

# Flutter 布局 Widget —— 流式布局

流式布局：页面元素的宽度可以按照屏幕分辨率进行适配调整，但整体布局不变。

在 Flutter 中的流式布局是：

[Wrap \(https://docs.flutter.io/flutter/widgets/Wrap-class.html\)](https://docs.flutter.io/flutter/widgets/Wrap-class.html)

- 为什么需要流式布局？

前面讲到的，在 Flex、Row、Column 中，当子Widget的大小超过 主轴的大小后，就会报 layout 的 overflowed 错误，会在界面上看到黄黑色的条。

这个问题也可以用流式布局解决，将上面代码的 Row 换成 Wrap:

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

main() {
  runApp(new MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new MaterialApp(
      title: 'Test',
      home: new Scaffold(
        appBar: new AppBar(title: new
Text('Flutter 布局Widget -- 流式布局')),
        body: Wrap(
          children: <Widget>[Text('Hello
Flutter ' * 100)],
        ));
  }
}
```

运行后的效果就为：

## Wrap

Wrap 会把超出屏幕显示范围的 Widget 自动换行，所以称为流式布局。

## 代码所在位置

flutter\_widget\_demo/lib/wrap/WrapWidget.dart

# Wrap 的快速上手

Wrap 需要设置主轴方向，默认的主轴方向为水平方向，给其 children 参数添加 子Widget 即可，例如：

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

Wrap 在一个页面使用的完整 Demo 如下：

```
import 'package:flutter/material.dart';  
  
main() => runApp(new WrapWidget());  
  
class WrapWidget extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return new MaterialApp(  
      title: 'Test',  
      home: new Scaffold(  
        appBar: new AppBar(title: new  
Text('Flutter 布局Widget -- 流式布局')),  
        body: Wrap(  
          direction: Axis.horizontal,  
          spacing: 8.0, // 主轴 方向间距  
          runSpacing: 12.0, // 交叉轴 方向间距  
          alignment: WrapAlignment.center,  
          runAlignment: WrapAlignment.start,
```

```
children: <Widget>[
  new Chip(
    avatar: new CircleAvatar(
      backgroundColor:
Colors.blue, child: Text('A')),
    label: new Text('AAAAAAA'),
  ),
  new Chip(
    avatar: new CircleAvatar(
      backgroundColor:
Colors.blue, child: Text('M')),
    label: new Text('BBBBBB'),
  ),
  new Chip(
    avatar: new CircleAvatar(
      backgroundColor:
Colors.blue, child: Text('H')),
    label: new Text('CCCCCCCC'),
  ),
  new Chip(
    avatar: new CircleAvatar(
      backgroundColor:
Colors.blue, child: Text('J')),
    label: new Text('DDDDDDDD'),
  ),
  new Chip(
    avatar: new CircleAvatar(
      backgroundColor:
Colors.blue, child: Text('J')),
    label: new Text('EEEEEEEE'),
  ),
  new Chip(
    avatar: new CircleAvatar(
```

```

        backgroundColor:
Colors.blue, child: Text('J')),
        label: new
Text('FFFFFFFFFFFFFFFFFF'),
    ),
  ],
)))
}
}

```

运行效果：



## Wrap 的构造函数及参数说明

Wrap 的构造函数为：

```

class Wrap extends MultiChildRenderObjectWidget {
  Wrap({
    Key key,
    this.direction = Axis.horizontal,
    this.alignment = WrapAlignment.start,
    this.spacing = 0.0,
    this.runAlignment = WrapAlignment.start,
    this.runSpacing = 0.0,
    this.crossAxisAlignment =
WrapCrossAlignment.start,
    this.textDirection,
    this.verticalDirection =
VerticalDirection.down,
    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
direction	Axis	主轴的方向 默认是 Axis.horizontal
alignment	WrapAlignment	子Widget 在主轴上的排布方式，默认值为 WrapAlignment.start WrapAlignment 的值：MainAxisAlignment

runAlignment	WrapAlignment	Wrap 会自动换成或换 runAlignment 就是每 每列的对齐方式，如果 为水平方向，就是每行 果主轴为竖直方向，就 列，默认值为 WrapAlignment.start WrapAlignment 的值 MainAxisAlignment   每行或每列之间的间距 默认是0.0 子Widget 在交叉轴上的 方式， WrapCrossAlignmen WrapCrossAlignmen 和 MainAxisAlignment 一样 表示 子Widget 在主轴 上的布局顺序
runSpacing	double	表示 子Widget 在交叉 向上的布局顺序
crossAxisAlignment	WrapCrossAlignment	Wrap布局 里排列的内
textDirection	TextDirection	
verticalDirection	VerticalDirection	
children	List< Widget>	