectional.bottomEnd	下边 end 对齐



- fit: non-positioned子Widget 如何去适应Stack的大小

fit 的类型是 StackFit:

<b>StackFit 的值</b>	<b>含义</b>
StackFit.loose	使用 子Widget 自身的大小
StackFit.expand	子Widget 扩伸到Stack的大小
StackFit.passthrough	Stack的父Widget 的约束无修改的传递给 Stack的子Widget

- overflow: 如何显示超出 Stack显示空间的 子widget

overflow 的类型为 Overflow:

**Overflow 的值 含义**

Overflow.visible 超出部分仍能看见

Overflow.clip 超出部分会被剪裁

# Stack 的子Widget

为了确定 子Widget 到父容器四个角的位置,Stack 将 子Widget 分为两类:

## 1. positioned 子Widget

positioned 子Widget 是指被 Positioned 嵌套起来的 Widget, Positioned 可以控制 子Widget 到父容器四个边的距离。

## 2. non-positioned 子Widget

non-positioned子Widget 就是不用 Positioned 嵌套起来的 Widget,non-positioned子Widget 使用 Stack 设置的 alignment 来确定自己在父容器里的位置。

## 1. Positioned

### Positioned 的快速上手

Positioned 在 Stack 里使用的代码 Demo 如下:

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

void main() => runApp(StackWidget());

class StackWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Demo",
      theme: ThemeData(
        primaryColor: Colors.blue,
```

```
),
home: Scaffold(
  appBar: AppBar(title: Text("Flutter布局
Widget -- 层叠布局")),
  body: Stack(
    fit: StackFit.expand,
    children: <Widget>[
      Positioned(
        left: 50,
        top: 100,
        child: Image.asset(
          "images/flutter.png",
          width: 200,
          fit: BoxFit.cover,
        ),
      ),
      Text('Hello Flutter'),
    ],
  ),
);
}
```

运行效果为：



3:02



## Flutter布局Widget -- 层叠布局

Hello Flutter



### Positioned 的构造函数及参数说明

Positioned 的构造函数为：

```

class Positioned extends ParentDataWidget<Stack>
{
    const Positioned({
        Key key,
        this.left,
        this.top,
        this.right,
        this.bottom,
        this.width,
        this.height,
        @required Widget child,
    }) : assert(left == null || right == null ||
width == null),
        assert(top == null || bottom == null ||
height == null),
        super(key: key, child: child);
    ...
}

```

参数名字	参数类型	意义	必选 or 可选
key	Key	Widget 的标识	可选
left	double	离 Stack 左边的距离	可选
top	double	离 Stack 上边的距离	可选
right	double	离 Stack 右边的距离	可选
bottom	double	离 Stack 底边的距离	可选
width	double	指定 Widget 的宽度	可选
height	double	指定 Widget 的高度	可选
children	List	Stack布局 里排列的内容	可选

注意，此处的 width、height 是用于配合 left、top 、right、bottom 来定位 Widget 的。举个例子，在水平方向上，你只能指定 left、right、width 三个属性中的两个，如指定 left 和 width 后，

right 会自动算出  $(\text{left} + \text{width})$ ，如果同时指定三个属性则会报错，垂直方向同理。