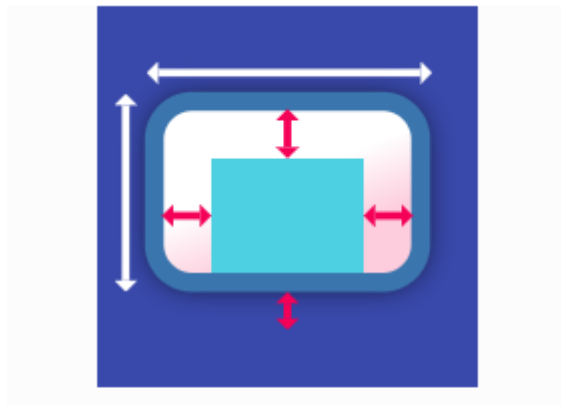


# Container容器类



## Container

一个拥有绘制、定位、调整大小的 widget。

一个方便的小部件，结合常见的绘画，定位和大小 小部件。

一个容器首先围绕着孩子 [填充](#) (任何夸大了 边界出现在 [装饰](#))，然后应用更多 [约束](#) 填充程度 (加入 width 和 height 作为约束条件，如果是 null)。 然后包围容器 额外的空空间的描述 [保证金](#)。

在绘画过程中，容器首先适用 [变换](#)，然后 描绘了 [装饰](#) 填充填充程度，那么它描绘了孩子，最后描绘 [foregroundDecoration](#) 也填充垫 程度。

容器没有孩子要尽可能大，除非传入的 约束是无界的，在这种情况下，他们试图尽可能小 可能的。 有孩子的容器大小自己孩子。 的 width, height, [约束](#) 构造函数的参数覆盖 这一点。

## 布局行为

看到 [BoxConstraints](#) 介绍盒布局模型。

自 [容器](#) 结合其他一些部件每个国家都有他们自己的 布局的行为，[容器](#) 的布局行为有点复杂。

t1; 博士: [容器](#) 努力，为了: 荣誉 [对齐](#) 大小本身 的 [孩子](#)，荣誉 width, height, [约束](#)，扩大符合家长，要尽可能小。

更具体地说：

如果小部件没有孩子, 没有height, 没有width, 没有[约束](#), 和家长提供了无限的约束[容器](#)试图 尺寸尽可能小。

如果小部件没有孩子, 也没有[对齐](#), 但height, width, 或[约束](#)提供, 那么[容器](#)试图尽可能小 可能考虑到这些约束和父母的 约束。

如果小部件没有孩子, 没有height, 没有width, 没有[约束](#), 没有[对齐](#), 但父母提供了有界约束[容器](#)父母提供的扩展以适应约束。

如果小部件有一个[对齐](#)和家长提供了无限 约束, 那么[容器](#)试图大小本身周围的孩子。

如果小部件有一个[对齐](#), 家长提供了有界的 约束, 那么[容器](#)试图扩大适应父母, 然后孩子在本身的位置[对齐](#)。

否则, 小部件有一个[孩子](#)但是没有height, 没有width, 没有[约束](#), 没有[对齐](#), [容器](#)通过了约束从父母到孩子和尺寸匹配 的孩子。

的[保证金](#)和[填充](#)属性也影响到布局, 如描述 在这些属性的文档。(仅仅增加的影响上述规则)。 的[装饰](#)可以隐式地增加[填充](#) (例如在一个边界[BoxDecoration](#)做出贡献[填充](#)); 看到[Decoration.padding](#)。

## 示例代码

这个例子显示了一个48 x48绿色广场(放置在一个[中心](#)小部件 情况下, 父部件的大小都有自己的意见[容器](#)应), 保证金, 让它远离吗 相邻的部件:

```
new Center(  
  child: new Container(  
    margin: const EdgeInsets.all(10.0),  
    color: const Color(0xFF00FF00),  
    width: 48.0,  
    height: 48.0,  
  ),  
)
```

这个例子展示了如何使用的许多特性[容器](#)在一次。 的[约束](#)将合适的字体大小+充足的空间 垂直, 水平扩展以适应父母。 的[填充](#)是 用于确保内容和文本之间的空间。 的color让蒂尔。 的[对齐](#)导致[孩子](#)是 集中在盒子里。 的[foregroundDecoration](#)覆盖 nine-patch形象 到文本上。 最后, [变换](#)适用于轻微旋转的 整个装置完成的效果。

```
new Container(  
  constraints: new BoxConstraints.expand(  
    height: Theme.of(context).textTheme.display1.fontSize * 1.1 + 200.0,  
  ),  
)
```

```
padding: const EdgeInsets.all(8.0),
color: Colors.teal.shade700,
alignment: Alignment.center,
child: new Text('Hello World', style:
Theme.of(context).textTheme.display1.copyWith(color: Colors.white)),
foregroundDecoration: new BoxDecoration(
  image: new DecorationImage(
    image: new NetworkImage('https://www.example.com/images/frame.png'),
    centerSlice: new Rect.fromLTRB(270.0, 180.0, 1360.0, 730.0),
  ),
),
transform: new Matrix4.rotationZ(0.1),
)
```

参见:

- [AnimatedContainer](#) 当一个变种, 顺利启动属性 他们的变化。
- [边境](#), 使用的样品 [容器](#) 严重。
- [墨水](#), 它描绘了一幅 [装饰](#) 在一个 [材料](#), 允许 [InkResponse](#) 和 [墨水池](#) 溅到油漆。
- 的 [的目录布局小部件](#)。

继承

- [对象](#)
- [Diagnosticable](#)
- [DiagnosticableTree](#)
- [小部件](#)

- [StatelessWidget](#)

- 容器

## 构造函数

[容器](#) ({[关键](#) 关键, [AlignmentGeometry](#) 对齐, [EdgeInsetsGeometry](#) 填充, [颜色](#) 颜色, [装饰](#) 装饰, [装饰](#) foregroundDecoration, [双](#) 宽度, [双](#) 高度, [BoxConstraints](#) 约束, [EdgeInsetsGeometry](#) 保证金, [Matrix4](#) 变换, [小部件](#) 孩子})

创建一个小部件, 结合常见的绘画、定位、规模小部件。[...]

## 属性

[对齐](#) → [AlignmentGeometry](#)

对齐的child在容器内。[...]

最后

[孩子](#) → [小部件](#)

的child由容器中。[...]

最后

[约束](#) → [BoxConstraints](#)

额外的限制适用于儿童。[...]

最后

[装饰](#) → [装饰](#)

后面的装饰涂料child。[...]

最后

[foregroundColor](#) → [装饰](#)

前面的装饰涂料child。

最后

[保证金](#) → [EdgeInsetsGeometry](#)

空的空间包围decoration和child。

最后

[填充](#) → [EdgeInsetsGeometry](#)

空的空间内雕decoration。 的child(如果有的话) 放在这个填充。 [\[...\]](#)

最后

[变换](#) → [Matrix4](#)

之前画的变换矩阵应用容器。

最后

[hashCode](#) → [int](#)

这个对象的哈希码。 [\[...\]](#)

只读的, 遗传的

[关键](#) → [关键](#)

控制一个小部件替换另一个小部件在树上。 [\[...\]](#)

最后, 继承了

[runtimeType](#) → [类型](#)

一个对象的运行时类型的代表。

只读的, 遗传的

## 方法

[构建](#) ([BuildContext](#) 上下文) → [小部件](#)

描述了由这个小部件的用户界面的一部分。 [\[...\]](#)

[debugFillProperties](#) ([DiagnosticPropertiesBuilder](#) 属性) → 无效

[createElement](#)() → [StatelessElement](#)

创建一个[StatelessElement](#)在树上来管理这个小部件的位置。 [\[...\]](#)

继承了

[debugDescribeChildren](#)() → [列表](#) <[DiagnosticsNode](#)>

返回一个列表[DiagnosticsNode](#)描述该节点的对象 的孩子。 [\[...\]](#)

@protected, 继承了

[noSuchMethod](#)([调用](#) 调用) → 动态

当用户访问一个不存在的方法或属性调用。 [\[...\]](#)

继承了

[toDiagnosticsNode](#) ({[字符串](#) 的名字, [DiagnosticsTreeStyle](#) 风格}) → [DiagnosticsNode](#)

返回一个对象被调试的调试表示 工具和[toStringDeep](#)。 [\[...\]](#)

继承了

[toString](#) ({[DiagnosticLevel](#) minLevel: [DiagnosticLevel.debug](#)}) → [字符串](#)

返回该对象的字符串表示。

继承了

[toStringDeep](#) ({[字符串](#) prefixLineOne:”, [字符串](#)  
[字符串](#) prefixOtherLines, [DiagnosticLevel](#) minLevel:DiagnosticLevel.debug}) →  
[字符串](#)

返回一个字符串表示该节点及其后代。 [\[...\]](#)

继承了

[toStringShallow](#) ({[字符串](#) 乔伊  
纳:”、“”, [DiagnosticLevel](#) minLevel:DiagnosticLevel.debug}) →[字符串](#)  
返回一行详细描述的对象。 [\[...\]](#)

继承了

[toStringShort](#)() →[字符串](#)

短, 这个小部件的文本描述。

继承了

## 操作

[运算符==](#) (动态 其他) →[bool](#)

相等操作符。 [\[...\]](#)

继承了