网格

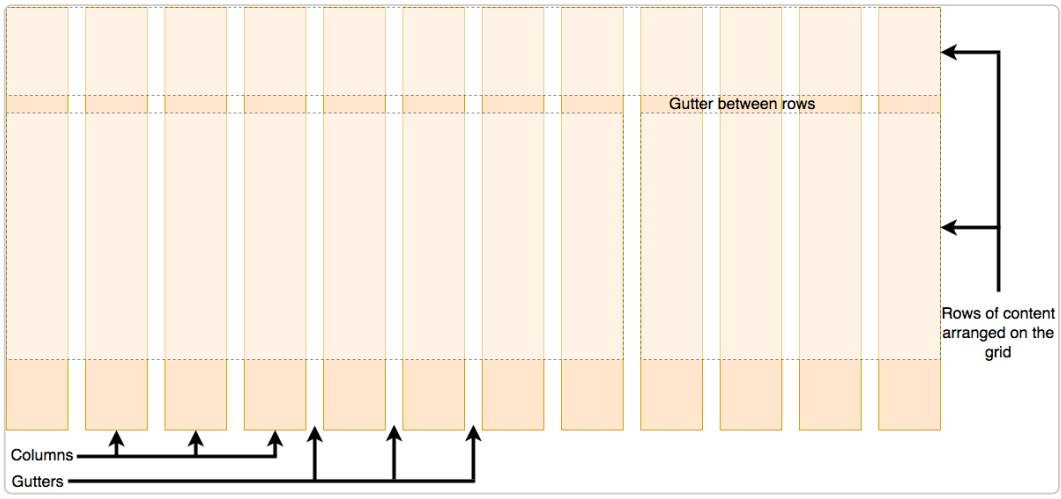
CSS Grid Layout 是一种用于网络的二维布局系统。它使您可以按行和列布置内容。它具有许多功能，可以使构建复杂的布局变得简单明了。本文将解释您开始使用网格布局所需了解的所有知识。

先决条件：	HTML 基础知识（研究 HTML 简介）和 CSS 工作原理的概念（研究 CSS和 样式框简介。）
客观的：	了解网格布局的基本概念以及如何使用 CSS 网格实现它。

什么是网格布局?

网格是水平线和垂直线的集合，创建了一个图案，我们可以根据该图案排列我们的设计元素。它们帮助我们创建布局，使我们的元素不会在我们从一个页面移动到另一个页面时跳来跳去或改变宽度，从而在我们的网站上提供更大的一致性。

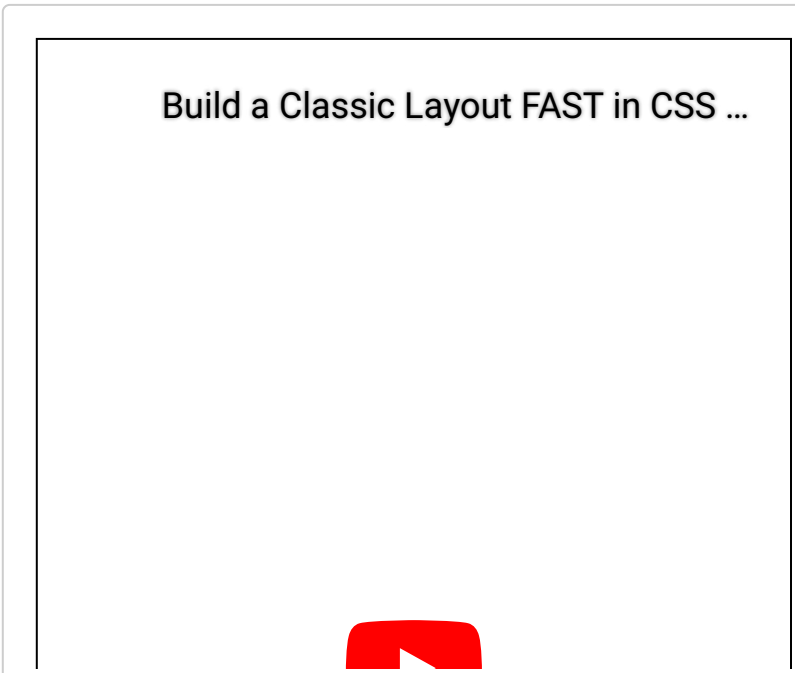
网格通常有**columns**、**rows**，然后是每行和每列之间的间隙。这些间隙通常称为**排水沟**。



在 CSS 中创建网格

决定了你的设计需要的网格后，你可以使用 CSS Grid Layout 来创建它。我们将首先了解网格布局的基本功能，然后探索如何为您的项目创建一个简单的网格系统。

以下视频提供了使用 CSS 网格的直观解释：



定义网格

作为起点，下载并在文本编辑器和浏览器中打开起点文件（您也可以 在此处实时查看 ）。您将看到一个带有容器的示例，其中包含一些子项。默认情况下，这些显示在正常流中，因此框显示在另一个下方。我们将在本课的第一部分使用此文件，进行更改以查看其网格的行为方式。

要定义网格，我们使用属性 `grid` 的值 `display` 。与 Flexbox 一样，这会启用网格布局；容器的所有直接子代都成为网格项。将此添加到文件中的 CSS：

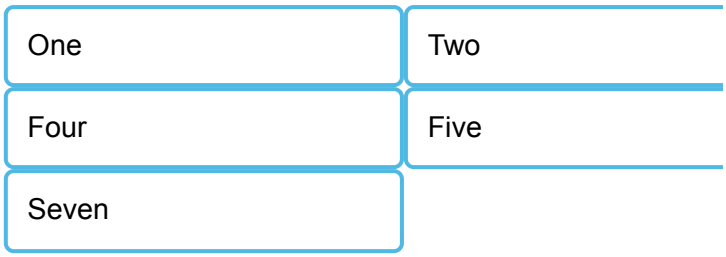
```
.container {
  display: grid;
}
```

与 Flexbox 不同，项目不会立即看起来有任何不同。声明 `display: grid` 为您提供了一个单列网格，因此您的项目将继续像在正常流程中一样显示在另一个下方。

要查看看起来更像网格的内容，我们需要向网格添加一些列。让我们添加三个 200 像素的列。您可以使用任何长度单位或百分比来创建这些列轨道。

```
.container {
  display: grid;
  grid-template-columns: 200px 200px 200px;
}
```

将第二个声明添加到您的 CSS 规则，然后重新加载页面。您应该会看到这些项目已经重新排列，以便在网格的每个单元格中都有一个。



使用 fr 单元的灵活网格

除了使用长度和百分比创建网格外，我们还可以使用 `fr` . 该 `fr` 单位代表网格容器中可用空间的一小部分，以灵活调整网格行和列的大小。

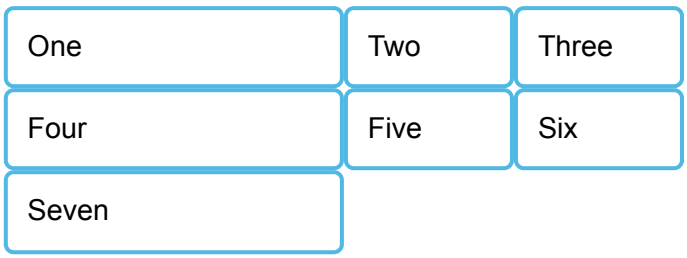
将您的曲目列表更改为以下定义，创建三个 `1fr` 曲目：

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

您现在拥有灵活的轨道。单位 `fr` 按比例分配空间。您可以像这样为曲目指定不同的正值：

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
}
```

第一个轨道获取 `2fr` 可用空间，另外两个轨道获取可用空间 `1fr`，使第一个轨道变大。您可以混合 `fr` 使用固定长度的单位。在这种情况下，固定轨道所需的空间首先用完，然后再将剩余空间分配给其他轨道。



> **注意：** 该 `fr` 单元分配*可用*空间，而不是*所有*空间。因此，如果您的其中一个曲目内部有大东西，则可以共享的可用空间就会减少。

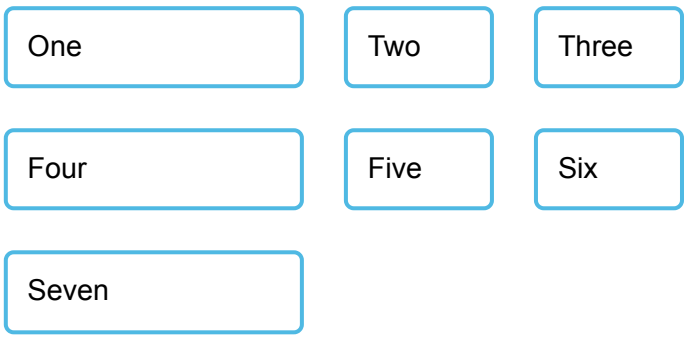
轨道之间的间隙

要在轨道之间创建间隙，我们使用以下属性：

- `column-gap` 列之间的差距
- `row-gap` 对于行之间的间隙
- `gap` 作为两者的简写

```
.container {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  gap: 20px;
}
```

这些间隙可以是任何长度单位或百分比，但不能是 `fr` 单位。



> **注意：** 属性 `gap` （ `column-gap` , `row-gap` 和 `gap` ）过去以 为前缀 `grid-` 。规范已更改，但前缀版本将作为别名保留。为了安全起见并使您的代码更安全，您可以添加这两个属性：
>
> ```
> .container {
>   display: grid;
>   grid-template-columns: 2fr 1fr 1fr;
>   grid-gap: 20px;
>   gap: 20px;
> }
> ```

重复曲目列表

您可以使用 CSS 功能重复全部或部分曲目列表 `repeat()` 。将您的曲目列表更改为以下内容：

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  gap: 20px;
}
```

您现在将 `1fr` 像以前一样获得三首曲目。传递给该函数的第一个值 `repeat()` 指定您希望列表重复的次数，而第二个值是一个曲目列表，它可以是您想要重复的一个或多个曲目。

隐式和显式网格

到目前为止，我们只指定了列轨道，但正在创建行来保存我们的内容。*这是显式*网格与*隐式*网格的示例。

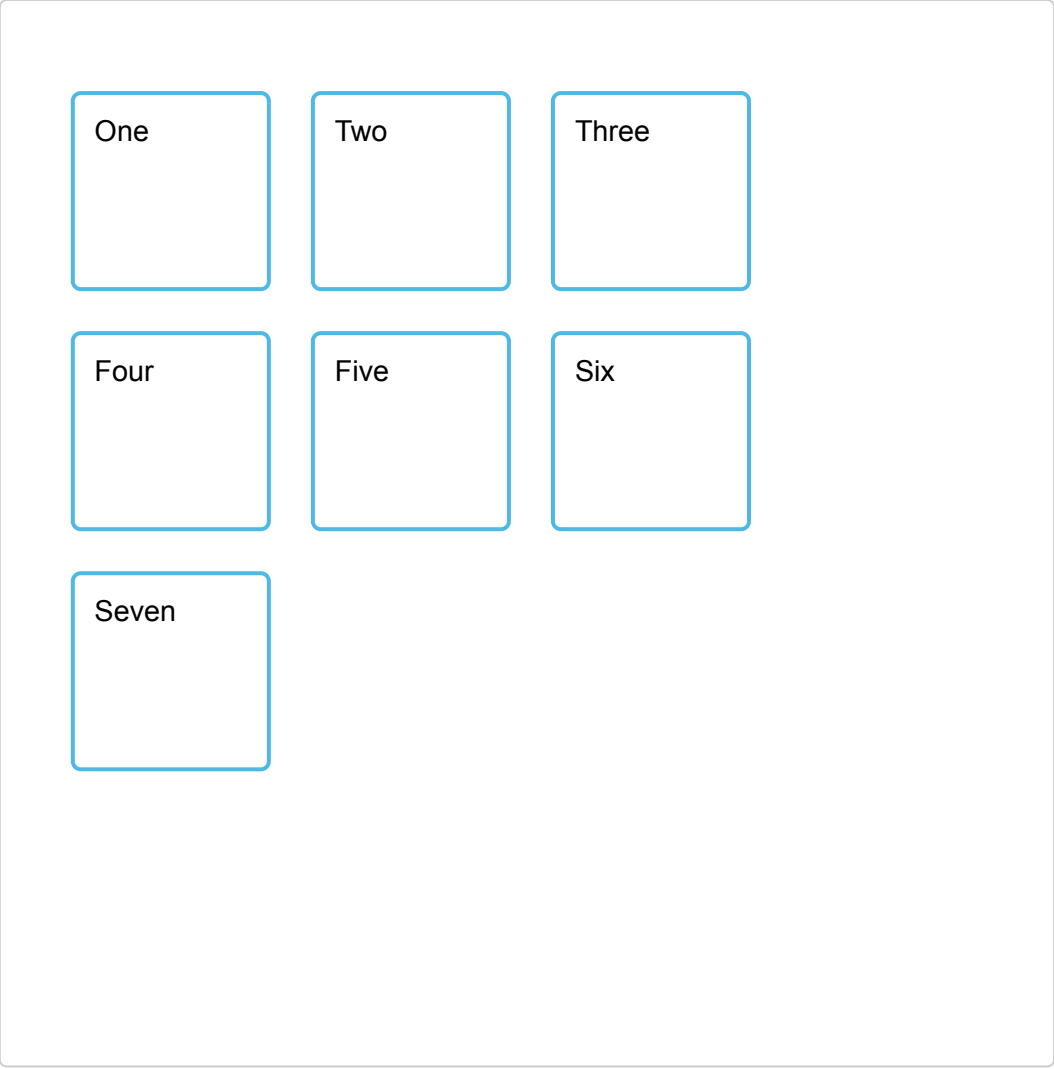
区别：

- 显式网格：使用 `grid-template-columns` 或创建 `grid-template-rows` 。

- 隐式网格：当内容放置在该网格之外时扩展定义的显式网格，例如通过绘制额外的网格线进入我们的行。

默认情况下，在隐式网格中创建的轨道是 `auto` 有大小的，这通常意味着它们足够大以容纳它们的内容。如果你想给隐式网格轨道一个大小，你可以使用 `grid-auto-rows` 和 `grid-auto-columns` 属性。如果您将 `grid-auto-rows` 值添加 `100px` 到您的 CSS，您将看到那些创建的行现在高 100 像素。

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: 100px;
  gap: 20px;
}
```



最小最大值() 函数

如果我们将内容添加到高于 100 像素的轨道中，我们的 100 像素高轨道将

可能会导致试图在每个维度上都达到像素完美的设计出现问题。

该 `minmax()` 函数允许我们设置轨道的最小和最大大小，例如 `minmax(100px, auto)`。最小尺寸为 100 像素，但最大尺寸为 `auto`，它会扩展以容纳更多内容。尝试更改 `grid-auto-rows` 为使用 minmax 值：

```
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-auto-rows: minmax(100px, auto);
  gap: 20px;
}
```

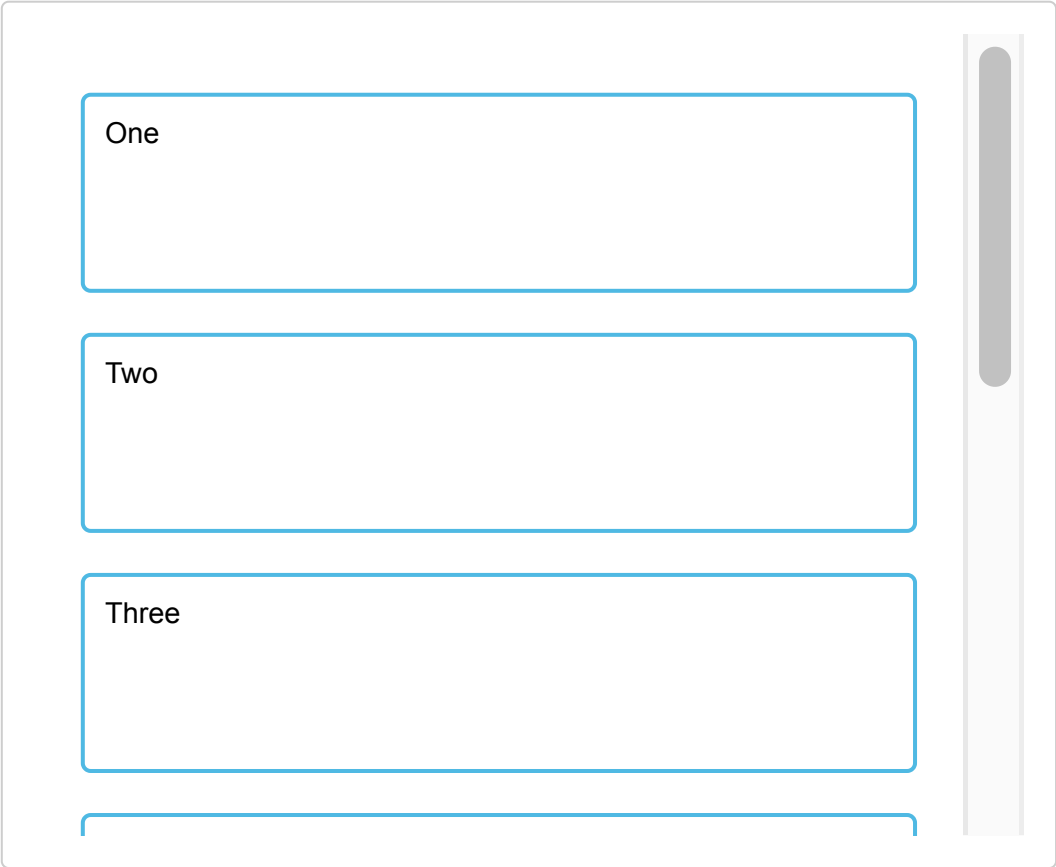
如果您添加额外的内容，您会看到轨道会扩展以适应它。请注意，扩展发生在该行的右边。

尽可能多的列

`minmax()` 我们可以结合我们在曲目列表、重复记谱和创建有用模式方面学到的一些经验教训。有时能够要求网格创建适合容器的尽可能多的列会很有帮助。 `grid-template-columns` 我们通过设置using函数的值来做到这一点 `repeat()`，但不是传入数字，而是传入关键字 `auto-fill`。对于函数的第二个参数，我们使用 `minmax()` 的最小值等于我们想要的最小轨道大小，最大值为 `1fr`。

现在使用下面的 CSS 在您的文件中试试这个：

```
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  grid-auto-rows: minmax(100px, auto);
  gap: 20px;
}
```



这是可行的，因为网格正在创建尽可能多的 200 像素的列以适应容器，然后在所有列之间共享剩余的空间。 `1fr` 正如我们所知，最大值是在轨道之间均匀分配空间。

基于行的放置

我们现在从创建网格转到在网格上放置东西。我们的网格总是有线条——这些线条以 1 开头并与文档的书写模式相关。例如，英语中的第 1 列行（从左到右书写）位于网格的左侧，第 1 行行位于顶部，而在阿拉伯语中（从右到左书写），第 1 列行会在右边。

我们可以通过指定开始和结束行来按照这些行安排事物。我们使用以下属性来做到这一点：

- `grid-column-start`
- `grid-column-end`
- `grid-row-start`
- `grid-row-end`

这些属性都可以有一个行号作为它们的值。您还可以使用速记属性：

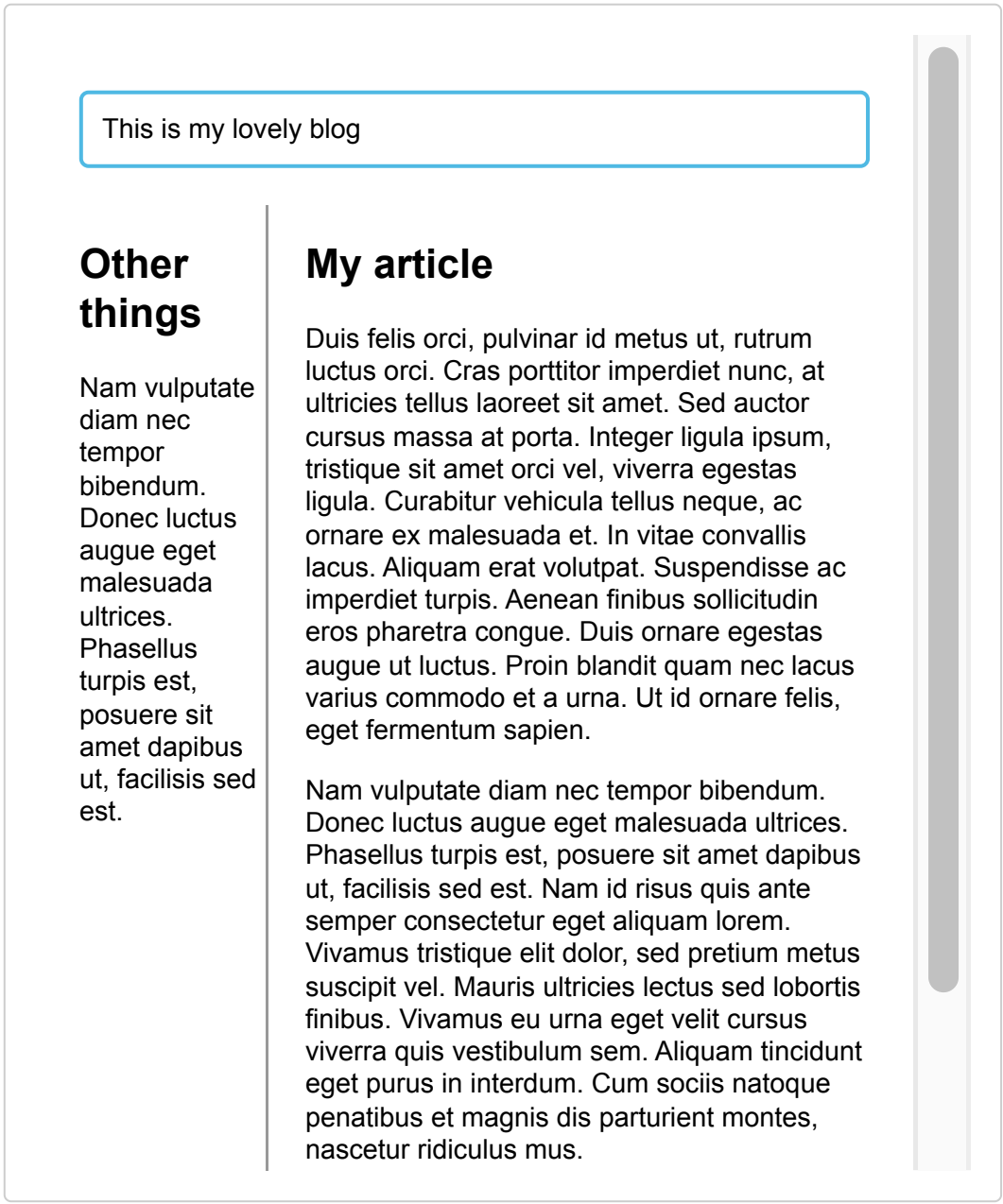
- grid-column
- grid-row

这些允许您一次指定开始和结束行，用正斜杠分隔 / 。

下载此文件作为起点 或在此处实时查看 。它有一个定义的网格和一个简单的文章概述。您可以看到*自动放置*是将每个项目放置到网格上它自己的单元格中。

让我们使用网格线来安排我们网站的所有元素。将以下规则添加到 CSS 的底部：

```
header {
  grid-column: 1 / 3;
  grid-row: 1;
}

article {
  grid-column: 2;
  grid-row: 2;
}

aside {
  grid-column: 1;
  grid-row: 2;
}

footer {
  grid-column: 1 / 3;
  grid-row: 3;
}
```



> **注意：** 您还可以使用该值 `-1` 来定位结束列或行行，然后使用负值从末尾向内计数。另请注意，线总是从显式网格的边缘开始计数，而不是隐式网格。

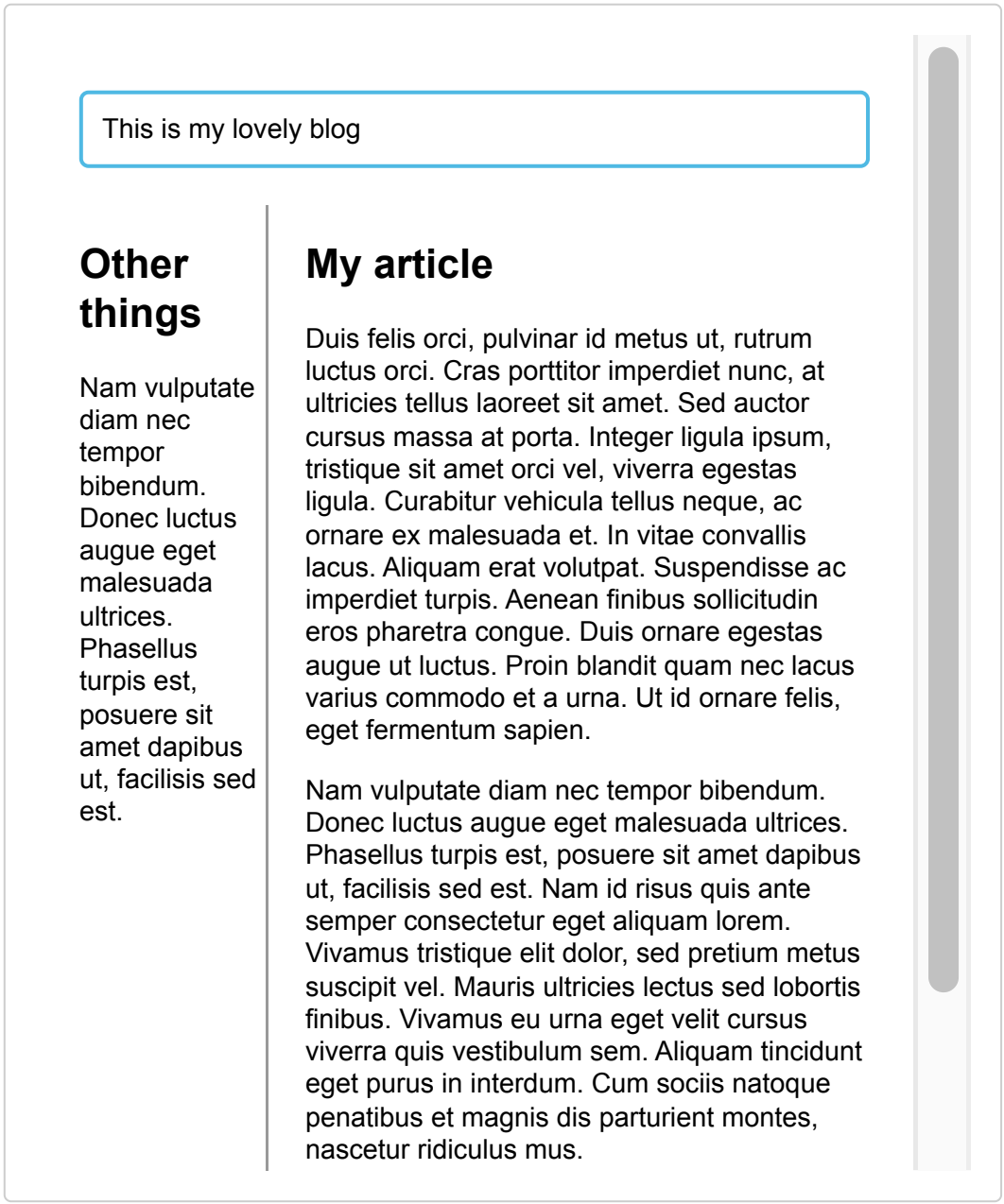
使用网格模板区域定位

在网格上排列项目的另一种方法是使用该 `grid-template-areas` 属性并为设计的各种元素命名。

从上一个示例中删除基于行的定位（或重新下载文件以获得新起点）并添加以下 CSS。

```
.container {
  display: grid;
  grid-template-areas:
    "header header"
    "sidebar content"
    "footer footer";
  grid-template-columns: 1fr 3fr;
  gap: 20px;
}

header {
  grid-area: header;
}

article {
  grid-area: content;
}

aside {
  grid-area: sidebar;
}

footer {
  grid-area: footer;
}
```

重新加载页面，您会看到您的物品已经像以前一样放置，我们不需要使用任何行号！



规则 `grid-template-areas` 如下：

- 您需要填充网格的每个单元格。
- 要跨越两个单元格，请重复名称。
- 要将单元格留空，请使用 `.`（句点）。
- 区域必须是矩形的——例如，不能有 L 形区域。
- 区域不能在不同的位置重复。

您可以尝试我们的布局，将页脚更改为仅位于文章下方，将侧边栏一直向下延伸。这是描述布局的一种非常好的方式，因为仅通过查看 CSS 就可以清楚地知道到底发生了什么。

CSS Grid 中的网格框架

网格“框架”往往基于 12 或 16 列网格。使用 CSS Grid，您不需要任何第三方工具来为您提供这样的框架——它已经存在于规范中。

下载起点文件 。它有一个容器，其中定义了 12 列网格和我们在前两个示例中使用的相同标记。我们现在可以使用基于行的放置将我们的内容放置在 12 列网格上。

```
header {
  grid-column: 1 / 13;
  grid-row: 1;
}

article {
  grid-column: 4 / 13;
  grid-row: 2;
}

aside {
  grid-column: 1 / 4;
  grid-row: 2;
}

footer {
  grid-column: 1 / 13;
  grid-row: 3;
}
```

This is my lovely blog

Other things

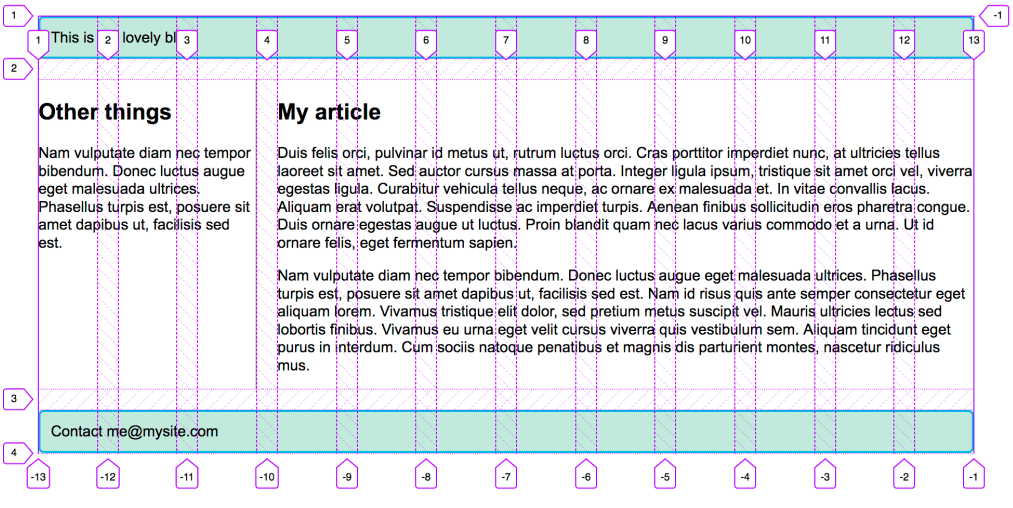
Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est.

My article

Duis felis orci, pulvinar id metus ut, rutrum luctus orci. Cras porttitor imperdiet nunc, at ultricies tellus laoreet sit amet. Sed auctor cursus massa at porta. Integer ligula ipsum, tristique sit amet orci vel, viverra egestas ligula. Curabitur vehicula tellus neque, ac ornare ex malesuada et. In vitae convallis lacus. Aliquam erat volutpat. Suspendisse ac imperdiet turpis. Aenean finibus sollicitudin eros pharetra congue. Duis ornare egestas augue ut luctus. Proin blandit quam nec lacus varius commodo et a urna. Ut id ornare felis, eget fermentum sapien.

Nam vulputate diam nec tempor bibendum. Donec luctus augue eget malesuada ultrices. Phasellus turpis est, posuere sit amet dapibus ut, facilisis sed est. Nam id risus quis ante semper consectetur eget aliquam lorem. Vivamus tristique elit dolor, sed pretium metus suscipit vel. Mauris ultricies lectus sed lobortis finibus.

如果您使用Firefox Grid Inspector 在您的设计上叠加网格线，您可以看到我们的 12 列网格是如何工作的。



测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否保留了此信息 — 请参阅测试您的技能：网格。

概括

在本概述中，我们介绍了 CSS 网格布局的主要功能。您应该能够开始在您的设计中使用它。要进一步深入了解规范，请阅读我们关于网格布局的指南，可以在下面找到。

也可以看看

- CSS 网格指南
- CSS Grid Inspector：检查网格布局
- CSS-Tricks Guide to Grid—— 一篇以视觉上吸引人的方式解释关于网格的一切的文章
- Grid Garden — 一款学习和更好地理解 Grid 基础知识的教育游戏

此页面最后修改于 2023 年 3 月 22 日由MDN 贡献者提供。