## Table表格布局

DIV+CSS 布局

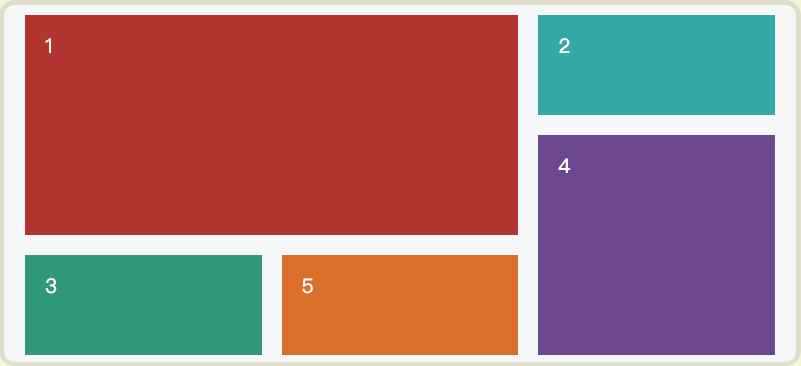
弹性盒布局

## 网格布局

## Grid网格布局

### 基本介绍

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。



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。

### 一、 基本概念

#### ·**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）。项目只能是容器的顶层子元素，不包含项目的子元素。

#### ·行和列

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

#### ·**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。

#### ·**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。

### 二、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。

#### 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，容器元素是块级元素，但也可以设置成行内元素。

div{

display:inline-grid;

}

#### 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。接下来做个3行3列的网格。

<style>

.box{

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

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

}

.box{

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

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

}

行和列简写属性：

 grid-template: repeat(3,1fr)/1fr 2fr 1fr;

grid-template:rows / columns;

值也可以是以下几种情况：

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。接受两个参数，第一个参数是重复的次数，第二个参数是所要重复的值。

上面的代码用repeat(),改写如下：

.box{

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

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

}

2)、fr

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。

.box{

grid-template-columns:1fr 1fr 2fr ;

grid-template-rows: 1fr 1fr 1fr;

}

3)、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。minmax(100px, 1fr)表示列宽不小于100px，不大于1fr。

.box{

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

grid-template-rows:1fr 1fr 1fr;

}

4)、auto关键字

auto关键字表示由浏览器自己决定长度。

.box{

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

grid-template-rows:1fr 1fr 1fr;

}

5)、网格线的名称

.box{

height: 650px;

display: grid;

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

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

}

#### 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）

grid-gap属性是grid-column-gap和grid-row-gap的合并简写形式

**语法**：

.box{

grid-gap: grid-row-gap grid-column-gap

}

#### 4.grid-template-areas 属性

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。

.box{

grid-template-areas: 'a b c'

'd e f'

'g h i';

}

上面代码先划分出9个单元格，然后将其定名为a到i的九个区域，分别对应这九个单元格。多个单元格合并成一个区域的写法如下。

.box{

grid-template-areas: 'a a a'

'b b b'

'c c c';

}

#### 5.justify-items 属性，align-items 属性，place-items 属性

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）。

**语法**：

.box{

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

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

}

**这两个属性的写法完全相同，可以取下面这些值都。**

● start：对齐单元格的起始边缘。

● end：对齐单元格的结束边缘。

● center：单元格内部居中。

● stretch：拉伸，占满单元格的整个宽度（默认值）。

● place-items属性是align-items属性和justify-items属性的合并简写形式。

语法：

.box{

place-items: align-items justify-items;

}

#### 6.justify-content 属性，align-content 属性，place-content 属性

内容区域在容器里面的水平和垂直位置，简写形式 place-content。

**语法**：

.box{

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

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

}

**有可能出现的值**

● start：对齐容器的起始边框。

● end：对齐容器的结束边框。

● center：容器内部居中。

● 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。

● place-content属性是align-content属性和justify-content属性的合并简写形式。

**语法**：

.box{

place-content: align-content justify-content

}

#### 7.grid-auto-flow属性

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。

语法:

.box{

grid-auto-flow: row | column | row dense | column dense;

}

**可能出现的值：**

● row：默认值，"先行后列"，即先放满第一行，再开始放入第二行。

● column："先列后行"。

● row dense：表示"先行后列"，并且尽可能紧密放满，尽量不出现空格。

● column dense："先列后行"，尽量不出现空格。

### 三、项目属性

#### 1.grid-column-start 属性，grid-column-end 属性，grid-row-start 属性，grid-row-end 属性

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。

比如将第一个项目指定四个边框位置。

代码实现：

.item1{

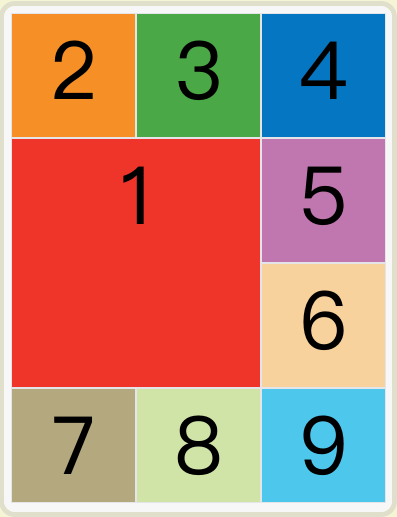
grid-column-start: 1;

grid-column-end: 3;

grid-row-start: 2;

grid-row-end: 4;

}



图示19-9 项目位置可以指定任意位置示例

除了1号项目以外，其他项目都没有指定位置，由浏览器自动布局，这时它们的位置由容器的grid-auto-flow属性决定，这个属性的默认值是row，因此会"先行后列"进行排列。读者可以把这个属性的值分别改成column、row dense和column dense，看看其他项目的位置发生了怎样的变化。

#### 2.grid-column 属性grid-row 属性

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。

**语法**：

.item{

grid-column: / ;

grid-row: / ;

}

如果想实现项目part1占据第一行，从第一根列线到第三根列线。

代码实现：

.item1 {

grid-column: 1 / 3;

grid-row: 1 / 2;

}

/\* 等同于 \*/

.item1 {

grid-column-start: 1;

grid-column-end: 3;

grid-row-start: 1;

grid-row-end: 2;

}

#### 3.grid-area 属性

grid-area属性指定项目放在哪一个区域。

代码实现：

.item1 {

grid-area: e;

}

#### 4.justify-self 属性，align-self 属性，place-self 属性

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。

**语法：**

.item{

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

align-self: start | end | center | stretch;

}

**这两个属性都可以取下面四个值。**

● start：对齐单元格的起始边缘。

● end：对齐单元格的结束边缘。

● center：单元格内部居中。

● stretch：拉伸，占满单元格的整个宽度（默认值）。

● place-self属性是align-self属性和justify-self属性的合并简写形式。

语法：

.item{

place-self: align-self justify-self;

}