1. 以实现视图下方圆角的效果 [实践] 2022-01-24 00:00
2. 实现 Flutter 应用原生内容的分享 [实践] 2022-01-24 00:00
3. Flutter、Dart、Flutter 插件编译过程 [实践] 2022-01-24 00:00
4. 实现多平台视频播放器的设计 [实践] 2022-01-24 00:00
5. 实现多平台图片加载器 [实践] 2022-01-24 00:00
6. 多线程并发编程入门实践 [实践] 2022-01-24 00:00
7. 实现多平台数据存储模块的设计 [实践] 2022-01-24 00:00
8. 实现多平台网络请求模块的设计 [实践] 2022-01-24 00:00
9. 实现多平台日志模块的设计 [实践] 2022-01-24 00:00

一、自定义多平台视图 CustomSingleChildLayout

自定义多平台视图 CustomSingleChildLayout 的作用是实现多平台视图自定义布局，根据视图大小，按照一定规则，调整一个子视图的位置。这个子视图通常称为视图，自定义布局，可以实现任一自定义布局逻辑，例如圆形布局，矩形布局，三角形布局，等等。

1. CustomSingleChildLayout 的用法

CustomSingleChildLayout 的作用是实现多平台视图自定义布局，SingleChildLayoutDelegate 类是实现视图，自定义布局逻辑的抽象类。

```
class CustomSingleChildLayout extends SingleChildRenderObjectWidget {
  /// 构造函数
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是子视图
  /// 参数 key 是 Widget 的 key
  CustomSingleChildLayout({SingleChildLayoutDelegate? delegate, Widget? child, Key? key})
      : _delegate = delegate,
        super(child: child, key: key);

  /// 用于创建 RenderCustomSingleChildLayoutBox 实例
  /// 参数 context 是 BuildContext 的实例
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是 RenderBox 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  /// 返回值 RenderCustomSingleChildLayoutBox 的实例
  @override
  RenderCustomSingleChildLayoutBox createRenderObject(BuildContext context, RenderCustomSingleChildLayoutBox? parent) {
    assert(_delegate != null);
    return RenderCustomSingleChildLayoutBox(delegate: _delegate, parent: parent);
  }

  /// 用于更新 RenderCustomSingleChildLayoutBox 实例的属性
  /// 参数 context 是 BuildContext 的实例
  /// 参数 renderObject 是 RenderCustomSingleChildLayoutBox 的实例
  @override
  void updateRenderObject(BuildContext context, RenderCustomSingleChildLayoutBox? renderObject) {
    assert(_delegate != null);
    assert(renderObject != null);
    renderObject!.delegate = _delegate;
  }

  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是子视图
  /// 参数 key 是 Widget 的 key
  final SingleChildLayoutDelegate? _delegate;
}
```

自定义布局逻辑，通过 SingleChildLayoutDelegate 类实现。

2. SingleChildLayoutDelegate 的抽象类

每个自定义布局，都会对应一个 SingleChildLayoutDelegate 的子类，例如 CustomSingleChildLayoutDelegate 类，这个子类用于实现自定义布局逻辑。

```
class CustomSingleChildLayoutDelegate extends SingleChildLayoutDelegate {
  /// 构造函数
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  /// 参数 key 是 Widget 的 key
  CustomSingleChildLayoutDelegate({required this.getPositionForChild, required this.getSize, this.shouldRelayout, Key? key});

  /// 用于计算子视图的位置
  /// 参数 size 是 RenderBox 的 size
  /// 参数 child 是 RenderBox 的实例
  /// 返回值 Offset 的实例，表示子视图的位置
  final Offset Function(Size size, RenderBox child) getPositionForChild;

  /// 用于计算视图的大小
  /// 参数 constraints 是 BoxConstraints 的实例
  /// 返回值 Size 的实例，表示视图的大小
  final Size Function(BoxConstraints constraints) getSize;

  /// 用于判断是否需要重新布局
  /// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
  /// 返回值 true 表示需要重新布局，false 表示不需要重新布局
  final bool Function(CustomSingleChildLayoutDelegate oldDelegate) shouldRelayout;
}
```

自定义布局逻辑，通过 getPositionForChild、getSize、shouldRelayout 三个函数实现。

3. 实现自定义圆角效果

假设视图的宽度为 width，高度为 height，圆角半径为 radius，则视图的圆角效果，可以通过以下三个步骤实现：

1. 计算视图的宽度 width、高度 height、圆角半径 radius，分别对应 size.width、size.height、radius。
2. 计算视图的左上角、右上角、右下角、左下角的坐标，分别对应 Offset(0, 0)、Offset(width, 0)、Offset(width, height)、Offset(0, height)。
3. 计算视图的左上角、右上角、右下角、左下角的圆角，分别对应 Offset(radius, radius)、Offset(width - radius, radius)、Offset(width - radius, height - radius)、Offset(radius, height - radius)。

```
class CustomSingleChildLayoutDelegate extends SingleChildLayoutDelegate {
  /// 构造函数
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  /// 参数 key 是 Widget 的 key
  CustomSingleChildLayoutDelegate({required this.getPositionForChild, required this.getSize, this.shouldRelayout, Key? key});

  /// 用于计算子视图的位置
  /// 参数 size 是 RenderBox 的 size
  /// 参数 child 是 RenderBox 的实例
  /// 返回值 Offset 的实例，表示子视图的位置
  final Offset Function(Size size, RenderBox child) getPositionForChild;

  /// 用于计算视图的大小
  /// 参数 constraints 是 BoxConstraints 的实例
  /// 返回值 Size 的实例，表示视图的大小
  final Size Function(BoxConstraints constraints) getSize;

  /// 用于判断是否需要重新布局
  /// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
  /// 返回值 true 表示需要重新布局，false 表示不需要重新布局
  final bool Function(CustomSingleChildLayoutDelegate oldDelegate) shouldRelayout;
}
```

以实现矩形的为例进行举例，效果如下所示：

```
/// 实现矩形的 getPositionForChild 函数
/// 参数 size 是 RenderBox 的 size
/// 参数 child 是 RenderBox 的实例
/// 返回值 Offset 的实例，表示子视图的位置
Offset getPositionForChild(Size size, RenderBox child) {
  return Offset(0, 0);
}

/// 实现矩形的 getSize 函数
/// 参数 constraints 是 BoxConstraints 的实例
/// 返回值 Size 的实例，表示视图的大小
Size getSize(BoxConstraints constraints) {
  return Size(200, 100);
}

/// 实现矩形的 shouldRelayout 函数
/// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
/// 返回值 true 表示需要重新布局，false 表示不需要重新布局
bool shouldRelayout(CustomSingleChildLayoutDelegate oldDelegate) {
  return false;
}
```

根据图示，根视图宽为 375 px，height 为 667px，则视图的坐标为 (0, 0)，width 为 200 px，height 为 100 px，则视图的 size 为 Size(200, 100)，getPositionForChild 函数的返回值为 Offset(0, 0)，getSize 函数的返回值为 Size(200, 100)。

```
class CustomSingleChildLayoutDelegate extends SingleChildLayoutDelegate {
  /// 构造函数
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  /// 参数 key 是 Widget 的 key
  CustomSingleChildLayoutDelegate({required this.getPositionForChild, required this.getSize, this.shouldRelayout, Key? key});

  /// 用于计算子视图的位置
  /// 参数 size 是 RenderBox 的 size
  /// 参数 child 是 RenderBox 的实例
  /// 返回值 Offset 的实例，表示子视图的位置
  final Offset Function(Size size, RenderBox child) getPositionForChild;

  /// 用于计算视图的大小
  /// 参数 constraints 是 BoxConstraints 的实例
  /// 返回值 Size 的实例，表示视图的大小
  final Size Function(BoxConstraints constraints) getSize;

  /// 用于判断是否需要重新布局
  /// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
  /// 返回值 true 表示需要重新布局，false 表示不需要重新布局
  final bool Function(CustomSingleChildLayoutDelegate oldDelegate) shouldRelayout;
}

/// 实现矩形的 getPositionForChild 函数
/// 参数 size 是 RenderBox 的 size
/// 参数 child 是 RenderBox 的实例
/// 返回值 Offset 的实例，表示子视图的位置
Offset getPositionForChild(Size size, RenderBox child) {
  return Offset(0, 0);
}

/// 实现矩形的 getSize 函数
/// 参数 constraints 是 BoxConstraints 的实例
/// 返回值 Size 的实例，表示视图的大小
Size getSize(BoxConstraints constraints) {
  return Size(200, 100);
}

/// 实现矩形的 shouldRelayout 函数
/// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
/// 返回值 true 表示需要重新布局，false 表示不需要重新布局
bool shouldRelayout(CustomSingleChildLayoutDelegate oldDelegate) {
  return false;
}
```

4. 实现圆角效果

实现圆角效果，可以通过 Canvas 的 drawPath 函数实现，drawPath 函数，参数 path 表示路径，paint 表示画笔，path 和 paint 的具体内容，请参考 Canvas 的 drawPath 函数。

```
/// 参数 canvas 是 Canvas 的实例
/// 参数 path 是 Path 的实例
/// 参数 paint 是 Paint 的实例
void drawPath(Canvas canvas, Path path, Paint paint) {
  canvas.drawPath(path, paint);
}
```

二、自定义多平台视图 CustomSingleChildLayout

1. 实现 CustomSingleChildLayout

自定义多平台视图 CustomSingleChildLayout 的作用是实现多平台视图自定义布局，CustomSingleChildLayoutDelegate 类是实现视图，自定义布局逻辑的抽象类。

```
class CustomSingleChildLayout extends SingleChildRenderObjectWidget {
  /// 构造函数
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是子视图
  /// 参数 key 是 Widget 的 key
  CustomSingleChildLayout({SingleChildLayoutDelegate? delegate, Widget? child, Key? key})
      : _delegate = delegate,
        super(child: child, key: key);

  /// 用于创建 RenderCustomSingleChildLayoutBox 实例
  /// 参数 context 是 BuildContext 的实例
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  /// 返回值 RenderCustomSingleChildLayoutBox 的实例
  @override
  RenderCustomSingleChildLayoutBox createRenderObject(BuildContext context, RenderCustomSingleChildLayoutBox? parent) {
    assert(_delegate != null);
    return RenderCustomSingleChildLayoutBox(delegate: _delegate, parent: parent);
  }

  /// 用于更新 RenderCustomSingleChildLayoutBox 实例的属性
  /// 参数 context 是 BuildContext 的实例
  /// 参数 renderObject 是 RenderCustomSingleChildLayoutBox 的实例
  @override
  void updateRenderObject(BuildContext context, RenderCustomSingleChildLayoutBox? renderObject) {
    assert(_delegate != null);
    assert(renderObject != null);
    renderObject!.delegate = _delegate;
  }

  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是子视图
  /// 参数 key 是 Widget 的 key
  final SingleChildLayoutDelegate? _delegate;
}
```

CustomSingleChildLayout 的作用是实现多平台视图自定义布局，其中，RenderCustomSingleChildLayoutBox 类是实现视图，自定义布局逻辑的抽象类。

```
class CustomSingleChildLayoutBox extends RenderBoxWithChildMixin<RenderBox> {
  /// 构造函数
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是 RenderBox 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  CustomSingleChildLayoutBox({SingleChildLayoutDelegate? delegate, RenderBox? child, RenderCustomSingleChildLayoutBox? parent});

  /// 用于计算子视图的位置
  /// 参数 size 是 RenderBox 的 size
  /// 参数 child 是 RenderBox 的实例
  /// 返回值 Offset 的实例，表示子视图的位置
  Offset Function(Size size, RenderBox child) getPositionForChild;

  /// 用于计算视图的大小
  /// 参数 constraints 是 BoxConstraints 的实例
  /// 返回值 Size 的实例，表示视图的大小
  Size Function(BoxConstraints constraints) getSize;

  /// 用于判断是否需要重新布局
  /// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
  /// 返回值 true 表示需要重新布局，false 表示不需要重新布局
  bool Function(CustomSingleChildLayoutDelegate oldDelegate) shouldRelayout;

  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是 RenderBox 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  CustomSingleChildLayoutBox({SingleChildLayoutDelegate? delegate, RenderBox? child, RenderCustomSingleChildLayoutBox? parent, Offset Function(Size size, RenderBox child)? getPositionForChild, Size Function(BoxConstraints constraints)? getSize, bool Function(CustomSingleChildLayoutDelegate oldDelegate)? shouldRelayout});
}
```

CustomSingleChildLayoutBox 的作用是实现多平台视图自定义布局，其中，RenderCustomSingleChildLayoutBox 类是实现视图，自定义布局逻辑的抽象类。

2. 实现自定义多平台视图

自定义多平台视图 CustomSingleChildLayoutBox 的作用是实现多平台视图自定义布局，CustomSingleChildLayoutDelegate 类是实现视图，自定义布局逻辑的抽象类。

```
class CustomSingleChildLayoutBox extends RenderBoxWithChildMixin<RenderBox> {
  /// 构造函数
  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是 RenderBox 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  CustomSingleChildLayoutBox({SingleChildLayoutDelegate? delegate, RenderBox? child, RenderCustomSingleChildLayoutBox? parent, Offset Function(Size size, RenderBox child)? getPositionForChild, Size Function(BoxConstraints constraints)? getSize, bool Function(CustomSingleChildLayoutDelegate oldDelegate)? shouldRelayout});

  /// 用于计算子视图的位置
  /// 参数 size 是 RenderBox 的 size
  /// 参数 child 是 RenderBox 的实例
  /// 返回值 Offset 的实例，表示子视图的位置
  Offset Function(Size size, RenderBox child) getPositionForChild;

  /// 用于计算视图的大小
  /// 参数 constraints 是 BoxConstraints 的实例
  /// 返回值 Size 的实例，表示视图的大小
  Size Function(BoxConstraints constraints) getSize;

  /// 用于判断是否需要重新布局
  /// 参数 oldDelegate 是 CustomSingleChildLayoutDelegate 的实例
  /// 返回值 true 表示需要重新布局，false 表示不需要重新布局
  bool Function(CustomSingleChildLayoutDelegate oldDelegate) shouldRelayout;

  /// 参数 delegate 是 SingleChildLayoutDelegate 的实例
  /// 参数 child 是 RenderBox 的实例
  /// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
  /// 参数 getPositionForChild 是 getPositionForChild 的实现
  /// 参数 getSize 是 getSize 的实现
  /// 参数 shouldRelayout 是 shouldRelayout 的实现
  CustomSingleChildLayoutBox({SingleChildLayoutDelegate? delegate, RenderBox? child, RenderCustomSingleChildLayoutBox? parent, Offset Function(Size size, RenderBox child)? getPositionForChild, Size Function(BoxConstraints constraints)? getSize, bool Function(CustomSingleChildLayoutDelegate oldDelegate)? shouldRelayout});
}
```

3. 实现 layout 中的定义子视图的位置

自定义多平台视图 CustomSingleChildLayoutBox 的作用是实现多平台视图自定义布局，其中，performLayout 函数，是 layout 的实现，用于计算子视图的位置，performLayout 函数，参数 size 表示视图的 size，参数 child 表示子视图，参数 parent 表示视图的父视图。

```
/// 参数 size 是 RenderBox 的 size
/// 参数 child 是 RenderBox 的实例
/// 参数 parent 是 RenderCustomSingleChildLayoutBox 的实例
void performLayout(Size size, RenderBox? child, RenderCustomSingleChildLayoutBox? parent) {
  assert(_delegate != null);
  assert(child != null);
  assert(parent != null);
  assert(size != null);
  assert(size.isFinite);
  assert(size.isPositive);
  assert(size.isNonNegative);
  assert(size.isNonZero);
  assert(size.isNaN);
  assert(size.isInfinite);
  assert(size.isNaN == false);
  assert(size.isInfinite == false);
  assert(size.isNonZero == true);
  assert(size.isNonNegative == true);
  assert(size.isPositive == true);
  assert(size.isFinite == true
```