## 

[一、概述](#_8zt950aij30e)

[二、基本概念](#_rtszjt1vzy69)

[2.1 容器和项目](#_m5jroxkszm7)

[2.2 行和列](#_c9oq70hec4xp)

[2.3 单元格](#_74ii3tls5wzo)

[2.4 网格线](#_hvz3enckwwl)

[三、容器属性](#_a6hu3tnlf5n)

[3.1 display 属性](#_6ka4cl7tk5o6)

[3.2 grid-template-columns 属性，grid-template-rows 属性](#_72hvn6brht1u)

[（1）repeat()](#_ttp8o1lfhpw4)

[（2）auto-fill 关键字](#_o3er9py2gdg8)

[（3）fr 关键字](#_wqixdng1mfj4)

[（4）minmax()](#_jzkqyxfxjb97)

[(5）auto 关键字](#_qnssa2uk31is)

[(6）网格线的名称](#_duae80uz03qu)

[（7）布局实例](#_x475yhbilm3j)

[3.3 grid-row-gap 属性，grid-column-gap 属性，grid-gap 属性](#_aw7z4xga05ki)

[3.4 grid-template-areas](#_v0ilsh26xm0q)

[3.5 grid-auto-flow](#_51qubbw9gpbb)

[3.6 justify-items ，align-items ，place-items](#_f5i3go54e4fo)

[3.7 justify-content ，align-content ，place-content](#_ozlyxzv7mq03)

[3.8 grid-auto-columns ，grid-auto-rows](#_68ikcyma1qyu)

[3.9 grid-template ，grid](#_kmgtasi5jjtt)

[四、项目属性](#_2yo2t8syb71j)

[4.1 grid-column-start ，grid-column-end ，grid-row-start ，grid-row-end](#_jkpaadwecmvr)

[4.2 grid-column ，grid-row](#_wzbxdijuoein)

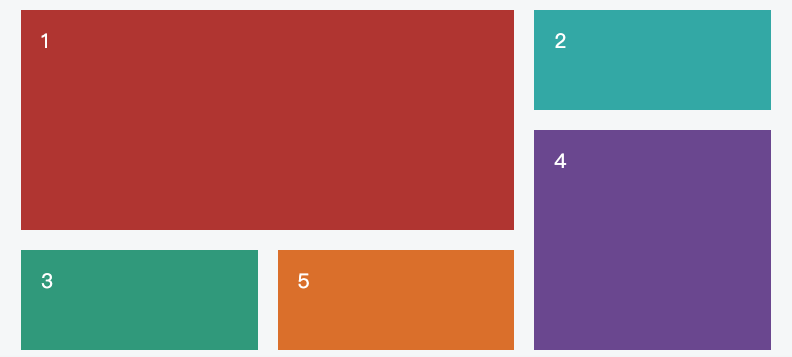
[4.3 grid-area](#_rgv6a3pkor8m)

[4.4 justify-self ，align-self ，place-self](#_l84maif11htb)

## 一、概述

网格布局（Grid）是最强大的 CSS 布局方案。

它将网页划分成一个个网格，可以任意组合不同的网格，做出各种各样的布局。以前，只能通过复杂的 CSS 框架达到的效果，现在浏览器内置了。



上图这样的布局，就是 Grid 布局的拿手好戏。

Grid 布局与 [Flex 布局](http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html)有一定的相似性，都可以指定容器内部多个项目的位置。但是，它们也存在重大区别。

Flex 布局是轴线布局，只能指定"项目"针对轴线的位置，可以看作是一维布局。Grid 布局则是将容器划分成"行"和"列"，产生单元格，然后指定"项目所在"的单元格，可以看作是二维布局。Grid 布局远比 Flex 布局强大。

## 二、基本概念

### 2.1 容器和项目

采用网格布局的区域，称为**"容器"（container）**。容器内部采用网格定位的子元素，称为**"项目"（item）**。注意：项目只能是容器的顶层子元素，不包含项目的子元素，比如上面代码的<p>元素就不是项目。Grid 布局只对项目生效。

### 2.2 行和列

容器里面的水平区域称为**"行"（row**），垂直区域称为**"列"（column）**。

### 2.3 单元格

行和列的交叉区域，称为**"单元格"（cell）**。

正常情况下，n行和m列会产生n x m个单元格。比如，3行3列会产生9个单元格。

### 2.4 网格线

划分网格的线，称为**"网格线"（grid line）**。水平网格线划分出行，垂直网格线划分出列。

正常情况下，n行有n + 1根水平网格线，m列有m + 1根垂直网格线，比如三行就有四根水平网格线。

## 三、容器属性

Grid 布局的属性分成两类。一类定义在**容器上面，称为容器属性**；另一类定义在**项目上面，称为项目属性**。这部分先介绍容器属性。

### 3.1 display 属性

display: grid指定一个容器采用网格布局。

div {

display: **grid;**

}

默认情况下，容器元素都是块级元素，但也可以设成行内元素。

div {

display: **inline-grid;**

}

***注意，设为网格布局以后，容器子元素（项目）的float、display: inline-block、display: table-cell、vertical-align和column-\*等设置都将失效。***

### 3.2 grid-template-columns 属性，grid-template-rows 属性

容器指定了网格布局以后，接着就要划分行和列。

grid-template-columns属性定义每一列的**列宽**，

grid-template-rows属性定义每一行的**行高**。

.container {

display: grid;

grid-template-columns: 100px 100px 100px;

grid-template-rows: 100px 100px 100px;

}

[上面代码](https://jsbin.com/qiginur/edit?css,output)指定了一个三行三列的网格，列宽和行高都是100px。

除了使用绝对单位，也可以使用**百分比**。

.container {

display: grid;

grid-template-columns: 33.33% 33.33% 33.33%;

grid-template-rows: 33.33% 33.33% 33.33%;

}

#### 

#### （1）repeat()

有时候，重复写同样的值非常麻烦，尤其网格很多时。这时，可以使用repeat()函数，简化重复的值。上面的代码用repeat()改写如下。

.container {

display: grid;

grid-template-columns: **repeat(3, 33.33%)**;

grid-template-rows: **repeat(3, 33.33%)**;

}

repeat()接受两个参数，**第一个参数是重复的次数**（上例是3），**第二个参数是所要重复的值。**

repeat()**重复某种模式**也是可以的。

grid-template-columns: repeat(2, 100px 20px 80px);

[上面代码](https://jsbin.com/cokohu/edit?css,output)定义了6列，第一列和第四列的宽度为100px，第二列和第五列为20px，第三列和第六列为80px。

#### 

#### （2）auto-fill 关键字

有时，单元格的大小是固定的，但是容器的大小不确定。如果希望每一行（或每一列）容纳尽可能多的单元格，这时可以使用auto-fill关键字表示自动填充。

.container {

display: grid;

grid-template-columns: **repeat(auto-fill, 100px)**;

}

[上面代码](https://jsbin.com/himoku/edit?css,output)表示每列宽度100px，然后自动填充，直到容器不能放置更多的列。

#### 

#### （3）fr 关键字

为了方便表示**比例关系**，网格布局提供了fr关键字（fraction 的缩写，意为"片段"）。如果两列的宽度分别为**1fr和2fr，就表示后者是前者的两倍**。

.container {

display: grid;

grid-template-columns: 1fr 1fr;

}

fr可以与绝对长度的单位结合使用，这时会非常方便。

.container {

display: grid;

grid-template-columns: **150px 1fr 2fr;**

}

上面代码表示，第一列的宽度为150像素，第二列的宽度是第三列的一半。

#### （4）minmax()

minmax()函数产生一个长度范围，表示长度就在这个范围之中。它接受两个参数，分别为最小值和最大值。

grid-template-columns: 1fr 1fr **minmax(100px, 1fr);**

上面代码中，minmax(100px, 1fr)表示列宽不小于100px，不大于1fr。

#### (5）auto 关键字

auto关键字表示由浏览器**自己决定长度**。

grid-template-columns: 100px auto 100px;

上面代码中，第二列的宽度，基本上等于该列单元格的最大宽度，除非单元格内容设置了min-width，且这个值大于最大宽度。

#### (6）网格线的名称

grid-template-columns属性和grid-template-rows属性里面，还可以使用方括号，指定每一根网格线的名字，方便以后的引用。

.container {

display: grid;

grid-template-columns: [c1] 100px [c2] 100px [c3] auto [c4];

grid-template-rows: [r1] 100px [r2] 100px [r3] auto [r4];

}

上面代码指定网格布局为3行 x 3列，因此有4根垂直网格线和4根水平网格线。方括号里面依次是这八根线的名字。

网格布局允许同一根线有多个名字，比如[fifth-line row-5]。

#### （7）布局实例

grid-template-columns属性对于网页布局非常有用。两栏式布局只需要一行代码。

.wrapper {

display: grid;

grid-template-columns: 70% 30%;}

上面代码将左边栏设为70%，右边栏设为30%。

传统的十二网格布局，写起来也很容易。

grid-template-columns: repeat(12, 1fr);

### 

### 3.3 grid-row-gap 属性，grid-column-gap 属性，grid-gap 属性

grid-row-gap属性设置行与行的间隔（**行间距**），

grid-column-gap属性设置列与列的间隔（**列间距**）。

.container {

grid-row-gap: 20px;

grid-column-gap: 20px;

}

上面一段 CSS 代码等同于下面的代码。

.container {

grid-gap: 20px 20px;}

### 

### 3.4 grid-template-areas

网格布局允许指定**"区域"（area）**，一个区域由单个或多个单元格组成。

grid-template-areas属性用于定义区域。

.container {

display: grid;

grid-template-columns: 100px 100px 100px;

grid-template-rows: 100px 100px 100px;

grid-template-areas: 'a b c'

'd e f'

'g h i';

}

多个单元格合并成一个区域的写法如下。

grid-template-areas: 'a a a'

'b b b'

'c c c';

下面是一个布局实例。

grid-template-areas: "header header header"

"main main sidebar"

"footer footer footer";

上面代码中，**顶部是页眉区域header**，**底部是页脚区域footer**，中间部分则为main和sidebar。

**如果某些区域不需要利用，则使用"点"（.）表示。**

grid-template-areas: 'a . c'

'd . f'

'g . i';

上面代码中，中间一列为点，表示没有用到该单元格，或者该单元格不属于任何区域。

***注意，区域的命名会影响到网格线。每个区域的起始网格线，会自动命名为区域名-start，终止网格线自动命名为区域名-end。***

***比如，区域名为header，则起始位置的水平网格线和垂直网格线叫做header-start，终止位置的水平网格线和垂直网格线叫做header-end。***

### 3.5 grid-auto-flow

划分网格以后，容器的子元素会按照顺序，自动放置在每一个网格。默认的放置顺序是"先行后列"，即先填满第一行，再开始放入第二行，即下图数字的顺序。

这个顺序由grid-auto-flow属性决定，默认值是row，即"先行后列"。也可以将它设成column，变成"先列后行"。

**grid-auto-flow: column;**

**grid-auto-flow属性除了设置成row和column，还可以设成row dense和column dense**。这两个值主要用于，某些项目指定位置以后，剩下的项目怎么自动放置。尽可能紧密填满，尽量不出现空格。

### 3.6 justify-items ，align-items ，place-items

justify-items属性设置**单元格内容的水平位置（左中右）**，

align-items属性设置**单元格内容的垂直位置（上中下）**。

.container {

justify-items: start | end | center | stretch;

align-items: start | end | center | stretch;}

### 

### 3.7 justify-content ，align-content ，place-content

justify-content属性是整个**内容区域**在容器里面的**水平位置**（左中右），

align-content属性是整个**内容区域**的**垂直位置**（上中下）。

.container {

justify-content: start | end | center | stretch | space-around | space-between | space-evenly;

align-content: start | end | center | stretch | space-around | space-between | space-evenly; }

* **stretch** - 项目大小没有指定时，**拉伸占据整个**网格容器。
* **space-around** - **每个项目两侧的间隔相等。**所以，项目之间的间隔比项目与容器边框的间隔大一倍。
* **space-between** - **项目与项目的间隔相等，**项目与容器边框之间没有间隔。
* **space-evenly** - 项目与项目的间隔相等，**项目与容器边框之间也是同样长度的间隔**。

### 

### 3.8 grid-auto-columns ，grid-auto-rows

自动生成多余的网格，以便放置项目。

grid-auto-columns属性和grid-auto-rows属性用来设置，浏览器自动创建的多余网格的列宽和行高。它们的写法与grid-template-columns和grid-template-rows完全相同。如果不指定这两个属性，浏览器完全根据单元格内容的大小，决定新增网格的列宽和行高。

.container {

display: grid;

grid-template-columns: 100px 100px 100px;

grid-template-rows: 100px 100px 100px;

grid-auto-rows: 50px;

}

上面代码指定新增的行高统一为50px（原始的行高为100px）。

### 3.9 grid-template ，grid

grid-template属性是grid-template-columns、grid-template-rows和grid-template-areas这三个属性的合并简写形式。

grid属性是grid-template-rows、grid-template-columns、grid-template-areas、grid-auto-rows、grid-auto-columns、grid-auto-flow这六个属性的合并简写形式。

**从易读易写的角度考虑，还是建议不要合并属性，所以这里就不详细介绍这两个属性了。**

## 四、项目属性

### 4.1 grid-column-start ，grid-column-end ，grid-row-start ，grid-row-end

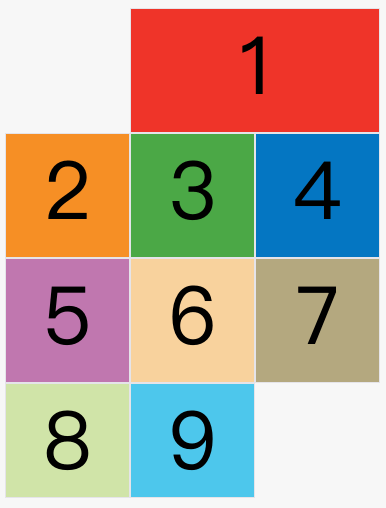
**项目的位置是可以指定的**，具体方法就是指定项目的四个边框，分别定位在哪根网格线。

* grid-column-start属性：左边框所在的垂直网格线
* grid-column-end属性：右边框所在的垂直网格线
* grid-row-start属性：上边框所在的水平网格线
* grid-row-end属性：下边框所在的水平网格线

.item-1 {

grid-column-start: 2;

grid-column-end: 4;}

[上面代码](https://jsbin.com/yukobuf/edit?css,output)指定，1号项目的左边框是第二根垂直网格线，右边框是第四根垂直网格线。

这四个属性的值还可以使用**span关键字，表示"跨越"**，即左右边框（上下边框）之间**跨越多少个网格。**

.item-1 {

grid-column-start: span 2;

}

[上面代码](https://jsbin.com/hehumay/edit?html,css,output)表示，1号项目的左边框距离右边框跨越2个网格。

使用这四个属性，如果产生了项目的重叠，则使用z-index属性指定项目的重叠顺序。

### 4.2 grid-column ，grid-row

grid-column属性是grid-column-start和grid-column-end的合并简写形式，

grid-row属性是grid-row-start属性和grid-row-end的合并简写形式。

.item {

grid-column: / ;

grid-row: / ;

}

下面是一个例子。

.item-1 {

grid-column: 1 / 3;

grid-row: 1 / 2;

}

/\* 等同于 \*/

.item-1 {

grid-column-start: 1;

grid-column-end: 3;

grid-row-start: 1;

grid-row-end: 2;}

上面代码中，项目item-1占据第一行，从第一根列线到第三根列线。

这两个属性之中，也可以使用span关键字，表示跨越多少个网格。

.item-1 {

background: #b03532;

grid-column: 1 / 3;

grid-row: 1 / 3;

}

/\* 等同于 \*/

.item-1 {

background: #b03532;

grid-column: 1 / span 2;

grid-row: 1 / span 2;

}

[上面代码](https://jsbin.com/volugow/edit?html,css,output)中，项目item-1占据的区域，包括第一行 + 第二行、第一列 + 第二列。

***斜杠以及后面的部分可以省略，默认跨越一个网格。***

### 4.3 grid-area

grid-area属性指定项目放在哪一个区域。

### 4.4 justify-self ，align-self ，place-self

**justify-self属性**设置单元格内容的水平位置（左中右），跟justify-items属性的用法完全一致，但只作用于**单个项目**。

**align-self属性**设置单元格内容的垂直位置（上中下），跟align-items属性的用法完全一致，也是只作用于**单个项目**。