# 4.2 线性布局（Row和Column）

所谓线性布局，即指沿水平或垂直方向排布子组件。Flutter中通过Row和Column来实现线性布局，类似于Android中的LinearLayout控件。Row和Column都继承自Flex，我们将在弹性布局一节中详细介绍Flex。

### 主轴和纵轴

对于线性布局，有主轴和纵轴之分，如果布局是沿水平方向，那么主轴就是指水平方向，而纵轴即垂直方向；如果布局沿垂直方向，那么主轴就是指垂直方向，而纵轴就是水平方向。在线性布局中，有两个定义对齐方式的枚举类MainAxisAlignment和CrossAxisAlignment，分别代表主轴对齐和纵轴对齐。

### Row

Row可以在水平方向排列其子widget。定义如下：

Row({  
 ...   
 TextDirection textDirection,   
 MainAxisSize mainAxisSize = MainAxisSize.max,   
 MainAxisAlignment mainAxisAlignment = MainAxisAlignment.start,  
 VerticalDirection verticalDirection = VerticalDirection.down,   
 CrossAxisAlignment crossAxisAlignment = CrossAxisAlignment.center,  
 List<Widget> children = const <Widget>[],  
})

* textDirection：表示水平方向子组件的布局顺序(是从左往右还是从右往左)，默认为系统当前Locale环境的文本方向(如中文、英语都是从左往右，而阿拉伯语是从右往左)。
* mainAxisSize：表示Row在主轴(水平)方向占用的空间，默认是MainAxisSize.max，表示尽可能多的占用水平方向的空间，此时无论子widgets实际占用多少水平空间，Row的宽度始终等于水平方向的最大宽度；而MainAxisSize.min表示尽可能少的占用水平空间，当子组件没有占满水平剩余空间，则Row的实际宽度等于所有子组件占用的的水平空间；
* mainAxisAlignment：表示子组件在Row所占用的水平空间内对齐方式，如果mainAxisSize值为MainAxisSize.min，则此属性无意义，因为子组件的宽度等于Row的宽度。只有当mainAxisSize的值为MainAxisSize.max时，此属性才有意义，MainAxisAlignment.start表示沿textDirection的初始方向对齐，如textDirection取值为TextDirection.ltr时，则MainAxisAlignment.start表示左对齐，textDirection取值为TextDirection.rtl时表示从右对齐。而MainAxisAlignment.end和MainAxisAlignment.start正好相反；MainAxisAlignment.center表示居中对齐。读者可以这么理解：textDirection是mainAxisAlignment的参考系。
* verticalDirection：表示Row纵轴（垂直）的对齐方向，默认是VerticalDirection.down，表示从上到下。
* crossAxisAlignment：表示子组件在纵轴方向的对齐方式，Row的高度等于子组件中最高的子元素高度，它的取值和MainAxisAlignment一样(包含start、end、 center三个值)，不同的是crossAxisAlignment的参考系是verticalDirection，即verticalDirection值为VerticalDirection.down时crossAxisAlignment.start指顶部对齐，verticalDirection值为VerticalDirection.up时，crossAxisAlignment.start指底部对齐；而crossAxisAlignment.end和crossAxisAlignment.start正好相反；
* children ：子组件数组。

### 示例

请阅读下面代码，先想象一下运行的结果：

Column(  
 //测试Row对齐方式，排除Column默认居中对齐的干扰  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: <Widget>[  
 Row(  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: <Widget>[  
 Text(" hello world "),  
 Text(" I am Jack "),  
 ],  
 ),  
 Row(  
 mainAxisSize: MainAxisSize.min,  
 mainAxisAlignment: MainAxisAlignment.center,  
 children: <Widget>[  
 Text(" hello world "),  
 Text(" I am Jack "),  
 ],  
 ),  
 Row(  
 mainAxisAlignment: MainAxisAlignment.end,  
 textDirection: TextDirection.rtl,  
 children: <Widget>[  
 Text(" hello world "),  
 Text(" I am Jack "),  
 ],  
 ),  
 Row(  
 crossAxisAlignment: CrossAxisAlignment.start,   
 verticalDirection: VerticalDirection.up,  
 children: <Widget>[  
 Text(" hello world ", style: TextStyle(fontSize: 30.0),),  
 Text(" I am Jack "),  
 ],  
 ),  
 ],  
);

实际运行结果如图4-1所示：

图4-1

解释：第一个Row很简单，默认为居中对齐；第二个Row，由于mainAxisSize值为MainAxisSize.min，Row的宽度等于两个Text的宽度和，所以对齐是无意义的，所以会从左往右显示；第三个Row设置textDirection值为TextDirection.rtl，所以子组件会从右向左的顺序排列，而此时MainAxisAlignment.end表示左对齐，所以最终显示结果就是图中第三行的样子；第四个Row测试的是纵轴的对齐方式，由于两个子Text字体不一样，所以其高度也不同，我们指定了verticalDirection值为VerticalDirection.up，即从低向顶排列，而此时crossAxisAlignment值为CrossAxisAlignment.start表示底对齐。

### Column

Column可以在垂直方向排列其子组件。参数和Row一样，不同的是布局方向为垂直，主轴纵轴正好相反，读者可类比Row来理解，下面看一个例子：

import 'package:flutter/material.dart';  
  
class CenterColumnRoute extends StatelessWidget {  
 @override  
 Widget build(BuildContext context) {  
 return Column(  
 crossAxisAlignment: CrossAxisAlignment.center,  
 children: <Widget>[  
 Text("hi"),  
 Text("world"),  
 ],  
 );  
 }  
}

运行效果如图4-2所示：

图4-2示例

解释：

* 由于我们没有指定Column的mainAxisSize，所以使用默认值MainAxisSize.max，则Column会在垂直方向占用尽可能多的空间，此例中为屏幕高度。
* 由于我们指定了 crossAxisAlignment 属性为CrossAxisAlignment.center，那么子项在Column纵轴方向（此时为水平方向）会居中对齐。注意，在水平方向对齐是有边界的，总宽度为Column占用空间的实际宽度，而实际的宽度取决于子项中宽度最大的Widget。在本例中，Column有两个子Widget，而显示“world”的Text宽度最大，所以Column的实际宽度则为Text("world") 的宽度，所以居中对齐后Text("hi")会显示在Text("world")的中间部分。

**实际上，Row和Column都只会在主轴方向占用尽可能大的空间，而纵轴的长度则取决于他们最大子元素的长度**。如果我们想让本例中的两个文本控件在整个手机屏幕中间对齐，我们有两种方法：

* 将Column的宽度指定为屏幕宽度；这很简单，我们可以通过ConstrainedBox或SizedBox（我们将在后面章节中专门介绍这两个Widget）来强制更改宽度限制，例如：
* ConstrainedBox(  
   constraints: BoxConstraints(minWidth: double.infinity),   
   child: Column(  
   crossAxisAlignment: CrossAxisAlignment.center,  
   children: <Widget>[  
   Text("hi"),  
   Text("world"),  
   ],  
   ),  
  );
* 将minWidth设为double.infinity，可以使宽度占用尽可能多的空间。
* 使用Center Widget；我们将在后面章节中介绍。

### 特殊情况

如果Row里面嵌套Row，或者Column里面再嵌套Column，那么只有最外面的Row或Column会占用尽可能大的空间，里面Row或Column所占用的空间为实际大小，下面以Column为例说明：

Container(  
 color: Colors.green,  
 child: Padding(  
 padding: const EdgeInsets.all(16.0),  
 child: Column(  
 crossAxisAlignment: CrossAxisAlignment.start,  
 mainAxisSize: MainAxisSize.max, //有效，外层Colum高度为整个屏幕  
 children: <Widget>[  
 Container(  
 color: Colors.red,  
 child: Column(  
 mainAxisSize: MainAxisSize.max,//无效，内层Colum高度为实际高度   
 children: <Widget>[  
 Text("hello world "),  
 Text("I am Jack "),  
 ],  
 ),  
 )  
 ],  
 ),  
 ),  
);

运行效果如图4-3所示：

图4-3

如果要让里面的Column占满外部Column，可以使用Expanded 组件：

Expanded(   
 child: Container(  
 color: Colors.red,  
 child: Column(  
 mainAxisAlignment: MainAxisAlignment.center, //垂直方向居中对齐  
 children: <Widget>[  
 Text("hello world "),  
 Text("I am Jack "),  
 ],  
 ),  
 ),  
)

运行效果如图4-4所示：

图4-4

我们将在介绍弹性布局时详细介绍Expanded。