

Flutter 布局 Widget —— 弹性布局

Flutter 中的弹性布局是 [Flex](https://docs.flutter.io/flutter/widgets/Flex-class.html) (<https://docs.flutter.io/flutter/widgets/Flex-class.html>)。

Flex 类似于 CSS 的 Flexbox。

Flex 有主轴和交叉轴，Flex 里的子Widget 默认沿主轴排列，并且 Flex 和 Expanded 配合使用可以实现子widge 按照一定比例来分配父容器空间。所以 Flex 叫做弹性布局。

Flex

代码所在位置

`flutter_widget_demo/lib/flex/FlexWidget.dart`

Flex 的快速上手

Flex 有一个必填参数：direction，用于确定主轴的方向，然后就可以在 children 里写子Widget。

Flex 使用的代码为：

```
Flex(  
    direction: Axis.horizontal,  
    children: <Widget>[  
        ...  
    ],  
)
```

Flex 的构造函数及参数说明

Flex 的构造函数为：

```

class Flex extends MultiChildRenderObjectWidget {
  Flex({
    Key key,
    @required this.direction,
    this.mainAxisAlignment =
MainAxisAlignment.start,
    this.mainAxisSize = MainAxisSize.max,
    this.crossAxisAlignment =
CrossAxisAlignment.center,
    this.textDirection,
    this.verticalDirection =
VerticalDirection.down,
    this.textBaseline,
    List<Widget> children = const <Widget>[],
  }) : assert(direction != null),
        assert(mainAxisAlignment != null),
        assert(mainAxisSize != null),
        assert(crossAxisAlignment != null),
        assert(verticalDirection != null),
        assert(crossAxisAlignment !=
CrossAxisAlignment.baseline || textBaseline !=
null),
        super(key: key, children: children);
  ...
}

```

参数名字	参数类型	意义	必选 or 可选
key	Key	Widget 的标识	可选
direction	Axis	主轴的方向 表示 子Widget	必选
mainAxisAlignment	MainAxisAlignment	在主轴的对齐方	可选

mainAxisSize	MainAxisSize	式 表示主轴应该占用多大的空间	可选
crossAxisAlignment	CrossAxisAlignment	表示 子Widget 在交叉轴的对齐方式	可选
textDirection	TextDirection	表示 子Widget 在主轴方向上的布局顺序	可选
verticalDirection	VerticalDirection	表示 子Widget 在交叉轴方向上的布局顺序	可选
textBaseline	TextBaseline	排列 子Widget 时使用哪个基线	可选
children	List< Widget>	Flex布局 里排列的内容	可选

direction : 主轴的方向

direction 的类型是 Axis:

Axis 的值	含义
Axis.horizontal	主轴方向为水平方向，那么 子Widget 就会沿水平方向排列，交叉轴就是垂直方向。
Axis.vertical	主轴方向为垂直方向，那么 子Widget 就会沿垂直方向排列，交叉轴就是水平方向。



mainAxisAlignment : 子Widget 在主轴的对齐方式

mainAxisAlignment 的类型是 MainAxisAlignment:

MainAxisAlignment 的值	含义
MainAxisAlignment.start	沿着主轴的起点对齐 textDirection 必须有值，以确定是从左边开始的还是从右边开始的
MainAxisAlignment.end	沿着主轴的终点对齐 textDirection 必须有值，以确定是在左边结束的还是在右边结束的

MainAxisAlignment.center	在主轴上居中对齐
MainAxisAlignment.spaceBetween	在主轴上，两端对齐，项目之间的间隔都相等。
MainAxisAlignment.spaceAround	在主轴上，将多余的控件均匀分布给 子Widget 之间，而且第一个 子Widget 和 最后一个子Widget 距边框的距离是两个 子Widget 距离的一半
MainAxisAlignment.spaceEvenly	在主轴上，将多余的控件均匀分布给 子Widget 之间，而且第一个 子Widget 和 最后一个子Widget 距边框的距离和子Widget 之间的距离一样



mainAxisSize : 表示主轴应该占用多大的空间

MainAxisSize 的
值

含义

MainAxisSize.min 主轴的大小是能显示完 子Widget 的最小大小，
主轴的大小就是 子Widget 的大小
MainAxisSize.max 主轴能显示的最大的大小，根据约束来判断

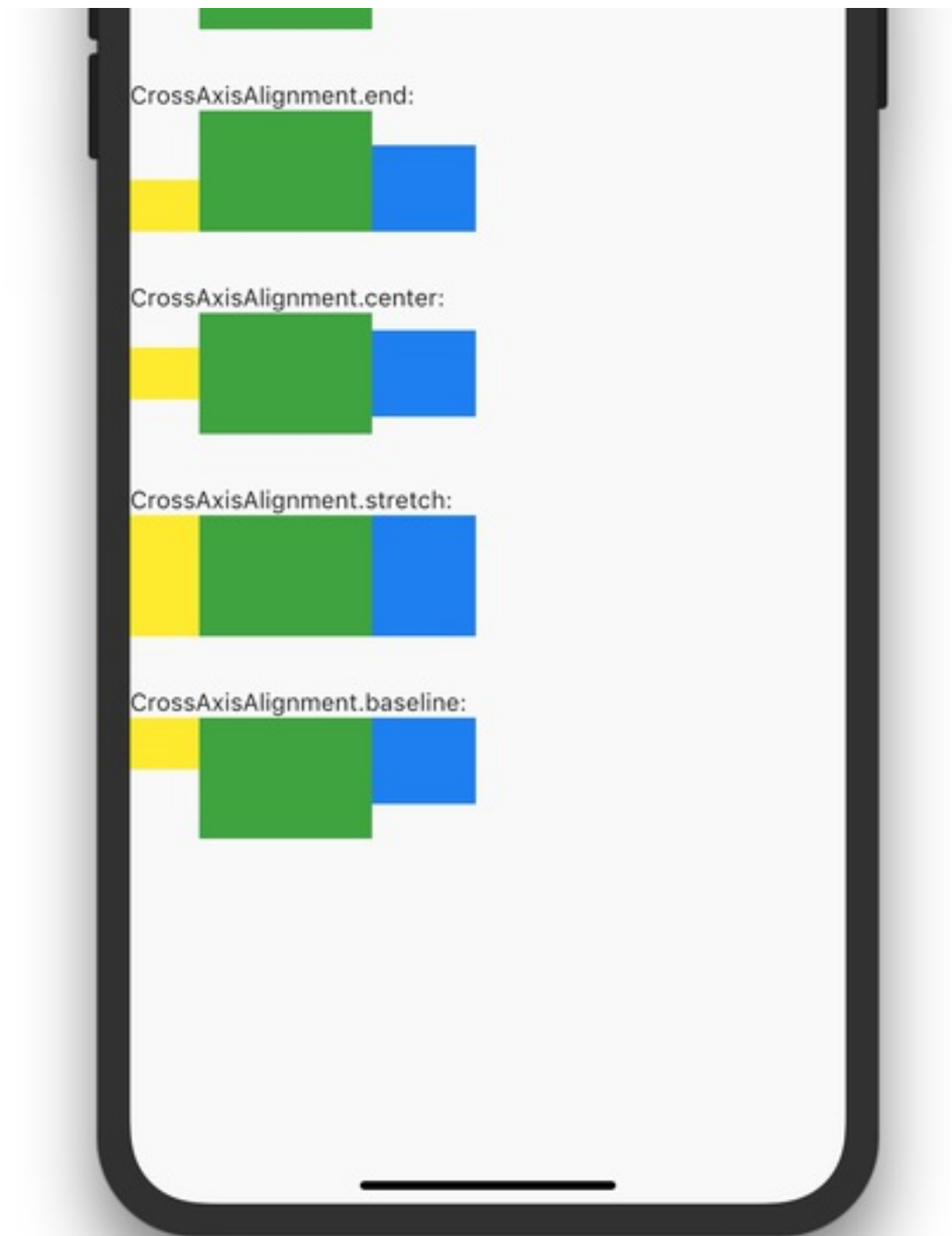
紫色代表主轴占用的空间：

crossAxisAlignment：表示 子Widget 在交叉轴的对齐方式

crossAxisAlignment 的类型是 CrossAxisAlignment：

CrossAxisAlignment 的值	含义
CrossAxisAlignment.start	沿着交叉轴的起点对齐 verticalDirection 必须有值，以确定是从左边开始的还是从右边开始的
CrossAxisAlignment.end	沿着主轴的终点对齐 verticalDirection 必须有值，以确定是在左边结束的还是在右边结束的
CrossAxisAlignment.center	在交叉轴上居中对齐
CrossAxisAlignment.stretch	要求 子Widget 在交叉轴上填满
CrossAxisAlignment.baseline	要求 子Widget 的基线在交叉轴上对齐





textDirection：表示 子Widget 在主轴方向上的布局顺序

textDirection 的类型是 TextDirection：

TextDirection 的值 含义

TextDirection.rtl 表示从右到左
TextDirection.ltr 表示从左到右

verticalDirection：表示 子Widget 在交叉轴方向上的布局顺序

verticalDirection 的类型是 VerticalDirection：

VerticalDirection 的值 含义

VerticalDirection.up 表示从下到上

VerticalDirection.down 表示从上到下

Flexible 与 Expanded

如果当 Flex 里的内容过长，超过主轴的大小，例如如下的代码：

```
Flex(  
  direction: Axis.horizontal,  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: <Widget>[  
    Text('Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!')  
  ],  
)
```

Flex 的主轴是水平方向，而Text 里的内容太多，超过了屏幕的宽度，就会抛出 layout 错误：

```
A RenderFlex overflowed by 267 pixels on the  
right.
```

界面上也会看到黑黄的条：

为了避免子Widget在Row、Column、Flex中超界，就可以使用Flexible与Expanded。Flexible与Expanded可以让Row、Column、Flex的子Widget具有弹性能力。

比如上面的例子用Flexible或Expanded来改写：

```
Flexible(  
  child: Text(  
    'Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!'),  
)
```

或

```
Expanded(  
  child: Text(  
    'Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!Hello  
Flutter!Hello Flutter!Hello Flutter!'),  
)
```

运行的效果如下：

用Flexible与Expanded来包子Widget，当子Widget要超过主轴的大小时，会自动换行，但是Flexible与Expanded也有不同的地方，Expanded是Flexible的子类。

Flexible的构造函数为：

```
class Flexible extends ParentDataWidget<Flex> {
  const Flexible({
    Key key,
    this.flex = 1,
    this.fit = FlexFit.loose,
    @required Widget child,
  }) : 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
flex	int	此 Widget 的弹性因子	可选
fit	FlexFit	如何分配 弹性Widget 在可用空间里的大小	可选
child	Widget	要显示的 Widget	必选

Expanded 的构造函数为：

```
class Expanded extends Flexible {
  /// Creates a widget that expands a child of a
  [Row], [Column], or [Flex]
  /// expand to fill the available space in the
  main axis.
  const Expanded({
    Key key,
    int flex = 1,
    @required Widget child,
  }) : super(key: key, flex: flex, fit:
  FlexFit.tight, 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
flex	int	此 Widget 的弹性因子	可选
child	Widget	要显示的 Widget	必选

可以明显看到，Flexible 和 Expanded 的 fit 参数不同，Flexible 的 fit 是 FlexFit.loose，Expanded 的 fit 参数是 FlexFit.tight。所以，当还有剩余空间时，Expanded 会占满剩余的所有空间，而 Flexible 只会占用自身大小的空间。

这里举一个例子，当 Text 的内容不够长时：

```
Flexible(  
  child: Container(  
    color: Colors.yellow,  
    child: Text('使用 Flexible 来包裹 子Widget'),  
  ),  
)
```

```
Expanded(  
  child: Container(  
    color: Colors.yellow,  
    child: Text('使用 Expanded 来包裹 子Widget'),  
  ),  
)
```

效果如下：

Flexible 会占用自身大小，而 Expanded 会占满全屏幕。

总结

Flexible 与 Expanded 可以让 Row、Column、Flex 的子Widget 具有弹性能力，当子Widget 要超过主轴的大小时，会自动换行，当还有剩余空间时，Expanded 会占满剩余的所有空间，而 Flexible 只会占用自身大小的空间。

Flexible 和 Expanded 的 flex 弹性系数

Flexible 和 Expanded 还有一个很重要的参数：flex，flex 为弹性系数，其布局过程如下：

1. 如果 flex 为0或null，则 child 是没有弹性的，称为 非弹性子Widget，非弹性子Widget 的大小就是其本身的大小，不会被扩展去占用多余的空间。
2. 如果 flex 大于0，child 是有弹性的，称为 弹性子Widget，首先会计算出第一步所有 flex为0或null 的子Widget 的大小，然后会按照 弹性子Widget的flex 占 所有弹性子Widget的 flex 总和 的比例来分割主轴的空闲空间。

Flexible 的 flex 的使用

Flexible 的 flex 的使用方式就是使用 Flexible 嵌套这些 Widget，然后设置 flex 的值：

```
Flexible(  
    flex: 1,  
    child: ...  
)
```

Demo 如下：

```
Flex(  
  direction: Axis.horizontal,  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: <Widget>[  
    Flexible(  
      flex: 1,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.yellow,  
      ),  
    ),  
    Flexible(  
      flex: 2,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.green,  
      ),  
    ),  
    Flexible(  
      flex: 1,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.blue,  
      ),  
    ),  
  ],  
)
```

使用 Flexible 包裹三个宽高都为 30 的色块，并设置 flex 为 1、2、1，效果如下：

因为 子Widget 的宽度是固定的，所以 Flexible 只会占用本身的大小。

Expanded 的 flex 使用

Expanded 的 flex 的使用方式就是使用 Expanded 嵌套这些 Widget，然后设置 flex 的值：

```
Expanded(  
  flex: 1,  
  child: ...  
)
```

Demo 代码如下：


```
Flex(  
  direction: Axis.horizontal,  
  mainAxisAlignment: MainAxisAlignment.start,  
  children: <Widget>[  
    Expanded(  
      flex: 1,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.yellow,  
      ),  
    ),  
    Expanded(  
      flex: 2,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.green,  
      ),  
    ),  
    Expanded(  
      flex: 1,  
      child: Container(  
        height: 30.0,  
        width: 30.0,  
        color: Colors.blue,  
      ),  
    ),  
  ],  
)
```

使用 Expanded 包裹三个宽高都为 30 的色块，并设置 flex 为 1、2、1，效果如下：

虽然三个色块的宽度是固定的，但是 Expanded 还是按照比例瓜分了剩余的全部空间。