Flutter深入浅出组件篇---ClipRect、ClipRRect

Jimi

ClipRect介绍

`ClipRect` 是给子组件裁剪为给定的矩形大小，默认情况下，`ClipRect`会阻止其子组件在边界之外进行会话，如果需要对子组件进行大小和位置的限定，那么还可以通过自定义裁剪路径。

示例代码

本文中很多效果都没有截图，可下载源代码运行项目 源代码地址，或者通过视频教程查看 视频教程地址

什么情况下使用ClipRect?

需要对子组件进行裁剪的时候我们就使用`ClipRect`。

ClipRect的属性和说明

字段	属性	描述
clipper	CustomClipper<Rect>	自定义裁剪

clipBehavior	Clip	子组件边缘裁剪的方式，默认 Clip.hardEdge
child	Widget	子组件

ClipRect 使用

ClipRect 基本使用

我们这里展示一张图片，用`ClipRect`进行包裹，当超出的部分将会被裁剪

```
import 'package:flutter/material.dart';
import 'package:flutter_code/ClipRectExample/
ClipperPath.dart';

class ClipRectExample extends StatefulWidget {
  @override
  _ClipRectExampleState createState() =>
_ClipRectExampleState();
}

class _ClipRectExampleState extends State<ClipRectExample>
{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("ContainerExample"),
      ),
      body: Center(
```

```
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            ClipRect(
              child: Align(
                alignment: Alignment.topLeft,
                widthFactor: 0.5,
                child: Image.network("https://
img1.baidu.com/it/
u=2324541312,3167046558&fm=253&fmt=auto&app=120&f=JPEG?
w=601&h=400"),
                ),
              ),
            ],
          ),
        ),
      );
  }
}
```

效果展示

ClipRect 自定义裁剪使用

第一步：定义自定义裁剪

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

class ClipperPath extends CustomClipper<Rect>{
  @override
  Rect getClip(Size size) {
    return new Rect.fromLTRB(100, 10, size.width,
size.height);
  }

  @override
  bool shouldReclip(CustomClipper<Rect> oldClipper) {
    return true;
  }
}
```

第二步：使用自定义裁剪

```
import 'package:flutter/material.dart';
import 'package:flutter_code/ClipRectExample/
ClipperPath.dart';

class ClipRectExample extends StatefulWidget {
  @override
  _ClipRectExampleState createState() =>
_ClipRectExampleState();
}

class _ClipRectExampleState extends State<ClipRectExample>
{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("ClipRectExample"),
      ),
```

```
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            ClipRect(
              /// 自定义裁剪路径
              clipper: ClipperPath(),
              child: Align(
                alignment: Alignment.topLeft,
                widthFactor: 1,
                child: Image.network("https://
img1.baidu.com/it/
u=2324541312,3167046558&fm=253&fmt=auto&app=120&f=JPEG?
w=601&h=400"),
              ),
            ),
          ],
        ),
      ),
    );
  }
}
```

效果展示

ClipRRect 介绍

`ClipRRect` 是使用圆角矩形剪辑其子项的小部件，默认情况下，ClipRRect 使用自己的边界作为剪辑的基本矩形，但可以使用自定义剪辑器自定义剪辑的大小和位置。

什么情况下使用 ClipRRect?

当需要对子组件进行圆角裁剪的时候可以用`ClipRRect`，当然还可以自定义裁剪。

ClipRRect 的属性和说明

字段	属性	描述
borderRadius	BorderRadius	裁剪的边框大小
clipper	CustomClipper<RRect>	自定义裁剪器
clipBehavior	Clip	子组件边缘裁剪的方式，默认 Clip.hardEdge
child	Widget	子组件

ClipRRect 使用

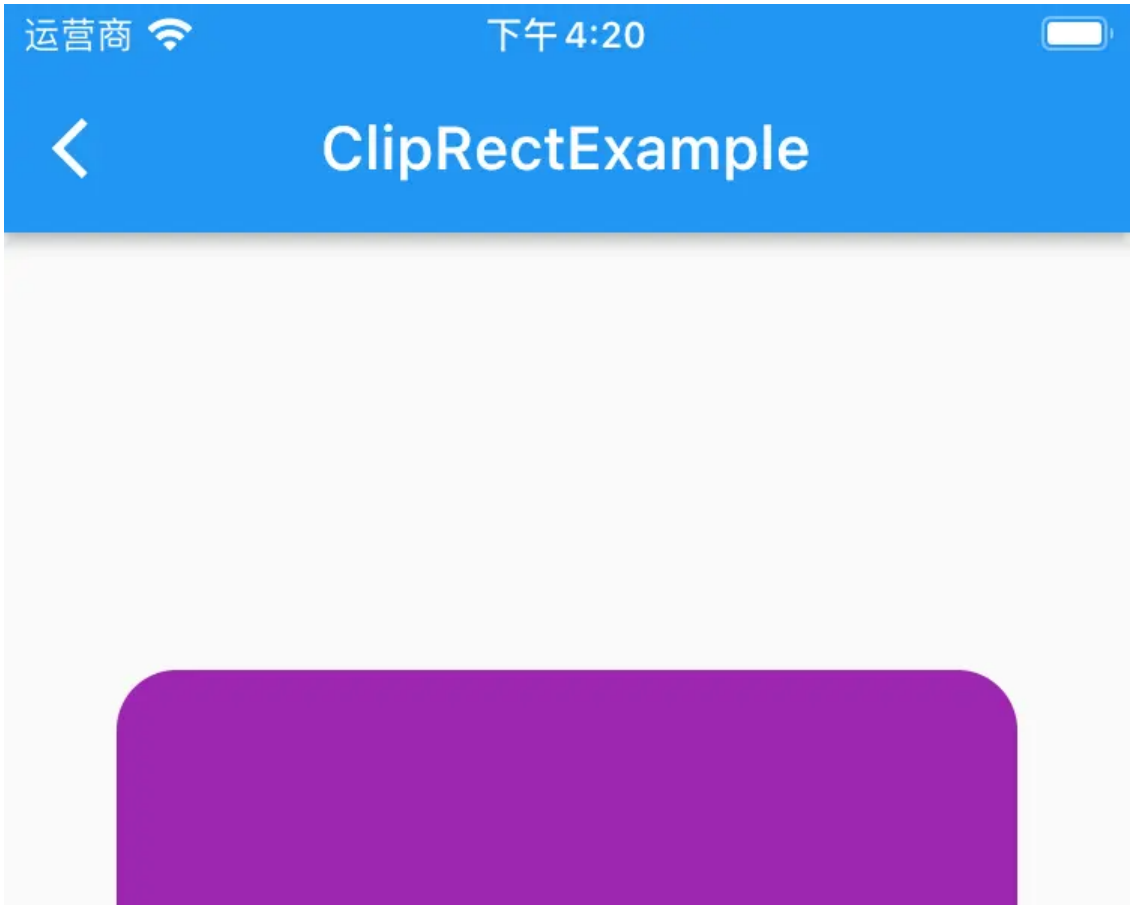
ClipRRect基本使用

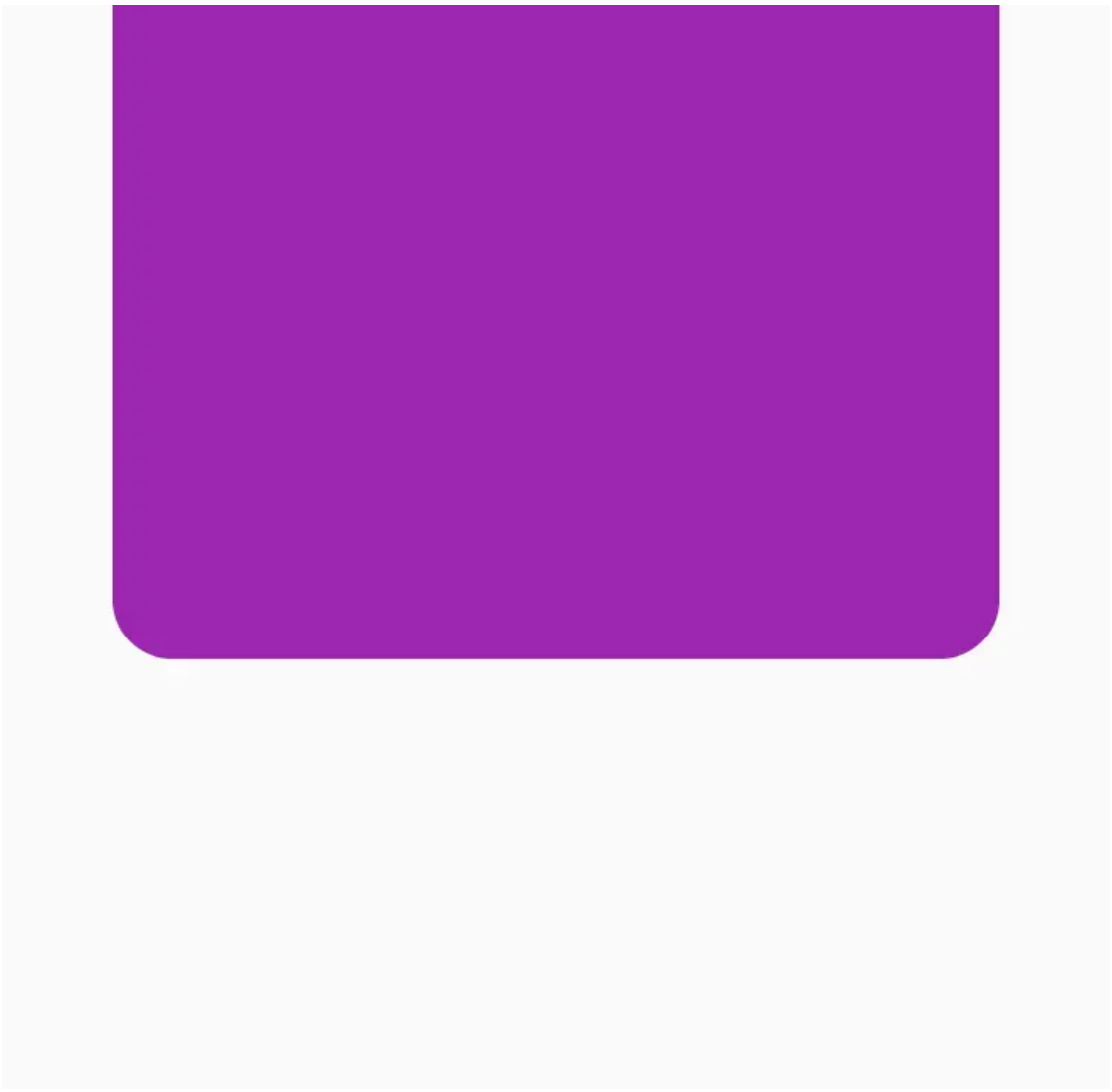
这里我们定义一个盒子的宽高是300，背景颜色是紫色，我们ClipRRect进行包裹看下效果

```
import 'package:flutter/material.dart';
import 'package:flutter_code/ClipRectExample/
ClipperPath.dart';
import 'package:flutter_code/ClipRectExample/
ClipperRPath.dart';

class ClipRectExample extends StatefulWidget {
  @override
  _ClipRectExampleState createState() =>
_ClipRectExampleState();
}

class _ClipRectExampleState extends State<ClipRectExample>
{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("ClipRectExample"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            ClipRRect(
              borderRadius:
```

```
BorderRadius.all(Radius.circular(20)),
              child: Container(
                color: Colors.purple,
                width: 300,
                height: 300,
              )
            ),
          ],
        ),
      ),
    );
  }
}
```

效果展示





ClipRRect 自定义裁剪使用

第一步：定义自定义裁剪

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

class ClipperRPath extends CustomClipper<RRect>{

  @override
  RRect getClip(Size size) {
```

```
    return RRect.fromRectAndCorners(
      Rect.fromCenter(
          center: Offset(100, 100),
          width: 200,
          height: 100
      ),
      topLeft: Radius.circular(50),
      bottomRight: Radius.circular(50)
    );
  }

  @override
  bool shouldReclip(CustomClipper<RRect> oldClipper) {
    return false;
  }
}
```

第二步：使用自定义裁剪

```
import 'package:flutter/material.dart';
import 'package:flutter_code/ClipRectExample/
ClipperPath.dart';
import 'package:flutter_code/ClipRectExample/
ClipperRPath.dart';

class ClipRectExample extends StatefulWidget {
  @override
  _ClipRectExampleState createState() =>
_ClipRectExampleState();
}

class _ClipRectExampleState extends State<ClipRectExample>
{
  @override
  Widget build(BuildContext context) {
```

```
    return Scaffold(
      appBar: AppBar(
        title: Text("ClipRectExample"),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            ClipRRect(
              clipper: ClipperRPath(),
              child: Container(
                color: Colors.purple,
                width: 300,
                height: 300,
              )
            ),
          ],
        ),
      ),
    );
  }
}
```

效果展示

总结

本章我们对`ClipRect`以及`ClipRRect`进行了讲解，他们主要的功能都是对子组件进行裁剪，还可以自定义裁剪，如果只是需要对子组件进行圆角的裁剪，我们使用`ClipRRect`就可以，因为它更加的简单。