Grid

Material Design 响应式布局的栅格可适应屏幕大小和方向，确保布局在不同尺寸之间的一致性。

Grid 栅格组件 能确保不同布局间的视觉层面的舒适感，同时在众多不同设计中保持灵活性。 Material Design 基于 12 列的网格布局来做到 UI 的响应式。

{{"component": "modules/components/ComponentLinkHeader.js"}}

> ⚠️ `栅格` *组件不要与承载大量数据的表格（data grid）进行混淆；这个组件更倾向于在布局中使用。 如果需使用承载大量数据的表格，请看这里的* `数据表格` *组件*。

工作原理

栅格系统是通过 `Grid` 组件实现的：

- 它使用 CSS 弹性盒子模块 来做到高度灵活。
- 它有两种类型的布局： *containers* ， _items_。
- 每项的宽度是按百分比设置的，所以它们的大小总是相对于它们的父元素流动。
- 子项目（items）使用内边距来保持和其他块（items）的间距。
- 其中五个断点可供使用：xs，sm，md，lg 和 xl。
- Integer values can be given to each breakpoint, indicating how many of the 12 available columns are occupied by the component when the viewport width satisfies the breakpoint constraints.

若你对 **flexbox 不太熟悉**，我们建议你阅读 CSS-Tricks flexbox 手册。

Fluid grids 流式网格

流式网格可以通过列（column）来缩放和调整内容的大小。 在使用流式网格布局时，可以通过断点来决定该布局是否需要大幅改变。

基本栅格

每一列的宽度是 1 到 12 之间的整数值；这些宽度应用于任何断点，且表明了组件占用多少列。

A value given to a breakpoint applies to all the other breakpoints wider than it (unless overridden, as you can read later in this page). For example, `xs={12}` sizes a component to occupy the whole viewport width regardless of its size. 例如，无论组件的大小如何， `xs={12}` 都会占据整个视口的宽度。 For example, `xs={12}` sizes a component to occupy the whole viewport width regardless of its size.

{{"demo": "BasicGrid.js", "bg": true}}

有断点的栅格

Components may have multiple widths defined, causing the layout to change at the defined breakpoint. Width values given to larger breakpoints override those given to smaller breakpoints. 你可以给较大的断点指定宽度值。 那么它会覆盖给较小断点指定的宽度值。

For example, `xs={12} sm={6}` sizes a component to occupy half of the viewport width (6 columns) when viewport width is 600 or more pixels. For smaller viewports, the component fills all 12 available columns. 对于较小的视口，该组件将填充所有 12 个可用的列。 For smaller viewports, the component fills all 12 available columns.

{{"demo": "FullWidthGrid.js", "bg": true}}

Spacing 间距

To control space between children, use the `spacing` prop. The spacing value can be any positive number, including decimals and any string. 该属性借助 [`theme.spaming()`] 被转换为 CSS 属性。

{{"demo": "SpacingGrid.js", "bg": true}}

行、列间距

`rowSpacing` 和 `columnSpacing` 属性允许独立指定行和列间距。 The `rowSpacing` and `columnSpacing` props allow for specifying the row and column gaps independently. It's similar to the `row-gap` and `column-gap` properties of [CSS Grid].

{{"demo": "RowAndColumnSpacing.js", "bg": true}}

响应式的值

您可以根据活动的断点切换属性的值。 You can switch the props' value based on the active breakpoint. For instance, we can implement the ["recommended"] responsive layout grid of Material Design.

{{"demo": "ResponsiveGrid.js", "bg": true}}

下列属性支持响应式的值：

- `columns`
- `columnSpacing`
- `direction`
- `rowSpacing`
- `spacing`
- 系统中的所有[其它属性]

> ⚠️ *When using a responsive `columns` prop, each grid item needs its corresponding breakpoint. For instance, this is not working. The grid item misses the value for `md`: 例如，这种做法行不通。 For instance, this is not working. 网格项目丢失了 `md` 的值：*
>
> ```
> <Grid container columns={{ xs: 4, md: 12 }}>
>    <Grid item xs={2} />
> > </Grid>
> ```

交互式

下面是一个交互式的演示，你也可以探索不同设置下的视觉结果：

{{"demo": "InteractiveGrid.js", "hideToolbar": true, "bg": true}}

自适应布局

自适应布局可以让 *子项（items）* 之间平均地利用空间。 这也意味着你可以显式设置一个 *子项（item）* 的宽度，而使其他项的大小根据其宽度自动进行调整。

{{"demo": "AutoGrid.js", "bg": true}}

负边距

The Auto-layout makes the *items* equitably share the available space. That also means you can set the width of one *item* and the others will automatically resize around it.

The `Grid` component is using CSS flexbox internally. But as seen below, you can easily use [the system]() and CSS Grid to layout your pages.

复杂的栅格

The following demo doesn't follow the Material Design guidelines, but illustrates how the grid can be used to build complex layouts.

{{"demo": "ComplexGrid.js", "bg": true}}

嵌套栅格

The `container` and `item` props are two independent booleans; they can be combined to allow a Grid component to be both a flex container and child.

> *一个 flex **容器** 是通过将 `flex` 或 `inline-flex` 的计算显示赋予给一个元素而生成的。 Flex 容器的流入子容器称为 flex **items**， 它们使用 flex 布局模型进行布局。*

https://www.w3.org/TR/css-flexbox-1/#box-model

{{"demo": "NestedGrid.js", "bg": true}}

⚠ 给 Flex 容器、Flex 子项以及同时带有间距的 Grid 元素定义一个显式宽度会导致意外的行为，需要避免这样做：

```
<Grid spacing={1} container item xs={12}>
```

如果你需要这样做，那么请移出其中的一个属性。

列

You can change the default number of columns (12) with the `columns` prop.

{{"demo": "ColumnsGrid.js", "bg": true}}

设计局限

负边距

项目之间的边距以负边距的形式来实现。 这样做的话可能会产生意料之外的结果。 The spacing between items is implemented with a negative margin. This might lead to unexpected behaviors. For instance, to apply a background color, you need to apply `display: flex;` to the parent.

white-space: nowrap;

弹性布局元素的默认属性值为 `min-width: auto` 。 当子元素使用 `white-space: nowrap;` 时会出现布局冲突。 您可以从以下内容看到这个问题：

```
<Grid item xs>
  <Typography noWrap>
```

为了使项留在容器内，您需要设置 `min-width: 0` 。 在实际操作中，你可以设置 `zeroMinWidth` 属性来实现它：

```
<Grid item xs zeroMinWidth>
  <Typography noWrap>
```

{{"demo": "AutoGridNoWrap.js", "bg": true}}

direction: column | column-reverse

`direction="column"` 和 `direction="column-reverse"` 的容器**不支持**和断点有关的 `xs` ， `sm` ， `md` ， `lg` ，以及 `xl` 这几个属性。

They define the number of grids the component will use for a given breakpoint. 它们决定在某个断点下组件占几个网格。 They define the number of grids the component will use for a given breakpoint. They are intended to control **width** using `flex-basis` in `row` containers but they will impact height in `column` containers. If used, these props may have undesirable effects on the height of the `Grid` item elements. 如果使用这些属性，可能会对 `Grid` 块元素的高度产生副作用。 If used, these props may have undesirable effects on the height of the `Grid` item elements.

CSS 栅格布局

The `Grid` component is using CSS flexbox internally. But as seen below, you can easily use the system and CSS Grid to layout your pages. But as seen below, you can easily use the system and CSS Grid to layout your pages. But as seen below, you can easily use the system and CSS Grid to layout your pages.

{{"demo": "CSSGrid.js", "bg": true}}

System props

As a CSS utility component, the `Grid` supports all `system` properties. You can use them as props directly on the component. For instance, a padding: You can use them as props directly on the component. For instance, a padding: You can use them as props directly on the component. For instance, a padding:

```
<Grid item p={2}>
```