**背景：**

CSS Grid(网格) 布局（又称为 “Grid(网格)” ），是一个二维的基于网格的布局系统，它的目标是完全改变我们基于网格的用户界面的布局方式。CSS 一直用来布局我们的网页，但一直以来都存在这样或那样的问题。一开始我们用表格（table），然后是浮动（float），再是定位（postion）和内嵌块（inline-block），但是所有这些方法本质上都是只是 hack 而已，并且遗漏了很多重要的功能（例如垂直居中）。Flexbox 的出现很大程度上改善了我们的布局方式，但它的目的是为了解决更简单的一维布局，而不是复杂的二维布局（实际上 Flexbox 和 Grid 能协同工作，而且配合得非常好）。Grid(网格) 布局是第一个专门为解决布局问题而创建的 **CSS 模块**，只要我们一直在制作网站，我们就一直要讨论这些问题。

**基本流程为：**

首先，你必须使用 [display: grid](https://www.html.cn/archives/8510/#prop-display) 将容器元素定义为一个 grid(网格) 布局，使用 [grid-template-columns](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows) 和 [grid-template-rows](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows) 设置 列 和 行 的尺寸大小. 然后我们可以通过指定 [grid-column](https://www.html.cn/archives/8510/#prop-grid-column-row) 和 [grid-row](https://www.html.cn/archives/8510/#prop-grid-column-row) 将其网格项定位放入这个 grid(网格) 中， 也可以不指定具体的网格位置，让网格项根据网格布局默认的行为放入网格布局当中。

**重要子部件：**

网格容器（Grid Container）

应用 display: grid 的元素。这是所有 网格项（grid item）的直接父级元素。在这个例子中，container 就是**网格容器 (Grid Container)** 。

1. <div class="container">
2. <div class="item item-1"></div>
3. <div class="item item-2"></div>
4. <div class="item item-3"></div>
5. </div>

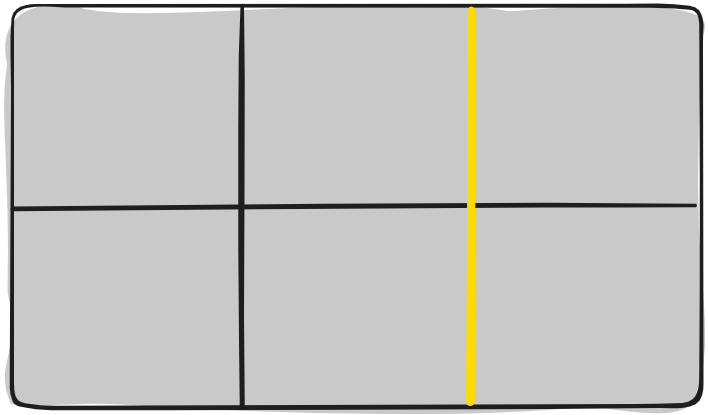
网格项(Grid Item)

网格容器（Grid Container）的子元素（例如直接子元素）。这里 item 元素就是网格项(Grid Item)，但是 sub-item 不是。

1. <div class="container">
2. <div class="item"></div>
3. <div class="item">
4. <p class="sub-item"></p>
5. </div>
6. <div class="item"></div>
7. </div>

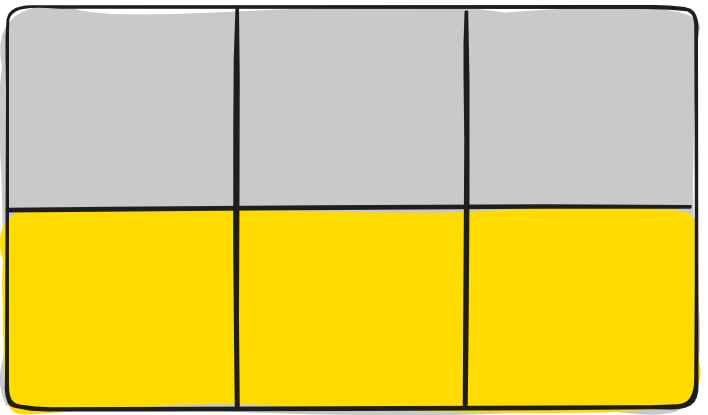
网格线(Grid Line)

构成网格结构的分界线。它们既可以是垂直的（“列网格线(column grid lines)”），也可以是水平的（“行网格线(row grid lines)”），并位于行或列的任一侧。例如，这里的黄线就是一条列网格线。



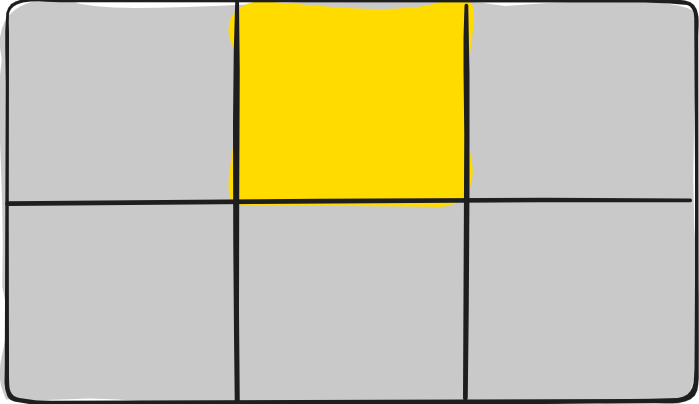
网格轨道(Grid Track)

两条相邻网格线之间的空间。你可以把它们想象成网格的列或行。下图是第二条和第三条 行网格线 之间的 网格轨道(Grid Track)。



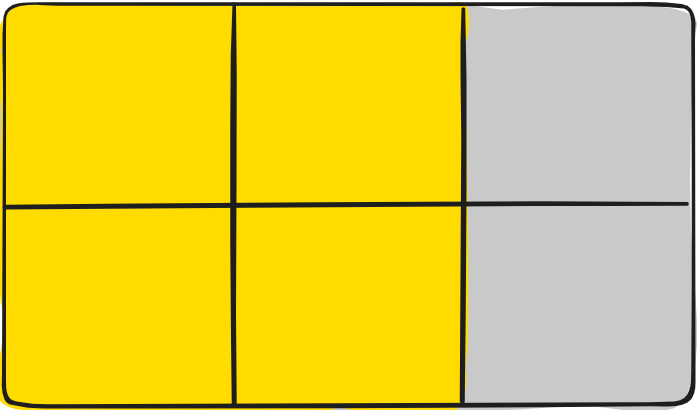
网格单元格(Grid Cell)

两个相邻的行和两个相邻的列网格线之间的空间。这是 Grid(网格) 系统的一个“单元”。下图是第 1 至第 2 条 行网格线 和第 2 至第 3 条 列网格线 交汇构成的 网格单元格(Grid Cell)。



网格区域(Grid Area)

4条网格线包围的总空间。一个 网格区域 (Grid Area) 可以由任意数量的 网格单元格 (Grid Cell) 组成。下图是 行网格线1和3，以及列网格线1和3 之间的网格区域。



**父元素 网格容器(Grid Container) 属性**

**1.display**

将元素定义为网格容器，并为其内容建立新的 网格格式上下文。

值：

grid ：生成一个块级网格

inline-grid ：生成一个内联网格

CSS 代码:

1. .container {
2. display: grid | inline-grid;
3. }

2.

**grid-template-columns/grid-template-rows**

使用空格分隔的值列表，用来定义网格的列和行。这些值表示 网格轨道(Grid Track) 大小，它们之间的空格表示网格线。

值：  
– <track-size>： 可以是长度值，百分比，或者等份网格容器中可用空间（使用 fr 单位）  
– <line-name>：你可以选择的任意名称.如果不给与名字（只留一个空格），则系统自动分配名字。如果取名则使用括号的方式命名。

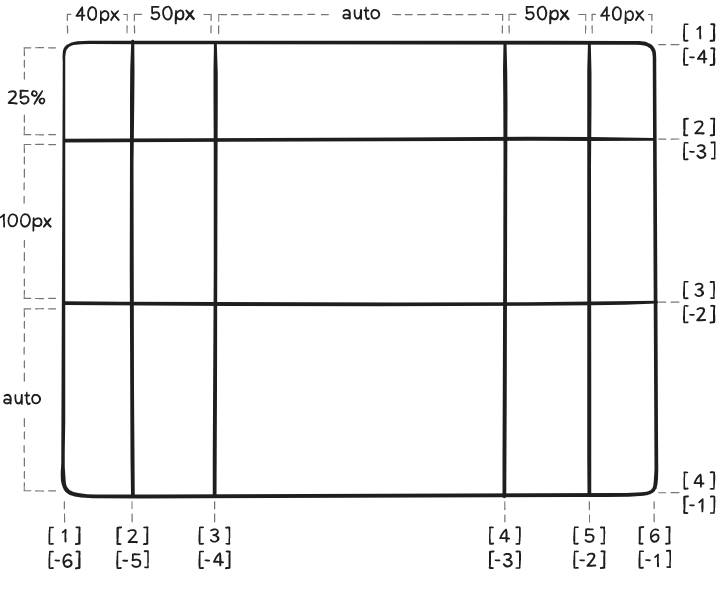
Fr单位：新的fr单位代表网格容器中可用空间的一等份。从而不需要按剩下空间除以几份得出百分比，可能出现除不尽的情况。

1. .container {
2. grid-template-columns: <track-size> ... | <line-name> <track-size> ...;
3. grid-template-rows: <track-size> ... | <line-name> <track-size> ...;
4. }

示例1：

不指定网格线的名称，只指定行列的数目以及大小

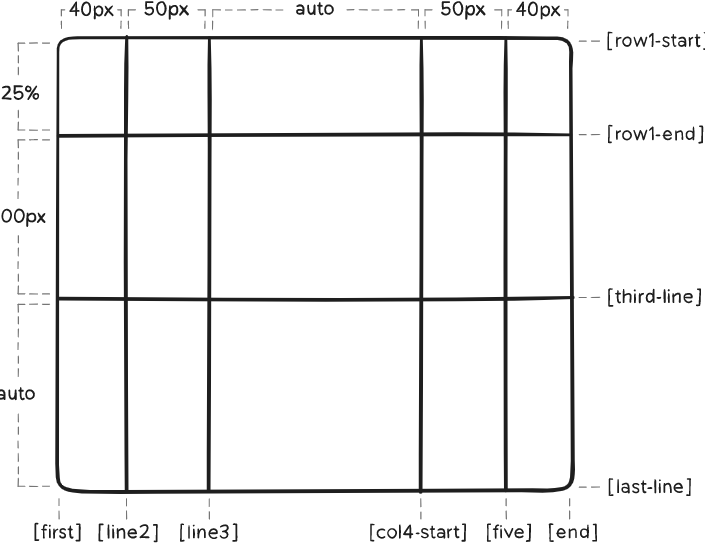
1. .container {
2. grid-template-columns: 40px 50px auto 50px 40px;
3. grid-template-rows: 25% 100px auto;
4. }



示例2：

指定行列数目以及大小，并且也指定网格线的名称

1. .container {
2. grid-template-columns: [first] 40px [line2] 50px [line3] auto [col4-start] 50px [five] 40px [end];
3. grid-template-rows: [row1-start] 25% [row1-end] 100px [third-line] auto [last-line];
4. }



🡪如果定义中包含多个重复值，则可以使用repeat(重复的次数， 重复的内容)表示法来简化定义

例如

1. .container {
2. grid-template-columns: repeat (3, 20px [col-start]);
3. }

将等价于

1. .container {
2. grid-template-columns: 20px [col-start] 20px [col-start] 20px [col-start];
3. }

🡪fr 单元允许你用等分网格容器剩余可用空间来设置 网格轨道 (Grid Track) 的大小

例如：

1. .container {
2. grid-template-columns: 1fr 50px 1fr 1fr;
3. }

在这个例子中，可用空间总量减去 50px 后，再给 fr 单元的值 3 等分

3.**grid-template-areas**

通过引用 [grid-area](https://www.html.cn/archives/8510/#prop-grid-area) 属性指定的 网格区域(Grid Area) 名称来定义网格模板。重复网格区域的名称导致内容跨越这些单元格。一个点号（.）代表一个空单元格。这个语法本身可视作网格的可视化结构。

值：

* <grid-area-name>：由网格项的 [grid-area](https://www.html.cn/archives/8510/#prop-grid-area) 指定的网格区域名称
* . （点号） ：代表一个空的网格单元
* none：不定义网格区域

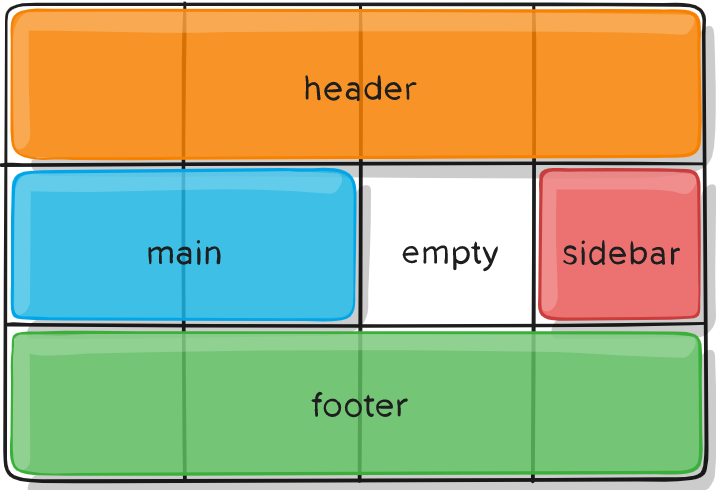
CSS 代码:

1. .container {
2. grid-template-areas:
3. "<grid-area-name> | . | none | ..."
4. "...";
5. }

例子：

1. .item-a {
2. grid-area: header;
3. }
4. .item-b {
5. grid-area: main;
6. }
7. .item-c {
8. grid-area: sidebar;
9. }
10. .item-d {
11. grid-area: footer;
12. }
14. .container {
15. grid-template-columns: 50px 50px 50px 50px;
16. grid-template-rows: auto;
17. grid-template-areas:
18. "header header header header"
19. "main main . sidebar"
20. "footer footer footer footer";
21. }

上面的代码将创建一个 4 列宽 3 行高的网格。整个顶行将由 **header** 区域组成。中间一排将由两个 **main** 区域，一个是空单元格，一个 **sidebar** 区域组成。最后一行全是 **footer** 区域组成。就如下图所示：



4. **grid-template**

用于定义[grid-template-rows](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-columns](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-areas](https://www.html.cn/archives/8510/#prop-grid-template-areas)的简写属性。

1. .container {
2. grid-template: none | <grid-template-rows> / <grid-template-columns>;
3. }

例子：

1. .container {
2. grid-template:
3. [row1-start] "header header header" 25px [row1-end]
4. [row2-start] "footer footer footer" 25px [row2-end]
5. / auto 50px auto;
6. }

这相当于分别[grid-template-rows](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-columns](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-areas](https://www.html.cn/archives/8510/#prop-grid-template-areas)配置语法下的：

1. .container {
2. grid-template-rows: [row1-start] 25px [row1-end row2-start] 25px [row2-end];
3. grid-template-columns: auto 50px auto;
4. grid-template-areas:
5. "header header header"
6. "footer footer footer";
7. }

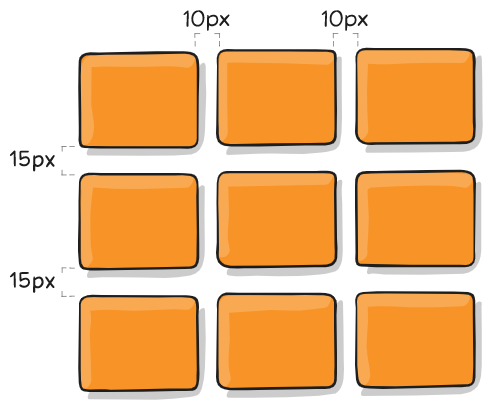
5.**grid-column-gap/grid-row-gap**

指定网格线(grid lines)的大小。你可以把它想象为设置列/行之间间距的宽度。

1. .container {
2. grid-column-gap: <line-size>;
3. grid-row-gap: <line-size>;
4. }

例子：

1. .container {
2. grid-template-columns: 100px 50px 100px;
3. grid-template-rows: 80px auto 80px;
4. grid-column-gap: 10px;
5. grid-row-gap: 15px;
6. }



6.**justify-items**

沿着 *inline*（行）轴线对齐网格项(grid items)（相反的属性是 [align-items](https://www.html.cn/archives/8510/#prop-align-items) 沿着 *block*（列）轴线对齐）。此值适用于容器内的所有网格项。

值：

* start：将网格项对齐到其单元格的左侧起始边缘（左侧对齐）
* end：将网格项对齐到其单元格的右侧结束边缘（右侧对齐）
* center：将网格项对齐到其单元格的水平中间位置（水平居中对齐）
* stretch：填满单元格的宽度（默认值）
* .container {
* justify-items: start | end | center | stretch;
* }

7**. align-items**

沿着 *block*（列）轴线对齐网格项(grid items)（相反的属性是 [justify-items](https://www.html.cn/archives/8510/#prop-justify-items) 沿着 *inline*（行）轴线对齐）。此值适用于容器内的所有网格项。

* 值：
* start：将网格项对齐到其单元格的顶部起始边缘（顶部对齐）
* end：将网格项对齐到其单元格的底部结束边缘（底部对齐）
* center：将网格项对齐到其单元格的垂直中间位置（垂直居中对齐）
* stretch：填满单元格的高度（默认值）
* .container {
* align-items: start | end | center | stretch;
* }

🡪这些行为也可以通过每个单独网格项(grid items) 的 [align-self](https://www.html.cn/archives/8510/#prop-align-self) 属性单独设置。

8.**place-items**

place-items 是设置 align-items 和 justify-items 的简写形式。

* 值：
* <align-items> <justify-items>：第一个值设置 align-items 属性，第二个值设置 justify-items 属性。如果省略第二个值，则将第一个值同时分配给这两个属性。

9.**justify-content**

有时，你的网格合计大小可能小于其 网格容器(grid container) 大小。 如果你的所有 网格项(grid items) 都使用像 px 这样的非灵活单位设置大小，就可能出现这种情况。在这种情况下，您可以设置网格容器内的网格的对齐方式。 此属性沿着 inline（行）轴线对齐网格（相反的属性是 [align-content](https://www.html.cn/archives/8510/#prop-align-content) ，沿着 block（列）轴线对齐网格）。

值：

* start：将网格对齐到 网格容器 (grid container) 的左侧起始边缘（左侧对齐）
* end：将网格对齐到 网格容器 的右侧结束边缘（右侧对齐）
* center：将网格对齐到 网格容器 的水平中间位置（水平居中对齐）
* stretch：调整 网格项 (grid items) 的宽度，允许该网格填充满整个 网格容器 的宽度
* space-around：在每个网格项之间放置一个均匀的空间，左右两端放置一半的空间
* space-between：在每个网格项之间放置一个均匀的空间，左右两端没有空间
* space-evenly：在每个网格项目之间放置一个均匀的空间，左右两端放置一个均匀的空间

10.**align-content**

有时，你的网格合计大小可能小于其 网格容器(grid container) 大小。 如果你的所有 网格项(grid items) 都使用像 px 这样的非灵活单位设置大小，就可能出现这种情况。在这种情况下，您可以设置网格容器内的网格的对齐方式。 此属性沿着 block（列）轴线对齐网格（相反的属性是 [justify-content](https://www.html.cn/archives/8510/#prop-justify-content) ，沿着 inline（行）轴线对齐网格）。

值：

start：将网格对齐到 网格容器 (grid container) 的顶部起始边缘（顶部对齐）

end：将网格对齐到 网格容器 的底部结束边缘（底部对齐）

center：将网格对齐到 网格容器 的垂直中间位置（垂直居中对齐）

stretch：调整 网格项 (grid items) 的高度，允许该网格填充满整个 网格容器 的高度

space-around：在每个网格项之间放置一个均匀的空间，上下两端放置一半的空间

space-between：在每个网格项之间放置一个均匀的空间，上下两端没有空间

space-evenly：在每个网格项目之间放置一个均匀的空间，上下两端放置一个均匀的空间

1. .container {
2. align-content: start | end | center | stretch | space-around | space-between | space-evenly;
3. }

11.**place-content**

place-content 是设置 align-content 和 justify-content 的简写形式。

值：

* <align-content> <justify-content>：第一个值设置 align-content 属性，第二个值设置 justify-content 属性。如果省略第二个值，则将第一个值同时分配给这两个属性。

12.**grid-auto-columns/grid-auto-rows**

指定任何自动生成的网格轨道(grid tracks)（又名隐式网格轨道）的大小。当网格中的网格项多于单元格时，或者当网格项位于显式网格之外时，就会创建隐式轨道。（参见[显式网格和隐式网格之间的区别](https://www.html.cn/archives/10327" \t "_blank)）

值：

<track-size>：可以是长度值，百分比，或者等份网格容器中可用空间的分数（使用 fr 单位）

1. .container {
2. grid-auto-columns: <track-size> ...;
3. grid-auto-rows: <track-size> ...;
4. }

13. **grid-auto-flow**

如果你有一些没有明确放置在网格上的网格项(grid items)，自动放置算法 会自动放置这些网格项。该属性控制自动布局算法如何工作。

值：

row：告诉自动布局算法依次填充每行，根据需要添加新行 （默认）

column：告诉自动布局算法依次填入每列，根据需要添加新列

dense：告诉自动布局算法在稍后出现较小的网格项时，尝试填充网格中较早的空缺 (dense 只会更改网格项的可视顺序，并可能导致它们出现乱序，这对可访问性不利)

14. **grid**

在一个声明中设置所有以下属性的简写： [grid-template-rows](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-columns](https://www.html.cn/archives/8510/#prop-grid-template-columns-rows), [grid-template-areas](https://www.html.cn/archives/8510/#prop-grid-template-areas), [grid-auto-rows](https://www.html.cn/archives/8510/#prop-grid-auto-columns-rows), [grid-auto-columns](https://www.html.cn/archives/8510/#prop-grid-auto-columns-rows), 和 [grid-auto-flow](https://www.html.cn/archives/8510/#prop-grid-auto-flow) 。（注意：您只能在单个网格声明中指定显式或隐式网格属性）。

**子元素 网格项(Grid Items) 属性**

1. **grid-column-start / grid-column-end / grid-row-start / grid-row-end**

通过引用特定网格线(grid lines) 来确定 网格项(grid item) 在网格内的位置。 [grid-column-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) / [grid-row-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) 是网格项开始的网格线，[grid-column-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) / [grid-row-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) 是网格项结束的网格线。

值：

<line> ：可以是一个数字引用一个编号的网格线，或者一个名字来引用一个命名的网格线

span <number> ：该网格项将跨越所提供的网格轨道数量

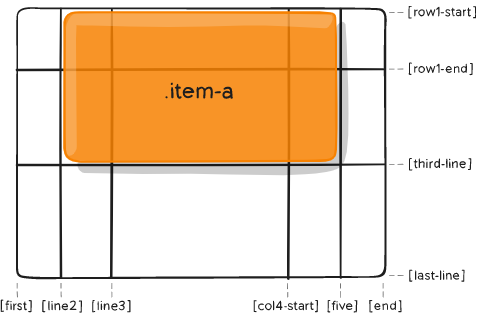
span <name> ：该网格项将跨越到它与提供的名称位置

auto：表示自动放置，自动跨度，默认会扩展一个网格轨道的宽度或者高度

1. .item {
2. grid-column-start: <number> | <name> | span <number> | span <name> | auto
3. grid-column-end: <number> | <name> | span <number> | span <name> | auto
4. grid-row-start: <number> | <name> | span <number> | span <name> | auto
5. grid-row-end: <number> | <name> | span <number> | span <name> | auto
6. }

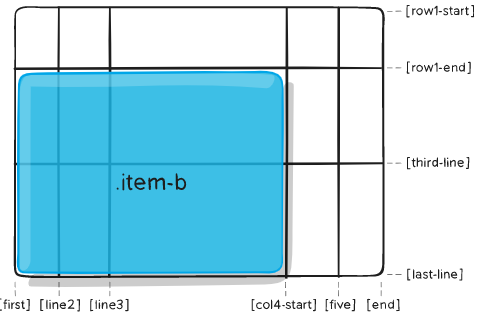
例子1：

1. .item-a {
2. grid-column-start: 2;
3. grid-column-end: five;
4. grid-row-start: row1-start
5. grid-row-end: 3;
6. }



例子2：通过span跨越网格

1. .item-b {
2. grid-column-start: 1;
3. grid-column-end: span col4-start;
4. grid-row-start: 2
5. grid-row-end: span 2
6. }



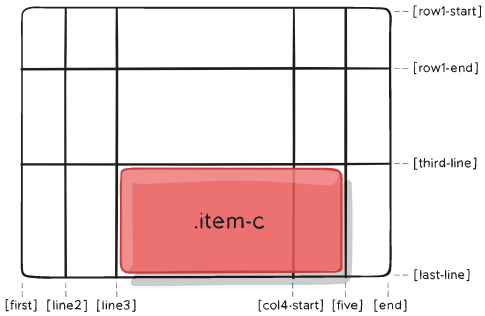
2. **grid-column/grid-row**

分别为 [grid-column-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) + [grid-column-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) 和 [grid-row-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) + [grid-row-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) 的简写形式。

1. .item {
2. grid-column: <start-line> / <end-line> | <start-line> / span <value>;
3. grid-row: <start-line> / <end-line> | <start-line> / span <value>;
4. }

例子：

1. .item-c {
2. grid-column: 3 / span 2;
3. grid-row: third-line / 4;
4. }



**3. grid-area**

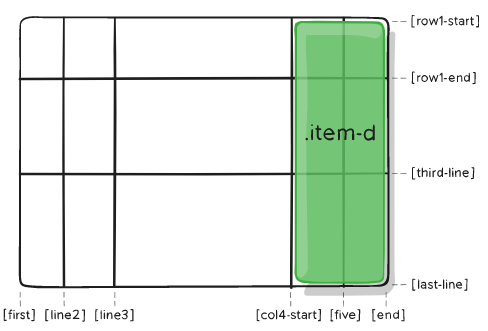
为网格项提供一个名称，以便可以 被使用网格容器 [grid-template-areas](https://www.html.cn/archives/8510/#prop-grid-template-areas) 属性创建的模板进行引用。 另外，这个属性可以用作[grid-row-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) + [grid-column-start](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) + [grid-row-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) + [grid-column-end](https://www.html.cn/archives/8510/#prop-grid-column-row-start-end) 的简写。

值：

* <name>：你所选的名称
* <row-start> / <column-start> / <row-end> / <column-end>：数字或分隔线名称
* .item {
* grid-area: <name> | <row-start> / <column-start> / <row-end> / <column-end>;
* }

例子：

1. .item-d {
2. grid-area: 1 / col4-start / last-line / 6
3. }



**4.justify-self (单独为此网格项设置)**

* 沿着 inline（行）轴线对齐网格项（ 相反的属性是 [align-self](https://www.html.cn/archives/8510/#prop-align-self) ，沿着 block（列）轴线对齐）。此值适用于单个网格项内的内容。
* 值：
* start：将网格项对齐到其单元格的左侧起始边缘（左侧对齐）
* end：将网格项对齐到其单元格的右侧结束边缘（右侧对齐）
* center：将网格项对齐到其单元格的水平中间位置（水平居中对齐）
* stretch：填满单元格的宽度（默认值）

5.**align-self** **(单独为此网格项设置)**

* 沿着 block（列）轴线对齐网格项(grid items)（ 相反的属性是 [justify-self](https://www.html.cn/archives/8510/#prop-justify-self) ，沿着 inline（行）轴线对齐）。此值适用于单个网格项内的内容。
* 值：
* start：将网格项对齐到其单元格的顶部起始边缘（顶部对齐）
* end：将网格项对齐到其单元格的底部结束边缘（底部对齐）
* center：将网格项对齐到其单元格的垂直中间位置（垂直居中对齐）
* stretch：填满单元格的高度（默认值）

6. **place-self** **(单独为此网格项设置)**

place-self 是设置 align-self 和 justify-self 的简写形式。