# css常用属性速查表

## #一、 css简介

html提供了布料，接着就该裁剪上色了。

- CSS 指层叠样式表 ( *C* ascading *S* tyle *S* heets)
- 样式定义如何显示HTML 元素
- 样式通常存储在 样式表 中
- 把样式添加到 HTML 中，是为了 解决内容与表现分离的问题
- 外部样式表可以极大提高工作效率
- 外部样式表通常存储在 *CSS* 文件 中
- 多个样式定义可 层叠 为一

## #二、定义css样式的方式

### #1、行内样式（内联样式）

应用内嵌样式到各个网页元素。

```
<p style="background: red"> I  love  java!</p>
```

1

### #2、内页样式（嵌入样式）

在网页上创建嵌入的样式表。

```
<!DOCTYPE html>
<html lang="en">
  <head>
     <style type="text/css">
          p{
              background: green;
          }
     </style>
 </head>
 <body>
     <p> I  love  java!</p>
</body>
</html>
```

### #3、外部样式

将网页链接到外部样式表。

```
<head>
      <meta charset="UTF-8">
      <title>外部样式</title>
      <link rel="stylesheet" href="style.css">
</head>
```

## #三、 css选择器

CSS选择器用于选择你想要的元素的样式的模式。

同俗点讲，就是你要给某个或某些元素上色（提供样式），先的选中它把，选择器帮你快速定位某个或某些元素。

"CSS"列表示在CSS版本的属性定义（CSS1，CSS2，或对CSS3）。

| 选择器 | 示例 | 示例说明 | CSS |
| --- | --- | --- | --- |
| .class （记住） | .intro | 选择所有class="intro"的元素，class选择器 | 1 |
| #id （记住） | #firstname | 选择所有id="firstname"的元素，id选择器 | 1 |
| * （记住） | * | 选择所有元素 | 2 |
| element（记住） | p | 选择所有 `<p>` 元素，元素选择器 | 1 |
| [element,element]（记住） | div,p | 选择所有 `<div>` 元素和 `<p>` 元素 | 1 |
| element element（记住） | div p | 选择 `<div>` 元素内的所有 `<p>` 元素 | 1 |
| element>element（记住） | div>p | 选择所有父级是 `<div>` 元素的 `<p>` 元素 | 2 |
| element+element（记住） | div+p | 选择所有紧接着 `<div>` 元素之后的 `<p>` 元素 | 2 |
| [ *attribute* ]（记住） | [target] | 选择所有带有target属性元素 | 2 |
| [ *attribute* = *value* ]（记住） | [target=-blank] | 选择所有使用target="-blank"的元素 | 2 |
| [ *attribute* ~= *value* ] | [title~=flower] | 选择标题属性包含单词"flower"的所有元素 | 2 |
| [ *attribute* \|= *language* ] | [lang\|=en] | 选择 lang 属性以 en 为开头的所有元素 | 2 |
| :link（了解） | a:link | 选择所有未访问链接 | 1 |
| :visited（了解） | a:visited | 选择所有访问过的链接 | 1 |
| :active（了解） | a:active | 选择活动链接 | 1 |
| :hover（了解） | a:hover | 选择鼠标在链接上面时 | 1 |
| :focus（了解） | input:focus | 选择具有焦点的输入元素 | 2 |
| :first-letter | p:first-letter | 选择每一个 `<p>` 元素的第一个字母 | 1 |
| :first-line | p:first-line | 选择每一个 `<p>` 元素的第一行 | 1 |
| :first-child | p:first-child | 指定只有当 `<p>` 元素是其父级的第一个子级的样式。 | 2 |
| :before（了解） | p:before | 在每个 `<p>` 元素之前插入内容 | 2 |
| :after（了解） | p:after | 在每个 `<p>` 元素之后插入内容 | 2 |
| element1~element2 | p~ul | 选择p元素之后的每一个ul元素 | 3 |

| 选择器 | 示例 | 示例说明 | CSS |
| --- | --- | --- | --- |
| [attribute^=value] | a[src^="https"] | 选择每一个src属性的值以"https"开头的元素 | 3 |
| [ *attribute* $= *value*]\|a[src ="].pdf"] | 选择每一个src属性的值以".pdf"结尾的元素 | 3 | |
| [ *attribute***= value*] | a[src*="itnan"] | 选择每一个src属性的值包含子字符串"itnan"的元素 | 3 |
| :first-of-type | p:first-of-type | 选择每个p元素是其父级的第一个p元素 | 3 |
| :last-of-type | p:last-of-type | 选择每个p元素是其父级的最后一个p元素 | 3 |
| :only-of-type | p:only-of-type | 选择每个p元素是其父级的唯一p元素 | 3 |
| :only-child | p:only-child | 选择每个p元素是其父级的唯一子元素 | 3 |
| :nth-child( *n* ) （记住） | p:nth-child(2) | 选择每个p元素是其父级的第二个子元素 | 3 |
| :nth-last-child( *n* ) | p:nth-last-child(2) | 选择每个p元素的是其父级的第二个子元素，从最后一个子项计数 | 3 |
| :nth-of-type( *n* ) | p:nth-of-type(2) | 选择每个p元素是其父级的第二个p元素 | 3 |
| :nth-last-of-type( *n* ) | p:nth-last-of-type(2) | 选择每个p元素的是其父级的第二个p元素，从最后一个子项计数 | 3 |
| :last-child | p:last-child | 选择每个p元素是其父级的最后一个子级。 | 3 |
| :enabled | input:enabled | 选择每一个已启用的输入元素 | 3 |
| :disabled | input:disabled | 选择每一个禁用的输入元素 | 3 |
| :checked | input:checked | 选择每个选中的输入元素 | 3 |
| :not( *selector* ) | :not(p) | 选择每个并非p元素的元素 | 3 |
| :read-only | :read-only | 用于匹配设置 "readonly"（只读） 属性的元素 | 3 |
| :required | :required | 用于匹配设置了 "required" 属性的元素 | 3 |
| :valid | :valid | 用于匹配输入值为合法的元素 | 3 |
| :invalid | :invalid | 用于匹配输入值为非法的元素 | 3 |

# #四、css三大特性

## #1、CSS层叠性

所谓层叠性是指多种CSS样式的叠加。

是浏览器处理冲突的一个能力,如果一个属性通过两个相同选择器设置到同一个元素上，那么这个时候一个属性就会将另一个属性层叠掉

比如先给某个标签指定了内部文字颜色为红色，接着又指定了颜色为蓝色，此时出现一个标签指定了相同样式不同值的情况，这就是样式冲突。

一般情况下，如果出现样式冲突，则会按照CSS书写的顺序，以最后的样式为准。

1. 样式冲突，遵循的原则是就近原则。那个样式离着结构近，就执行那个样式。
2. 样式不冲突，不会层叠

## #2、CSS继承性

所谓继承性是指书写CSS样式表时，子标签会继承父标签的`某些样式`，如文本颜色和字号。想要设置一个可继承的属性，只需将它应用于父元素即可。

注意：

https://www.cnblogs.com/thislbq/p/5882105.html

## #3、CSS优先级

定义CSS样式时，经常出现两个或更多规则应用在同一元素上，这时就会出现优先级的问题。

在考虑权重时，初学者还需要注意一些特殊的情况，具体如下：

继承样式的权重为0。即在嵌套结构中，不管父元素样式的权重多大，被子元素继承时，他的权重都为0，也就是说子元素定义的样式会覆盖继承来的样式。

行内样式优先。应用style属性的元素，其行内样式的权重非常高，可以理解为远大于100。总之，他拥有比上面提高的选择器都大的优先级。

权重相同时，CSS遵循就近原则。也就是说靠近元素的样式具有最大的优先级，或者说排在最后的样式优先级最大。

CSS定义了一个!important命令，该命令被赋予最大的优先级。也就是说不管权重如何以及样式位置的远近，!important都具有最大优先级。

> 优先级

定义css样式时，经常出现两个或更多规则应用在同一元素上，这时就会出现优先级的问题。

关于css权重，我们需要一套计算公式去计算，这就是css特性。

| 元素 | 贡献值 |
| --- | --- |
| 继承、* | 0，0，0，0 |
| 每个标签 | 0，0，0，1 |
| 类、伪类 | 0，0，1，0 |
| ID | 0，1，0，0 |
| 行内样式 | 1，0，0，0 |
| !important | 无穷大 |
| max-height、max-width覆盖width、height | 大于无穷大 |
| min-height、min-width | 大于max-height、max-width |

> 计算规则

1. 遇到有贡献值的就进行**累加**，例如： div ul li ---> 0,0,0,3 .nav ul li ---> 0,0,1,2 a:hover ---> 0,0,1,1 .nav a ---> 0,0,1,1 #nav p ---> 0,1,0,1
2. **数位没有进位：** 0,0,0,5+0,0,0,5 = 0,0,0,10而不是0,0,1,0，所以**不会存在10个div能赶上一个类选择器的情况** 。
3. **权重不会继承**，所以父元素的权重再高也和子元素没有关系
4. 如果有 `!important` 那么相加的那些无论多高就不管用，如果有 `max-height/max-width` 那么 `!important` 不管用，如果同时有 `min-height/min-width` 和 `max-height/max-width` ，那么 `max-height/max-width` 的不管用。

# #五、常用的单位

- px（像素）

px是绝对单位，一个像素代表一个点。如100px*100px的正方形，则是由宽度100个点，长度100个点组成的平面

- em

em是相对单位，它的参考对象是它的父级的字号，如父级字号是16px，如果设置元素的字号大小为2em的话，元素的字号大小则为32px

- rem

rem由页面的根元素html的字号决定，浏览器一般默认的字号为16px，如设置元素的字号大小为1rem，则元素的字号大小为16px

使用rem的好处是：当我们改变了浏览器的字号设置时，页面的字号也会随之发生变化，这个设置会非常方便老年人浏览网页

- 百分比

  相对与父元素的比例

# #六、元素的常用属性

## #1、字体属性：(font)

- **大小： font-size**

  x-large (特大) xx-small;(极小) 一般中文用不到，

  还可以使用单位来表示：ps em rem

- **样式：font-style**

  方便 italic(斜体) normal(正常)

- **行高：line-height**

  normal(正常) 单位：PX、 EM

- **粗细：font-weight**

  bold(粗体) lighter(细体) normal(正常)

  还可使用数值100-900，定义由粗到细的字符。400 等同于 normal，而 700 等同于 bold。

- **修饰：text-decoration**

  underline(下划线) overline(上划线) line-through(删除线)

- **常用字体： (font-family)**

  "Courier New", Courier, monospace, "Times New Roman", Times, serif, Arial, Helvetica, sans-serif, Verdana

## #2、背景属性：(background)

- **背景颜色**

  这个属性接受任何合法的颜色值，

  1、三基色比例表示， #RRGGBB，前两位是红色，中间2位是绿色，后两位是蓝色，最小是0，最大是F。

  2、rgba表示方式，rgb(255,0,0)或者rgb(100%,0,0)表示红色，可以传第四个参数代表透明度

  3、英文单词

  background-color: #FFFFFF background-color: rgb(255,0,0,.5) background-color: red

- **图片**

  {background-image: url();}

- **重复**

  {background-repeat: no-repeat;}

- **滚动**

  background-attachment:

  fixed(固定) scroll(滚动)

- **位置**

  background-position

  left(水平) top(垂直)

- **简写方法**

  background: #444444 url(2.jpg) no-repeat fixed top left;

  /简写·这个在阅读代码中经常出现，要认真的研究 /、

```
body
{
background-image:url('gradient2.png');
background-repeat:repeat-x;
}
```

## #3、列表属性：（List-style)

- **类型：list-style-type:**

  disc(圆点) circle;(圆圈) square(方块) decimal;(数字)

  lower-roman(小罗码数字) upper-roman ower-alpha（希腊） upper-alpha

- **位置：list-style-position**

  outside(外) inside

- **图像：list-style-image:**

  url(..);

## #4、边框属性：（Border)

- **边框类型：border-style**

  dotted(点线) dashed(虚线) solid（实线） double(双线)

  groove(槽线) ridge(脊状) inset (凹陷) outset;

- **边框宽度：border-width**
- **边框颜色：border-color**
- **简写方法：border：1px solidcolor;**

## #5、区块属性：（Block)

- **显示：display（重要）**

  none(不显示)

  inline(默认：内联元素)

  block(块)

  inline-block（行内块元素）

  list-item;(列表项)

> 块级元素(block)特性：

- 总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;
- 宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;

> 内联元素(inline)特性：

- 和相邻的内联元素在同一行;
- 宽度(width)、高度(height)、内边距的top/bottom(padding-top/padding-bottom)和外边距的top/bottom(margin-top/margin-bottom)都不可改变，就是里面文字或图片的大小;

**块级元素主要有：**

- div , dl , fieldset , form , h1 , h2 , h3 , h4 , h5 , h6 , hr , ol , p , table , ul , li

**内联元素主要有：**

- a , i , ./image , input , select , span , strong ,textarea

## #6、盒子模型（box）

对于块状元素，多个元素套在一起就像几个盒子套在一起，怎么紧凑的排版是个问题。


- **宽：width**

  表示内容区域的宽度

- **高：height**

  表示内容区域的宽度

- **外边距：margin**

  margin-top：本元素顶部距离上一个元素的距离

  margin-bottom：本元素低部距离上一个元素的距离

  margin-left：本元素左侧部距离上一个元素的距离

  margin-right：本元素右侧距离上一个元素的距离

  合并写： margin: 上 右 下 左 margin:10px 10px 10px 10px;

  还可以是 margin: 上下 左右 margin:10px 10px;

  左右居中 margin:0 auto; 或者 margin: auto;

  垂直居中后边说。

- **内边距：padding**

  padding-top：本元素顶部空出的距离

  padding-bottom：本元素低部空出的距离

  padding-left：本元素左侧部空出的距离

  padding-right：本元素右侧空出的距离

  合并写： padding: 上 右 下 左 padding:10px 10px 10px 10px;

  还可以是 padding: 上下 左右 padding:10px 10px;

- **box-sizing**：定义如何计算元素的宽度和高度:是否应该包含填充和边框

默认情况下，元素的宽度和高度计算如下： width + padding + border =元素的实际宽度 height + padding + border =元素的实际高度 这意味着：当您设置元素的宽度/高度时，元素通常看起来比您设置的大（因为元素的边框和填充被添加到元素的指定宽度/高度）。

content-box：More Actionscontent-box这是由 CSS2.1 规定的宽度高度行为。宽度和高度分别应用到元素的内容框。在宽度和高度之外绘制元素的内边距和边框。

border-box：元素指定的任何内边距和边框都将在已设定的宽度和高度内进行绘制。通过从已设定的宽度和高度分别减去边框和内边距才能得到内容的宽度和高度。

inherit：规定应从父元素继承 box-sizing 属性的值。

## #7、定位属性：（Position）

文档流是指在网页中将窗体自上而下分成一行行, 并在每行中按从左至右的顺序排放元素,即为文档流.(自己的理解是从头到尾按照文档的顺序，该在什么位置就在什么位置，自上而下，自左到右的顺序),这是普通流的说法。

绝对定位和固定定位会使元素脱离文档流布局。

- **定位：Position:**

  static：文档流定位，默认的。

  absolute：绝对定位，相对于一个父级是绝对定位，或者相对定位的元素而言的绝对，否则就是相对于视口

  relative;： 相对定位

  fixed: 固定定位

- **可见性：visibility**

  inherit（继承父元素）；

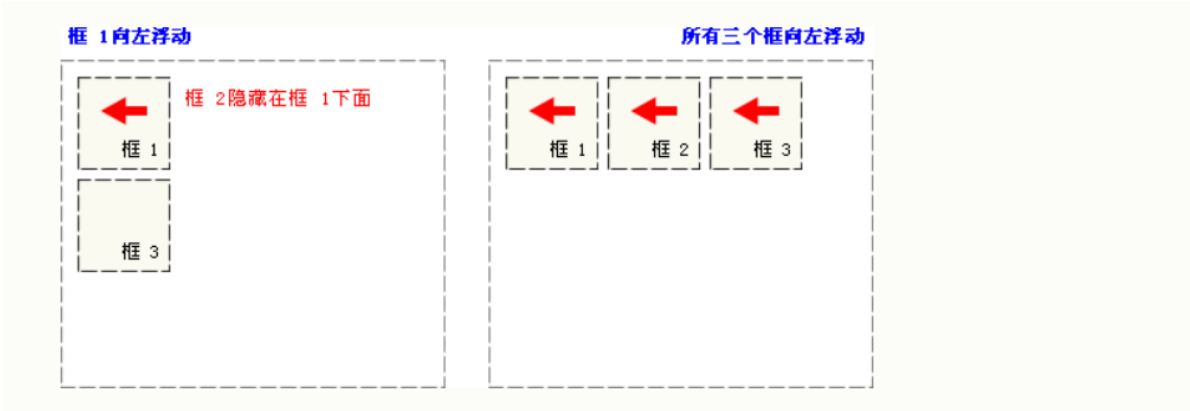
  visible（可见）

  hidden（隐藏）

- **溢出：overflow**

  visible（可见） hidden（隐藏） scroll（滚动）；

## #8、浮动




### # （1）定义浮动：float

left（左浮动） right（右浮动）

浮动的问题

浮动元素会脱离了文档流，造成父元素的高度坍塌，所以包围图片和文本的 div 不占据空间。于是使用了浮动的元素，后边紧跟其他元素会错乱，就要清除浮动。

## # （2）清除浮动

> 1、通过属性clear:both;达到清除浮动的目的；

元素浮动后，只需要在浮动元素添加多一个块级元素，并添加clear: both;属性，便可以达到清除浮动的目的。

```
<style type="text/css">
    li {
        list-style: none;
        height: 100px;
        width: 100px;
        float: left;
        background: red;
        margin-left: 20px;
    }
    ul{
        background: pink;
    }
</style>
<ul class="cc">
    <li></li>
    <li></li>
    <div style="clear: both;"></div>
</ul>
```

> 2、通过给父级元素添加伪类after，达到清除浮动的目的；

这种方式也是使用clear: both;的方式达到效果，只是变相的使用了伪类after，使得页面结构更简洁，也是常用的清理浮动的方式。

```
<style type="text/css">
    li {
        list-style: none;
        height: 100px;
        width: 100px;
        float: left;
        background: red;
        margin-left: 20px;
    }

    .cc:after {
        content: '';
        height: 0;
        line-height: 0;
        display: block;
        visibility: hidden;
        clear: both;
    }
    ul{
        background: pink;
    }
</style>
```

```
<ul class="cc">
    <li></li>
    <li></li>
</ul>
```

> 3、使用双伪类；

此方式和三原理一样，代码更简洁。

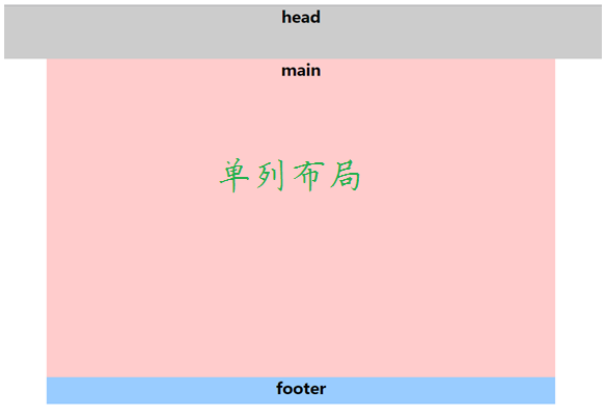
```
<style type="text/css">
    li {
        list-style: none;
        height: 100px;
        width: 100px;
        float: left;
        background: red;
        margin-left: 20px;
    }

    .cc:after,
    .cc:before {
        content: "";
        display: block;
        clear: both;
    }

    ul {
        background: pink;
    }
</style>
<ul class="cc">
    <li></li>
    <li></li>
</ul>
```

# #七、常见的网页布局

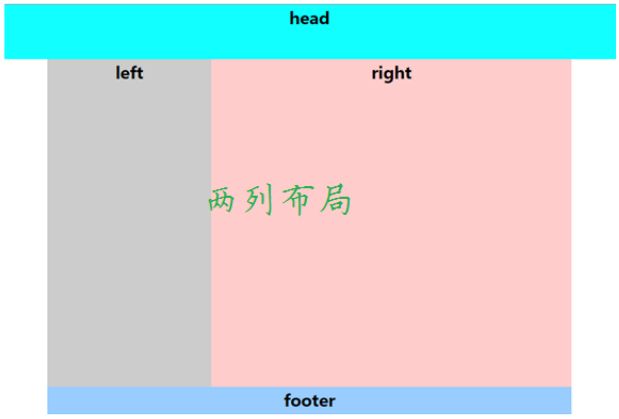
现在的网页差不多都采用分列布局，常见的有单列布局（如百度首页）、双列布局、三列布局和混合布局，超过三列的布局很少见，在此只介绍以上四种。

注：混合布局可以看作是在三列布局的基础上，再继续分块。本文仅从宏观结构上介绍，比如新浪、腾讯首页可以看作是三列布局，淘宝、京东首页可以看做是混合布局，这些大型网站会根据其内容更改其布局。

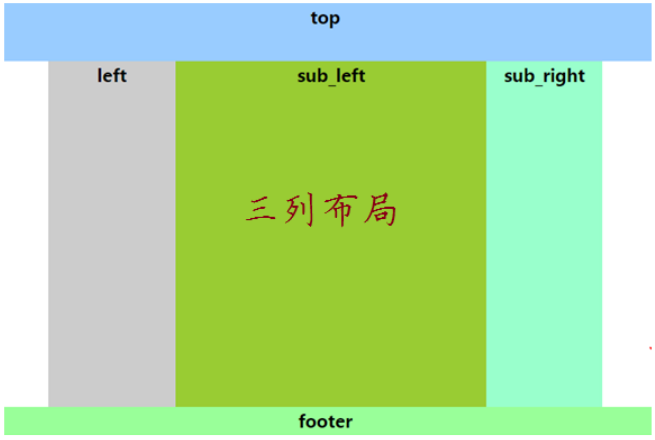
最简单的要数单列布局了，这种布局适合各种搜索引擎主页，干净的界面和较少的干扰信息给用户较好的体验。



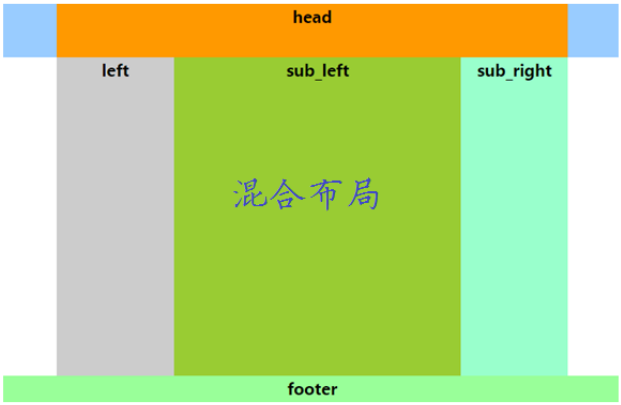

两列布局：




三列布局：




混合布局：



可以发现，网页布局无非就是并列、嵌套、层叠这几种，将网页的结构分辨清楚，要设计出类似的网站也就不是什么难题了。下面来分析一下布局的相关代码：

**单列布局**在于设置块状元素居中，只需设置**margin:0 auto**;(前面的0是上下外面局，可任意设置)。

两列布局在于设置中间主体并排分布，在**左边的块设置float:left**，在**右边的块设置float:right**即可，当然要使两者的width之和等于某一设定值；

三列布局只是在两列布局的基础上再将中间块进行二次分割，方法一致，在此不做赘述。

最后说一下混合布局中的一个注意事项：由于中间主体设置了float，最后的footer需要通过**清除浮动**来正确显示在主体下方，**clear:both**。

# #八、动画入门

在CSS3中，使用动画可以使网页上文字或者图像具有动画效果，可以使背景颜色从一种颜色平滑过渡到另外一种颜色。Transition功能支持从一个属性值平滑到另外一个属性值；Animation功能支持通过关键帧的指定来在页面上产生更复杂的动画效果。

浏览器前缀：

由于浏览器对动画的兼容性不同，我们在定义动画时通常要针对不同的浏览器，进行不同的处理。

```
    -moz-transition: width 2s ease-in 3s,height 3s;
    /* Firefox 4 */
    -webkit-transition: width 2s ease-in 3s,height 3s;
    /* Safari and Chrome */
    -o-transition: width 2s ease-in 3s,height 3s;
    /* Opera */
```

## #1、Transition功能

CSS3中Transition允许CSS的属性值在一定的时间区间内平滑地过渡。这种效果可以在鼠标单击、获得焦点、被点击或对元素任何改变中触发，并平滑地以动画效果改变CSS的属性值。

使用方法： transition:语法; -moz-transition:语法;//Firefox 4 -webkit-transition:语法;//Safari 和 Chrome -o-transition:语法;//Opera

语法： transition:property duration timing-function delay;

1. **执行变换的属性**：property。属性规定应用过渡效果的CSS属性的名称。(当指定的CSS属性改变时，过渡效果将开始) 三个类型： a.none 没有属性会获得过渡效果 b.all 所有属性都将获得过渡效果 c.property 定义应用过渡效果的CSS属性名称列表，列表以逗号分隔
2. **变化延续的时间**：duration。 规定完成过渡效果需要花费的时间（以秒或毫秒计），默认值0没有效果。
3. **在延续时间段**，变换的速率变化：timing-function。 值： a.ease:(逐渐变慢)默认值,ease函数等同于贝塞尔曲线(0.25,0.1,0.25,1.0). b.linear:(匀速),linear函数等同于贝塞尔曲线(0.0,0.0,1.0,1.0). c.ease-in:(加速),ease-in函数等同于贝塞尔曲线(0.42,0,1.0,1.0). d.ease-out:(减数),ease-out函数等同于贝塞尔曲线(0,0,0.58,1.0). e.ease-in-out:(加速然后减数),ease-in-out函数等同于贝塞尔曲线(0.42,0,0.58,1.0) f.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。
4. **变化延续时间**：delay。 delay是用来指定一个动画开始执行的时间，也就是说当改变元素属性值后多长时间开始执行transition效果，其取值:为数值，单位为s（秒）或者ms（毫秒）。

代码：

```
    <html>
```

```
    <head>
        <style>
            div {
                width: 100px;
                height: 100px;
                background: blue;
                transition: width 2s ease-in 3s,height 3s;
                -moz-transition: width 2s ease-in 3s,height 3s;
                /* Firefox 4 */
                -webkit-transition: width 2s ease-in 3s,height 3s;
                /* Safari and Chrome */
                -o-transition: width 2s ease-in 3s,height 3s;
                /* Opera */
            }
            div:hover {
                width: 300px;
                height: 300px;
            }
        </style>
    </head>
    <body>
        <div></div>
        <p>请把鼠标指针移动到蓝色的 div 元素上，就可以看到过渡效果。</p>
        <p><b>注释：</b>本例在 Internet Explorer 中无效。</p>
    </body>
</html>
```

## #2、Animation功能

IE 10和Firefox（>= 16）支持没有前缀的animation，firefox（<16）使用-moz-前缀，因为现在firefox的版本也都不低，所以firefox都直接使用没有前缀的animation。 而chrome，safari，opera不支持，所以必须使用webkit前缀。

在CSS中我们除了可以使用Transition实现动画效果，还可以使用Animation来实现动画效果。

1. 使用Transition和Animation的区别 Transition只能通过指定属性的开始值与结束值，然后通过两属性值之间的平滑过渡来实现动画效果，所以Transition不能实现复杂的动画效果；Animation功能是通过关键帧以及每个关键帧中的属性值来实现更为复杂的动画效果。
2. Animation的使用方法

   **@-webkit-keyframes 关键帧合集名称**{创建关键帧的代码} 0%~100%{ 本关键帧中的样式 }

   关键帧创建好了之后，还要在元素的样式中**使用该关键帧**。

   方法如下：

   元素{ -webkit-animation-name:关键帧合集名称; -webkit-animation-duration:5s; -webkit-animation-timing-function:linear; -webkit-animation-iteration-count:infinite; }

   a.-webkit-animation-name 指定合集名称 b. -webkit-animation-duration 整个动画执行完成所需的时间 c. -webkit-animation-timing-function:linear 实现动画的方法 d.-webkit-animation-iteration-count:infinite 如果设定为整数值，那么这个动画的播放次数就等于这个整数值；如果为infinite，则是无限循环
3. 实现动画的方法 a.linear：匀速。 b.ease-in：先慢，后沿着曲线加快。 c.ease-out：先快，后沿着曲线减速。 d.ease：先快，后沿着曲线减速，再沿着曲线加快。 e.ease-in-out：先慢，后沿着曲线加快，再沿着曲线减速。

使用示例如下：

```
div{
    animation:myAnim 1s;
    -webkit-animation:myAnim 1s;
}

@keyframes myAnim{
  0% { background: #f00; }
  50% { background: #0f0; }
  100% { background: yellowgreen; }
}

@-webkit-keyframes myAnim{
  0% { background: #f00; }
  50% { background: #0f0; }
  100% { background: yellowgreen; }
}
```

```
<meta charset="UTF-8">
<title>Animations属性</title>
<style>
    div{
        width: 200px;
        height: 200px;
        background-color: #ff6600;
    }

    @keyframes myAnim{
        0% { background: #f00; }
        50% { background: #0f0; transform:rotate(30deg)}
        100% { background: yellowgreen; }
    }

    @-webkit-keyframes myAnim{
        0% { background: #f00; }
        50% { background: #0f0; transform:rotate(30deg)}
        100% { background: yellowgreen; }
    }

    div:hover{
        animation-name: myAnim;
        animation-duration:5s;
        animation-timing-function:linear;
        animation-iteration-count:2;
        -webkit-animation-name: myAnim;
        -webkit-animation-duration:5s;
        -webkit-animation-timing-function:linear;
        -webkit-animation-iteration-count:2;
    }
</style>
</head>
<body>
<h1>Animations属性</h1>
<div></div>
</body>
</html>
```

这个不做要求

其实有个css动画库： https://animate.style/

第一步：引用

```
<link
    rel="stylesheet"

 href="https://cdnjs.cloudflare.com/ajax/libs/animate.css/4.1.1/animate.min.css"
/>
```

第二步：使用

```
<h1 class="animate__animated animate__bounce">An animated element</h1>
```

第三步：


点击复制，替换了class的后半部分即可。

# #九、flex布局

源自博客： https://blog.csdn.net/domunweb/article/details/95064383

## #1、 flex 解释

1、flex 布局 为 flexible BOX 的缩写 ， 意思为 弹性布局。 2、块级元素和行内块级元素都可以使用flex布局 3、Webkit内核的浏览器，需要加上-webkit前缀。

## #2、 flex 容器 属性

首先父容器要加上

```
display: flex
```

### # （1）flex-direction

此属性决定主轴的方向

```
.flex{
    flex-direction: row; // (默认值) 主轴水平方向，从左往右    如图:
    flex-direction: row-reverse; // 主轴水平方向的逆方向，从右往左
    flex-direction: column; // 主轴为垂直方向，从上往下
    flex-direction: column-reverse; // 主轴为垂直方向的逆方向，从下往上
}
```

## # （2）flex-wrap

此属性定义，如果一条轴线上排列不下，换行的方式

```
.flex{
    flex-wrap: nowrap; // （默认）不换行    如图：
    flex-wrap: wrap; // 正常换行 从上到下    如图：
    flex-wrap: wrap-reverse; // 逆方向换行 从下到上   如图：
}
```

## # （3）flex-flow

此属性定义，是flex-direction和flex-wrap的合并；

```
.flex {
    flex-flow: <flex-direction>  空格 <flex-wrap>;
}
```

## # （4）justify-content

此属性定义，项目在主轴上的对齐方式

```
.flex{
    justify-content: flex-start; // 左对齐（默认）
    justify-content: flex-end;  // 右对齐
    justify-content: center;    // 居中
    justify-content: space-between; // 两端对齐。且项目间间隔相等
    justify-content: space-around; // 每个项目两侧间隔相等，所以项目间 间隔 比项目与边框间间隔大一倍
    justify-content:  space-evenly; // 项目间间隔与项目与边框间 间隔均匀分配
}
```

## # （5）align-items

此属性定义，项目在交叉轴上的对齐方式

```
.flex{
    align-items: flex-start; // 交叉轴的起点对齐
    align-items: flex-end; // 交叉轴的终点对齐
    align-items:center; // 交叉轴的中点对齐
    align-items: baseline; // 项目的第一行文字的基线对齐
    align-items: stretch; // (默认值) 如果项目未设置高度或设为auto ，将充满整父级容器高度
}
```

## # （6）align-content

此属性定义，多个项目多根轴线的对齐方式，只有一个轴线时没有作用

```
.flex{
    align-content: flex-start; // 与交叉轴的起点对齐。
    align-content: flex-end; // 与交叉轴的终点对齐。
    align-content: center; // 与交叉轴的中点对齐。
    align-content: space-between; // 与交叉轴两端对齐，轴线之间的间隔平均分布。
    align-content: space-around; // 每根轴线两侧的间隔都相等。所以，轴线之间的间隔比轴线与边框的间隔大一倍。
    align-content: stretch; // （默认值）轴线占满整个交叉轴。
}
```

## #3、flex 项目 属性

### # （1）order

此属性决定项目的排列顺序，数值越小。排列越靠前。

```
.box{
    order: number; // 默认为 0 ,要讲哪个项目向前移动 将其order设置为负数。
}
```

### # （2）flex-grow

此属性决定项目的放大比列。

```
.box{
    flex-grow: 1; // 默认为 0
}
```

### # （3）flex-shrink

此属性决定项目的缩小比列。

```
.item {
  flex-shrink: number; // 默认为1.要将哪个项目缩小 值设为0 ；
}
```

### # （3）flex-basis

定义了元素的宽度，覆盖width

### # （4）flex

此属性定义为 flex-grow, flex-shrink 和 flex-basis的简写，默认值为0 1 auto。后两个属性可选。

```
.item{
    flex: flex-grow flex-shrink flex-basis; // 简写方式
}
```

### # （5）align-self

此属性允许单个项目有与其他项目不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父元素的align-items属性，如果没有父元素，则等同于stretch。

```
.item{
    align-self: auto;
    align-self: flex-start;
    align-self: flex-end;
    align-self: center;
    align-self: baseline;
    align-self: stretch;
}
```