HTML | 复制代码

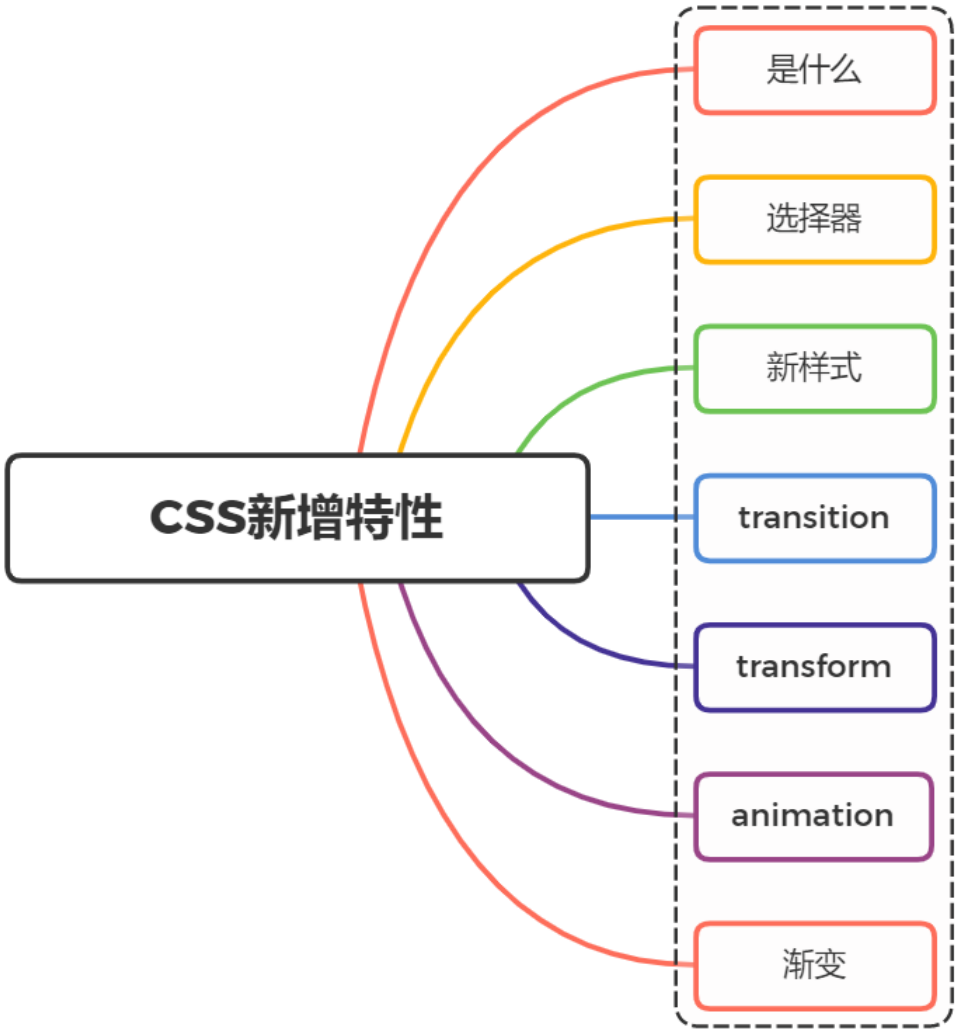
```html
<style>
    html {
        overflow: hidden;
        height: 100%
    }

    body {
        /* 视差元素的父级需要3D视角 */
        perspective: 1px;
        transform-style: preserve-3d;
        height: 100%;
        overflow-y: scroll;
        overflow-x: hidden;
    }
    #app{
        width: 100vw;
        height:200vh;
        background:skyblue;
        padding-top:100px;
    }
    .one{
        width:500px;
        height:200px;
        background:#409eff;
        transform: translateZ(0px);
        margin-bottom: 50px;
    }
    .two{
        width:500px;
        height:200px;
        background:#67c23a;
        transform: translateZ(-1px);
        margin-bottom: 150px;
    }
    .three{
        width:500px;
        height:200px;
        background:#e6a23c;
        transform: translateZ(-2px);
        margin-bottom: 150px;
    }
</style>
<div id="app">
    <div class="one">one</div>
    <div class="two">two</div>
```

```
        <div class="three">three</div>
    </div>
```

而这种方式实现视觉差动的原理如下：

- 容器设置上 transform-style: preserve-3d 和 perspective: xpx，那么处于这个容器的子元素就将位于3D空间中，
- 子元素设置不同的 transform: translateZ()，这个时候，不同元素在 3D Z轴方向距离屏幕（我们的眼睛）的距离也就不一样
- 滚动滚动条，由于子元素设置了不同的 transform: translateZ()，那么他们滚动的上下距离 translateY 相对屏幕（我们的眼睛），也是不一样的，这就达到了滚动视差的效果

11. CSS3新增了哪些新特性?



11.1. 是什么

`css`，即层叠样式表（Cascading Style Sheets）的简称，是一种标记语言，由浏览器解释执行用来使页面变得更美观

`css3` 是 `css` 的最新标准，是向后兼容的，`CSS1/2` 的特性在 `CSS3` 里都是可以使用的

而 `CSS3` 也增加了很多新特性，为开发带来了更佳的开发体验

11.2. 选择器

`css3` 中新增了一些选择器，主要为如下图所示：

选择器	例子	例子描述
element1~element2	p~ul	选择前面有 <p> 元素的每个 元素。
[attribute^=value]	a[src^="https"]	选择其 src 属性值以 "https" 开头的每个 <a> 元素。
[attribute$=value]	a[src$=".pdf"]	选择其 src 属性以 ".pdf" 结尾的所有 <a> 元素。
[attribute=value]*	a[src*="abc"]	选择其 src 属性中包含 "abc" 子串的每个 <a> 元素。
:first-of-type	p:first-of-type	选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
:last-of-type	p:last-of-type	选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
:only-of-type	p:only-of-type	选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
:only-child	p:only-child	选择属于其父元素的唯一子元素的每个 <p> 元素。
:nth-child(*n*)	p:nth-child(2)	选择属于其父元素的第二个子元素的每个 <p> 元素。
:nth-last-child(*n*)	p:nth-last-child(2)	同上，从最后一个子元素开始计数。
:nth-of-type(*n*)	p:nth-of-type(2)	选择属于其父元素第二个 <p> 元素的每个 <p> 元素。
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	同上，但是从最后一个子元素开始计数。
:last-child	p:last-child	选择属于其父元素最后一个子元素每个 <p> 元素。

11.3. 新样式

11.3.1. 边框

`css3` 新增了三个边框属性，分别是：

- border-radius：创建圆角边框
- box-shadow：为元素添加阴影

- border-image：使用图片来绘制边框

11.3.1.1. box-shadow

设置元素阴影，设置属性如下：

- 水平阴影
- 垂直阴影
- 模糊距离(虚实)
- 阴影尺寸(影子大小)
- 阴影颜色
- 内/外阴影

其中水平阴影和垂直阴影是必须设置的

11.3.2. 背景

新增了几个关于背景的属性，分别是 `background-clip` 、 `background-origin` 、 `background-size` 和 `background-break`

11.3.2.1. background-clip

用于确定背景画区，有以下几种可能的属性：

- background-clip: border-box; 背景从border开始显示
- background-clip: padding-box; 背景从padding开始显示
- background-clip: content-box; 背景显content区域开始显示
- background-clip: no-clip; 默认属性，等同于border-box

通常情况，背景都是覆盖整个元素的，利用这个属性可以设定背景颜色或图片的覆盖范围

11.3.2.2. background-origin

当我们设置背景图片时，图片是会以左上角对齐，但是是以 `border` 的左上角对齐还是以 `padding` 的左上角或者 `content` 的左上角对齐？ `border-origin` 正是用来设置这个的

- background-origin: border-box; 从border开始计算background-position
- background-origin: padding-box; 从padding开始计算background-position
- background-origin: content-box; 从content开始计算background-position

默认情况是 `padding-box` ，即以 `padding` 的左上角为原点

11.3.2.3. background-size

background-size属性常用来调整背景图片的大小，主要用于设定图片本身。有以下可能的属性：

- background-size: contain; 缩小图片以适合元素（维持像素长宽比）
- background-size: cover; 扩展元素以填补元素（维持像素长宽比）
- background-size: 100px 100px; 缩小图片至指定的大小
- background-size: 50% 100%; 缩小图片至指定的大小，百分比是相对包 含元素的尺寸

11.3.3. background-break

元素可以被分成几个独立的盒子（如使内联元素span跨越多行），`background-break` 属性用来控制背景怎样在这些不同的盒子中显示

- background-break: continuous; 默认值。忽略盒之间的距离（也就是像元素没有分成多个盒子，依然是一个整体一样）
- background-break: bounding-box; 把盒之间的距离计算在内；
- background-break: each-box; 为每个盒子单独重绘背景

11.3.4. 文字

11.3.5. word-wrap

语法：`word-wrap: normal|break-word`

- normal：使用浏览器默认的换行
- break-all：允许在单词内换行

11.3.6. text-overflow

`text-overflow` 设置或检索当当前行超过指定容器的边界时如何显示，属性有两个值选择：

- clip：修剪文本
- ellipsis：显示省略符号来代表被修剪的文本

11.3.7. text-shadow

`text-shadow` 可向文本应用阴影。能够规定水平阴影、垂直阴影、模糊距离，以及阴影的颜色

11.3.8. text-decoration

CSS3里面开始支持对文字的更深层次的渲染，具体有三个属性可供设置：

- text-fill-color: 设置文字内部填充颜色
- text-stroke-color: 设置文字边界填充颜色
- text-stroke-width: 设置文字边界宽度

11.3.9. 颜色

`css3` 新增了新的颜色表示方式 `rgba` 与 `hsla`

- rgba分为两部分，rgb为颜色值，a为透明度
- hala分为四部分，h为色相，s为饱和度，l为亮度，a为透明度

11.4. transition 过渡

`transition` 属性可以被指定为一个或多个 `CSS` 属性的过渡效果，多个属性之间用逗号进行分隔，必须规定两项内容：

- 过度效果
- 持续时间

语法如下：

CSS | 复制代码

```
transition:  CSS属性，花费时间，效果曲线(默认ease)，延迟时间(默认0)
```

上面为简写模式，也可以分开写各个属性

CSS | 复制代码

```
transition-property: width;
transition-duration: 1s;
transition-timing-function: linear;
transition-delay: 2s;
```

11.4.1. transform 转换

`transform` 属性允许你旋转，缩放，倾斜或平移给定元素

`transform-origin`：转换元素的位置（围绕那个点进行转换），默认值为`(x,y,z):(50%,50%,0)`

使用方式：

- transform: translate(120px, 50%)：位移
- transform: scale(2, 0.5)：缩放
- transform: rotate(0.5turn)：旋转
- transform: skew(30deg, 20deg)：倾斜

11.4.2. animation 动画

动画这个平常用的也很多，主要是做一个预设的动画。和一些页面交互的动画效果，结果和过渡应该一样，让页面不会那么生硬

animation也有很多的属性

- animation-name：动画名称
- animation-duration：动画持续时间
- animation-timing-function：动画时间函数
- animation-delay：动画延迟时间
- animation-iteration-count：动画执行次数，可以设置为一个整数，也可以设置为infinite，意思是无限循环
- animation-direction：动画执行方向
- animation-paly-state：动画播放状态
- animation-fill-mode：动画填充模式

11.5. 渐变

颜色渐变是指在两个颜色之间平稳的过渡， `css3` 渐变包括

- linear-gradient：线性渐变

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

- radial-gradient：径向渐变

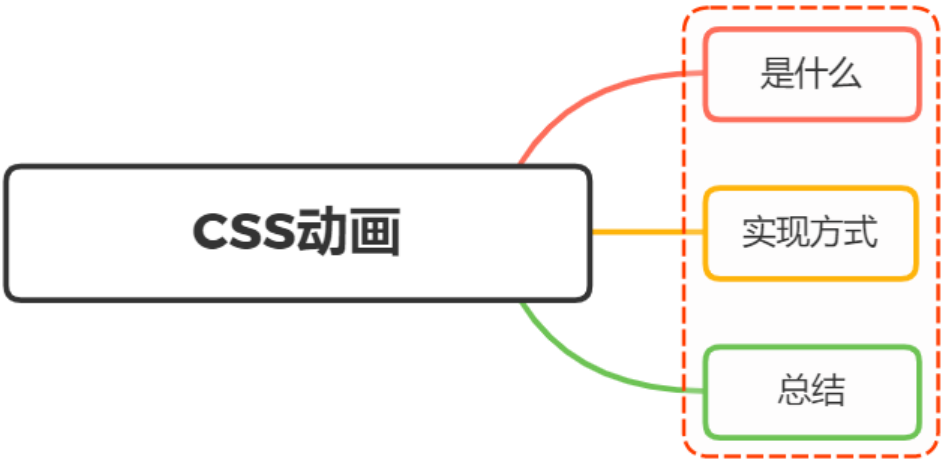
```
linear-gradient(0deg, red, green);
```

11.6. 其他

关于 `css3` 其他的新特性还包括 `flex` 弹性布局、 `Grid` 栅格布局，这两个布局在以前就已经讲过，这里就不再展示

除此之外，还包括多列布局、媒体查询、混合模式等等.....

12. css3动画有哪些?



12.1. 是什么

CSS动画（CSS Animations）是为层叠样式表建议的允许可扩展标记语言（XML）元素使用CSS的动画的模块

即指元素从一种样式逐渐过渡为另一种样式的过程

常见的动画效果有很多，如平移、旋转、缩放等等，复杂动画则是多个简单动画的组合

`css` 实现动画的方式，有如下几种：

- transition 实现渐变动画
- transform 转变动画
- animation 实现自定义动画

12.2. 实现方式

12.2.1. transition 实现渐变动画

`transition` 的属性如下：

- property:填写需要变化的css属性
- duration:完成过渡效果需要的时间单位(s或者ms)
- timing-function:完成效果的速度曲线
- delay: 动画效果的延迟触发时间

其中 `timing-function` 的值有如下：

值	描述
linear	匀速（等于 cubic-bezier(0,0,1,1)）
ease	从慢到快再到慢（cubic-bezier(0.25,0.1,0.25,1)）
ease-in	慢慢变快（等于 cubic-bezier(0.42,0,1,1)）
ease-out	慢慢变慢（等于 cubic-bezier(0,0,0.58,1)）
ease-in-out	先变快再到慢（等于 cubic-bezier(0.42,0,0.58,1)），渐显渐隐效果
cubic-bezier(*n*,*n*,*n*,*n*)	在 cubic-bezier 函数中定义自己的值。可能的值是 0 至 1 之间的数值

注意：并不是所有的属性都能使用过渡的，如 `display:none<->display:block`

举个例子，实现鼠标移动上去发生变化动画效果

HTML | 复制代码

```html
<style>
        .base {
            width: 100px;
            height: 100px;
            display: inline-block;
            background-color: #0EA9FF;
            border-width: 5px;
            border-style: solid;
            border-color: #5daf34;
            transition-property: width, height, background-color, border-width;
            transition-duration: 2s;
            transition-timing-function: ease-in;
            transition-delay: 500ms;
        }

        /*简写*/
        /*transition: all 2s ease-in 500ms;*/
        .base:hover {
            width: 200px;
            height: 200px;
            background-color: #5daf34;
            border-width: 10px;
            border-color: #3a8ee6;
        }
</style>
<div class="base"></div>
```

12.2.2. transform 转变动画

包含四个常用的功能：

- translate：位移
- scale：缩放
- rotate：旋转
- skew：倾斜

一般配合 `transition` 过度使用

注意的是， `transform` 不支持 `inline` 元素，使用前把它变成 `block`

举个例子

HTML 复制代码

```html
<style>
    .base {
        width: 100px;
        height: 100px;
        display: inline-block;
        background-color: #0EA9FF;
        border-width: 5px;
        border-style: solid;
        border-color: #5daf34;
        transition-property: width, height, background-color, border-width;
        transition-duration: 2s;
        transition-timing-function: ease-in;
        transition-delay: 500ms;
    }
    .base2 {
        transform: none;
        transition-property: transform;
        transition-delay: 5ms;
    }

    .base2:hover {
        transform: scale(0.8, 1.5) rotate(35deg) skew(5deg) translate(15px, 25px);
    }
</style>
 <div class="base base2"></div>
```

可以看到盒子发生了旋转，倾斜，平移，放大

12.2.3. animation 实现自定义动画

`animation` 是由 8 个属性的简写，分别如下：

属性	描述	属性值
animation-duration	指定动画完成一个周期所需要时间，单位秒（s）或毫秒（ms），默认是 0	
animation-timing-function	指定动画计时函数，即动画的速度曲线，默认是 "ease"	linear、ease、ease-in、ease-out、ease-in-out

animation-delay	指定动画延迟时间，即动画何时开始，默认是 0	
animation-iteration-count	指定动画播放的次数，默认是 1	
animation-direction 指定动画播放的方向	默认是 normal	normal、reverse、alternate、alternate-reverse
animation-fill-mode	指定动画填充模式。默认是 none	forwards、backwards、both
animation-play-state	指定动画播放状态，正在运行或暂停。默认是 running	running、pauser
animation-name	指定 @keyframes 动画的名称	

`CSS` 动画只需要定义一些关键的帧，而其余的帧，浏览器会根据计时函数插值计算出来，

通过 `@keyframes` 来定义关键帧

因此，如果我们想要让元素旋转一圈，只需要定义开始和结束两帧即可：

CSS | 复制代码

```css
@keyframes rotate{
    from{
        transform: rotate(0deg);
    }
    to{
        transform: rotate(360deg);
    }
}
```

`from` 表示最开始的那一帧， `to` 表示结束时的那一帧

也可以使用百分比刻画生命周期

```css
@keyframes rotate{
    0%{
        transform: rotate(0deg);
    }
    50%{
        transform: rotate(180deg);
    }
    100%{
        transform: rotate(360deg);
    }
}
```

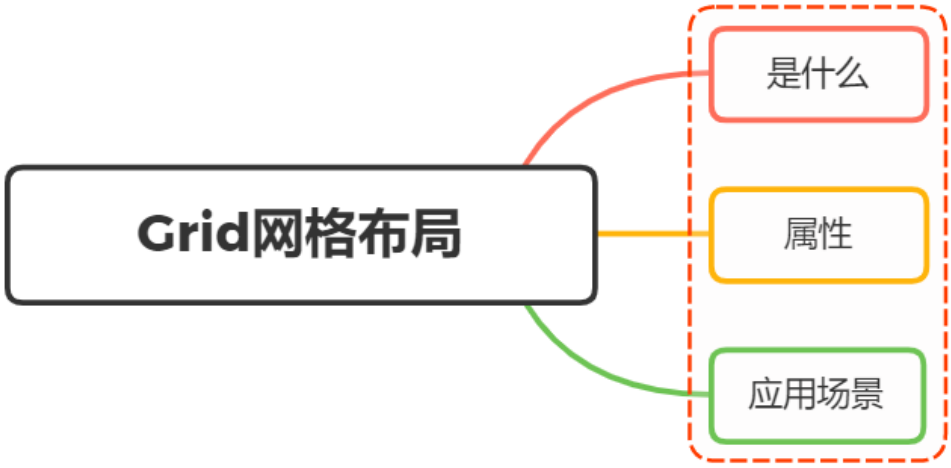
定义好了关键帧后，下来就可以直接用它了：

```css
animation: rotate 2s;
```

12.3. 总结

属性	含义
transition（过度）	用于设置元素的样式过度，和animation有着类似的效果，但细节上有很大的不同
transform（变形）	用于元素进行旋转、缩放、移动或倾斜，和设置样式的动画并没有什么关系，就相当于color一样用来设置元素的“外表”
translate（移动）	只是transform的一个属性值，即移动
animation（动画）	用于设置动画属性，他是一个简写的属性，包含6个属性

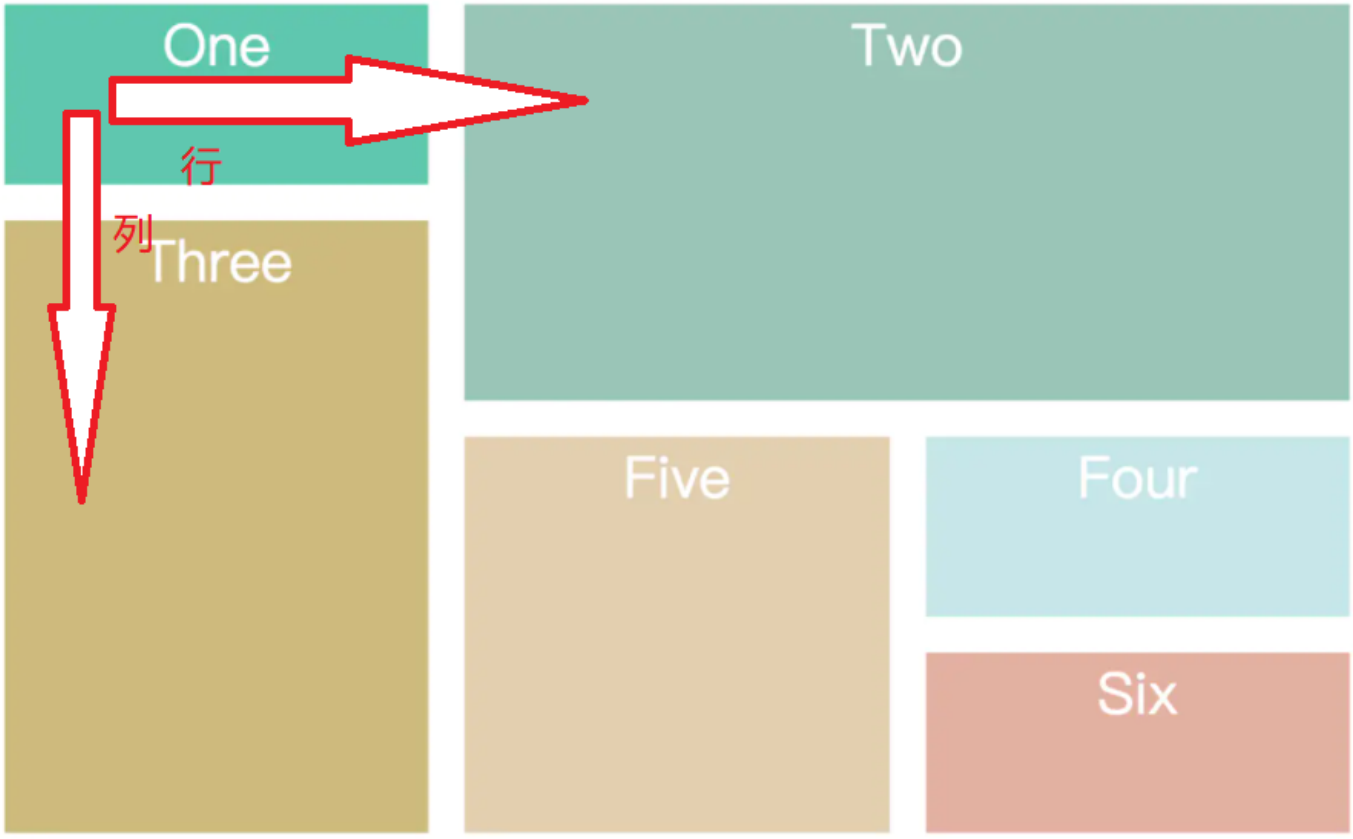
13. 介绍一下grid网格布局



13.1. 是什么

`Grid` 布局即网格布局，是一个二维的布局方式，由纵横相交的两组网格线形成的框架性布局结构，能够同时处理行与列

擅长将一个页面划分为几个主要区域，以及定义这些区域的大小、位置、层次等关系



这与之前讲到的 `flex` 一维布局不相同

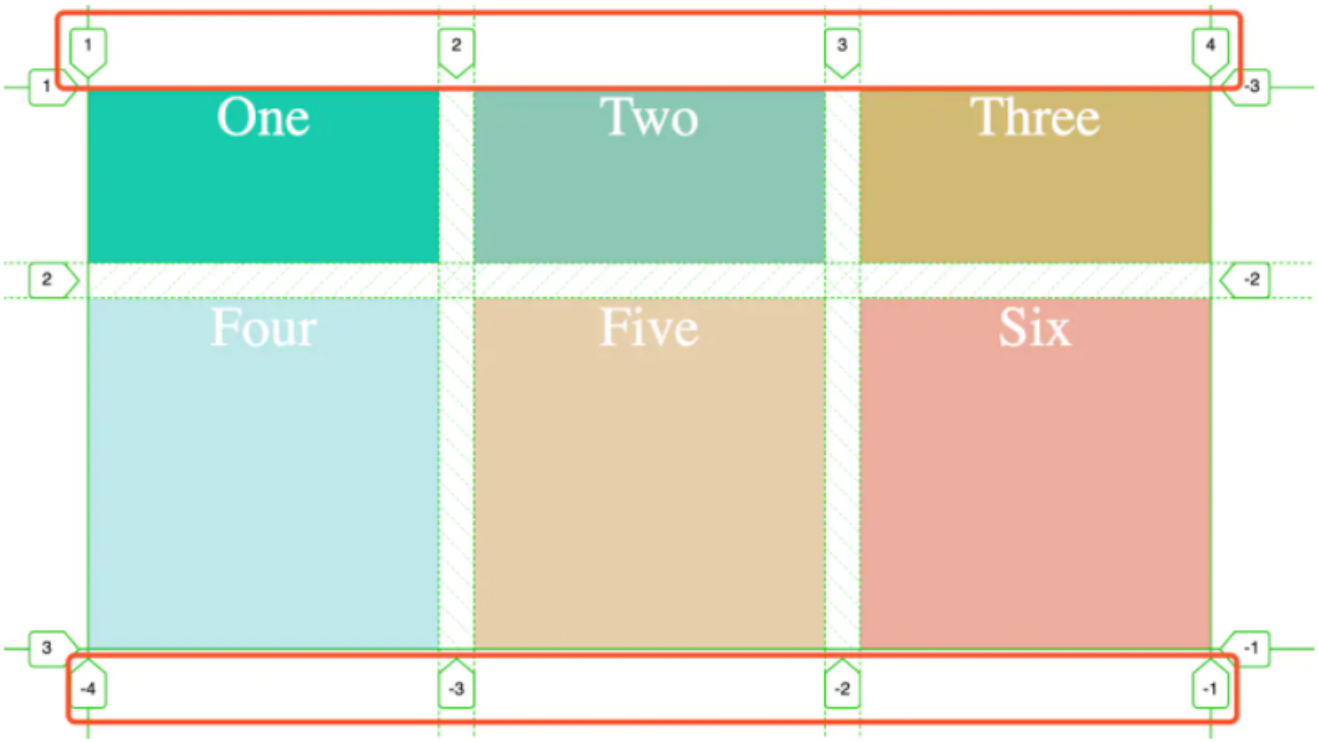
设置 `display:grid/inline-grid` 的元素就是网格布局容器，这样就能出发浏览器渲染引擎的网格布局算法

```
<div class="container">
    <div class="item item-1">
        <p class="sub-item"></p >
 </div>
    <div class="item item-2"></div>
    <div class="item item-3"></div>
</div>
```

上述代码实例中，`.container` 元素就是网格布局容器，`.item` 元素就是网格的项目，由于网格元素只能是容器的顶层子元素，所以 `p` 元素并不是网格元素

这里提一下，网格线概念，有助于下面对 `grid-column` 系列属性的理解

网格线，即划分网格的线，如下图所示：



上图是一个 2 x 3 的网格，共有3根水平网格线和4根垂直网格线

13.2. 属性

同样，`Grid` 布局属性可以分为两大类：

- 容器属性，
- 项目属性

关于容器属性有如下：

13.2.1. display 属性

文章开头讲到，在元素上设置 `display: grid` 或 `display: inline-grid` 来创建一个网格容器

- display：grid 则该容器是一个块级元素
- display: inline-grid 则容器元素为行内元素

13.2.2. grid-template-columns 属性，grid-template-rows 属性

`grid-template-columns` 属性设置列宽， `grid-template-rows` 属性设置行高

CSS | 复制代码

```css
.wrapper {
  display: grid;
  /*  声明了三列，宽度分别为 200px 200px 200px */
  grid-template-columns: 200px 200px 200px;
  grid-gap: 5px;
  /*  声明了两行，行高分别为 50px 50px  */
  grid-template-rows: 50px 50px;
}
```

以上表示固定列宽为 200px 200px 200px，行高为 50px 50px

上述代码可以看到重复写单元格宽高，通过使用 `repeat()` 函数，可以简写重复的值

- 第一个参数是重复的次数
- 第二个参数是重复的值

所以上述代码可以简写成

CSS | 复制代码

```css
.wrapper {
  display: grid;
  grid-template-columns: repeat(3,200px);
  grid-gap: 5px;
  grid-template-rows:repeat(2,50px);
}
```

除了上述的 `repeact` 关键字，还有：

- auto-fill：示自动填充，让一行（或者一列）中尽可能的容纳更多的单元格

> `grid-template-columns: repeat(auto-fill, 200px)` 表示列宽是 200 px，但列的数量是不固定的，只要浏览器能够容纳得下，就可以放置元素

- fr：片段，为了方便表示比例关系

> `grid-template-columns: 200px 1fr 2fr` 表示第一个列宽设置为 200px，后面剩余的宽度分为两部分，宽度分别为剩余宽度的 1/3 和 2/3

- minmax：产生一个长度范围，表示长度就在这个范围之中都可以应用到网格项目中。第一个参数就是最小值，第二个参数就是最大值

> `minmax(100px, 1fr)` 表示列宽不小于 `100px` ，不大于 `1fr`

- auto：由浏览器自己决定长度

> `grid-template-columns: 100px auto 100px` 表示第一第三列为 100px，中间由浏览器决定长度

13.2.3. grid-row-gap 属性， grid-column-gap 属性， grid-gap 属性

`grid-row-gap` 属性、 `grid-column-gap` 属性分别设置行间距和列间距。 `grid-gap` 属性是两者的简写形式

`grid-row-gap: 10px` 表示行间距是 10px

`grid-column-gap: 20px` 表示列间距是 20px

`grid-gap: 10px 20px` 等同上述两个属性

13.2.4. grid-template-areas 属性

用于定义区域，一个区域由一个或者多个单元格组成

CSS | 复制代码

```css
.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';
}
```

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

```css
grid-template-areas: 'a a a'
                     'b b b'
                     'c c c';
```

上面代码将9个单元格分成 `a` 、 `b` 、 `c` 三个区域

如果某些区域不需要利用，则使用"点"（ `.` ）表示

13.2.5. 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。

顺序就是由 `grid-auto-flow` 决定，默认为行，代表"先行后列"，即先填满第一行，再开始放入第二行

当修改成 `column` 后，放置变为如下：

13.2.6. 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` 属性设置单元格的垂直位置（上中下）

两者属性的值完成相同

CSS | 复制代码

```css
.container {
  justify-items: start | end | center | stretch;
  align-items: start | end | center | stretch;
}
```

属性对应如下：

- 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

13.2.7. justify-content 属性， align-content 属性， place-content 属性

`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）

CSS | 复制代码

```
.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;
}
```

两个属性的写法完全相同，都可以取下面这些值：

- 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

justify-content: start;

justify-content: end;

justify-content: center;