

## 文档声明

h5的文档声明，声明当前的网页是按照HTML5标准编写的  
编写网页时一定要将h5的文档声明写在网页的最上边  
如果不写文档声明，则会导致有些浏览器会进入一个怪异模式，  
进入怪异模式以后，浏览器解析页面会导致页面无法正常显示，所以为了避免进入该模式，一定要写文档声明

## 字符实体

语法：&实体的名字

&nbsp: 一个空格

&gt: 大于号

&lt: 小于号

&copy: 版权符号

等等还有很多字符实体

## meta

表示网页的元数据

### meta有哪些属性

charset: 指定网页的字符集

name: 指定的数据的名称

content: 指定的数据的内容（一般content和name配合使用，一个表示元数据的名字，一个表示元素数据的内容）

### 有哪些元数据：

keywords: 表示网页的关键字，便于搜索—宁搜索时使用，可以使用多个关键字，关键字放在content中，关键字之间用逗号隔开

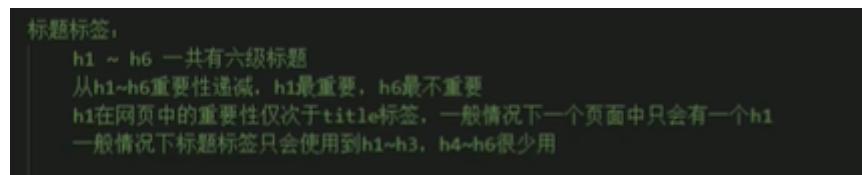
```
<meta name="keywords" content="HTML5,前端,CSS3">
```

description: 用于指定网站的描述，描述的内容会出现在搜索引擎的结果中

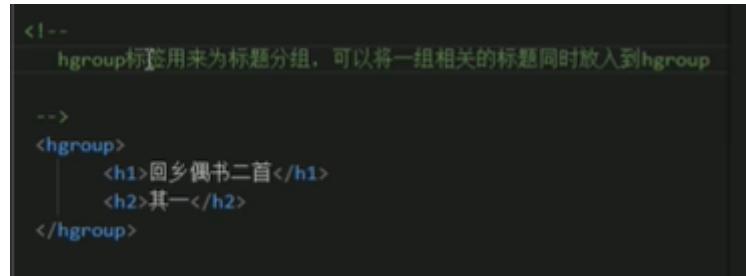
http-equiv: 用于页面的重定向，content中两个参数，用分号隔开，第一个写重定向的延迟事件(秒)，第二个是一个重定向地址

# 标签

h标签

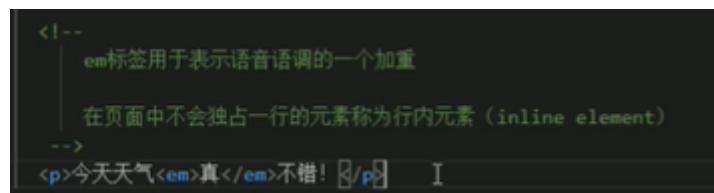


hgroup标签



em标签

表示语音语调的一个加重



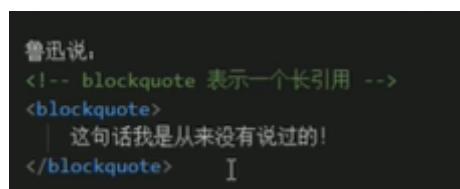
真这个字体会变斜

strong标签

表示强调重要的内容

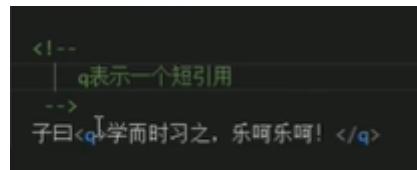
blockquote标签

表示长引用，是块元素，该行会缩进



q标签

表示短引用，是行内元素，该段文字会用双引号引起来



br标签

自结束标签，用来换行

p标签

在p标签中的内容就属于一个段落

span标签

**行内元素**，没有任何语义，一般用于网页中选中文字，行内元素现在最主要使用的就是行内元素

## 块级元素

div标签

没有语义，表示一个区块，目前还是我们的一个主要的布局元素，div可以代替下面所讲的任何标签，只不过div有点违反了h5的思想：**用html标签来确定各个内容的语义**

h5新增header、main、footer、nav、aside、article、section

header标签

网页的头部

main标签

网页的主体部分，一个页面一个main

footer

网页的底部

nav标签

网页的导航

aside标签

和主题相关的其他部分（侧边栏）

article标签

表示一个独立的文章

section标签

表示一个独立的区块，上面的标签都不能使用时可以用section

## 块和行内元素

网页中通过块元素进行页面布局，行内元素一般用来包裹文字

一般情况不会往行内元素中放块元素，块元素中基本上什么都能放

p元素中不能放任何块元素

浏览器在解析网页时，会自动对网页中不符合规范的内容进行修正，并不会报错：

标签写在了根元素的外部

p元素中套了块元素

根元素中出现了 head和abody以外的子元素

.....

## id属性

id属性 (唯一不重复的属性)

每一个标签都可以添加一个id属性

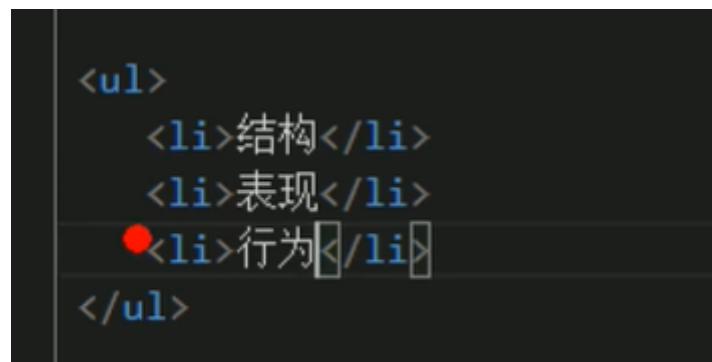
id属性就是元素的唯一标识，同一个页面中不能出现重复的id属性

## 列表

### 有序列表

使用ul标签创建有序列表

在ul中使用li标签来创建列表项



```
<ul>
  <li>结构</li>
  <li>表现</li>
  <li>行为</li>
</ul>
```

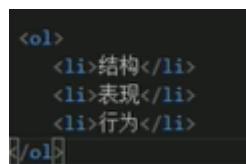
A screenshot of a code editor showing an ordered list. The list items are '结构', '表现', and '行为'. The third item, '行为', has a red dot next to its opening tag, likely indicating it is selected or being edited.

- 结构
- 表现
- 行为

### 无序列表

使用ol标签创建有序列表

在ol中使用li标签来创建列表项



```
<ol>
  <li>结构</li>
  <li>表现</li>
  <li>行为</li>
</ol>
```

A screenshot of a code editor showing an ordered list. The list items are '结构', '表现', and '行为'. The third item, '行为', has a red dot next to its opening tag, likely indicating it is selected or being edited.

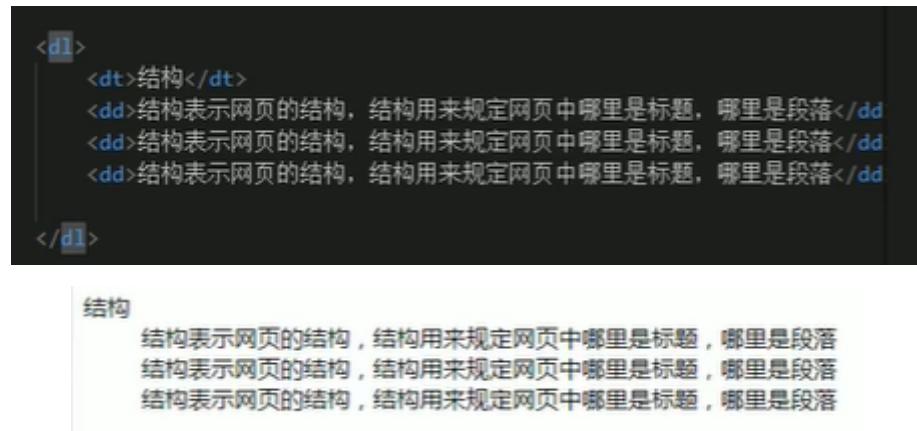
1. 结构
2. 表现
3. 行为

## 定义列表

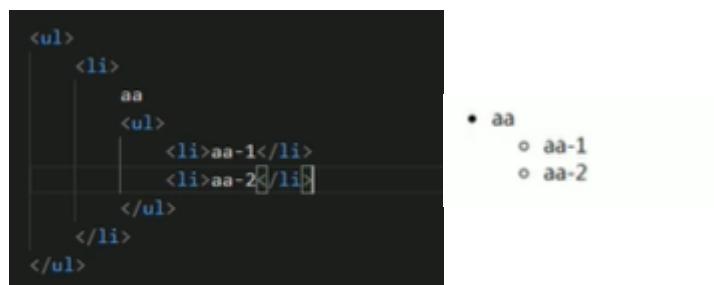
使用**dl**标签创建有序列表

在dl中使用**dt**标签来表示定义的内容

使用**dd**标签来对内容进行解释说明



列表之间可以互相嵌套



## 超连接

### a标签

超链接也是一个行内元素，特殊的是，a标签中可以放出它自身以外的任何的标签，包括块元素如div

### a标签的属性

超链接可以让我们跳转到其他页面，或者当前页面的其他位置

#### href属性

来指明跳转的地址：

或者将href的值设置为#，这样超链接就不会发生跳转，而是直接回到顶部

可以跳转到页面的指定位置，只需将href属性设置#目标元素的id属性值

```
17 | -->
18 | <a href="07.列表.html" target="_blank">超链接</a>
19 |
20 | <br><br>
21 | <a href="#bottom">去底部</a>
22 |
23 | <br><br>
24 | <a href="#p3">去第三个自然段</a>
25 |
26 | <p>在我的后园，可以看见墙外有两株树，一株是枣树，还有一株也是枣树。</p>
27 | <p>在我的后园，可以看见墙外有两株树，一株是枣树，还有一株也是枣树。</p>
28 | <p id="p3">在我的后园，可以看见墙外有两株树，一株是枣树，还有一株也是枣树。</p>
29 | <p>在我的后园，可以看见墙外有两株树，一株是枣树，还有一株也是枣树。</p>
30 | <p>在我的后园，可以看见墙外有两株树，一株是枣树，还有一株也是枣树。</p>
31 | <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Qu
32 |
33 |
34 | <!--
35 |     可以直接将超链接的href属性设置为#，这样点击超链接以后
36 |         页面不会发生跳转，而是转到当前页面的顶部的位置
37 |
38 |     可以跳转到页面的指定位置，只需将href属性设置 #目标元素的id属性值
39 |
40 |     id属性（唯一不重复的）
41 |         - 每一个标签都可以添加一个id属性
42 |         - id属性就是元素的唯一标识，同一个页面中不能出现重复的id属性
43 | -->
44 | <a id="bottom" href="#">回到顶部</a>
```

未设置的超链接href属性可以用#作为占位符

href属性也可以用**javascript:**来作为占位符，这个是真正的点击什么也不会发生

```
<!-- 在开发中可以将#作为超链接的路径的展位符使用 -->
<a href="#">这是一个新的超链接</a>

<br><br>

<!-- 可以使用 javascript:; 来作为href的属性，此时点击这个超链接什么也不会发生 -->
<a href="javascript:;">这是一个新的超链接</a>
<br><br>
```

## target属性

用来指明超连接打开的位置

可选值：

\_self：默认值， 默认在当前页面中打开超连接

\_blank：在一个新的页面中打开超连接

## 相对路径

```
<!--  
当我们需要跳转一个服务器内部的页面时，一般我们都会使用相对路径  
相对路径都会使用.或..开头  
./  
../  
.可以省略不写，如果不写./也不写../则就相当于写了./  
  
. 表示当前文件所在的目录  
- 在这里当前页面就是 09.相对路径.html  
- ./就等于 09.相对路径.html 所在的目录 path  
  
../ 表示当前文件所在目录的上一级目录
```

## img标签

图片标签用于向当前页面中引入一个外部图片

img标签是一个自结束标签

img这种元素属于**替换元素**（块和行内元素之间，具有两种元素的特点

**src属性**：指定的是外部图片的路径（路径规则和超链接是一样的）

**alt属性**：图片的描述，这个描述默认情况下不会显示，**有些浏览器会图片无法加载时显示。搜索引擎会根据alt中的内容来识别图片，如果不写alt属性则图片不会被搜索引擎所识别** **width属性**：图片的宽度（单位是像素） **height属性**：图片的高度

宽度和高度中如果只修改了一个，则另一个会等比例缩放

**注意：**

一般情况在pc端，不建议修改图片的大小，需要多大的图片就裁多大

但是在移动端，经常需要对图片进行缩放（大图缩小）

## 图片格式

图片的格式:

### jpeg(jpg)

- 支持的颜色比较丰富，不支持透明效果，不支持动图
- 一般用来显示照片

### gif

- 支持的颜色比较少，支持简单透明，支持动图
- 颜色单一的图片，动图

### png

- 支持的颜色丰富，支持复杂透明，不支持动图
- 颜色丰富，复杂透明图片（专为网页而生）

### webp

- 这种格式是谷歌新推出的专门用来表示网页中的图片的一种格式
- 它具备其他图片格式的所有优点，而且文件还特别的小
- 缺点：兼容性不好

### base64

- 将图片使用base64编码，这样可以将图片转换为字符，通过字符的形式来引入图片
- 一般都是需要和网页一起加载的图片才会使用base64

效果一样，用小的

效果不一样，用效果好的

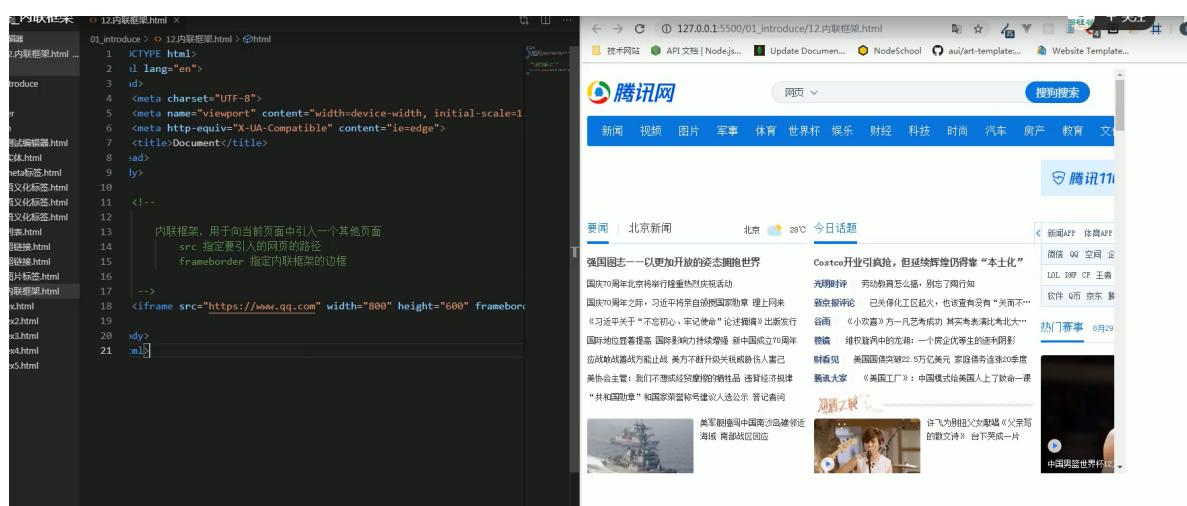
## 内联框架

### iframe标签

内联框架，用于向当前页面中引入一个其他页面

src: 指定要引入的网页的路径

frameborder: 指定内联框架的边框，1: 有框架，0: 没有框架



## 音视频

## audio标签

用来向页面中引入一个外部的音频文件的

音视频文件引入时，默认情况下不允许用户自己控制播放停止

属性：

**src**: 音频文件的地址

**controls**: 加上controls允许用户控制播放 (controls和autoplay都是直接加上就行，**不用赋值**)

**autoplay**: 加上autoplay，音频文件自动播放

如果设置了 autoplay则音乐在打开页面时会自动播放

但是目前来讲大部分浏览器都不会自动对音乐进行播放

**loop**: 音乐是否循环播放

```
<audio src="./source/audio.mp3" controls autoplay></audio>
```

## source标签

audio标签中可以嵌套一个source标签来指定外部文件路径

使用source标签的话有**两个好处**：

①里面可以放置多个source标签，可以放置多个音频文件，浏览器根据能否播放，会从上到下播放一个，解决个别浏览器对音频文件的兼容问题

②audio可以写一些文字，当音频文件加载不出来，浏览器自动修改错误，将audio标签去掉，文字才会显示，这样可以作为错误提示。

## embed标签

老版本的ie音视频标签，视频音频都可以使用，不做推荐

## video标签

属性和audio标签基本一样

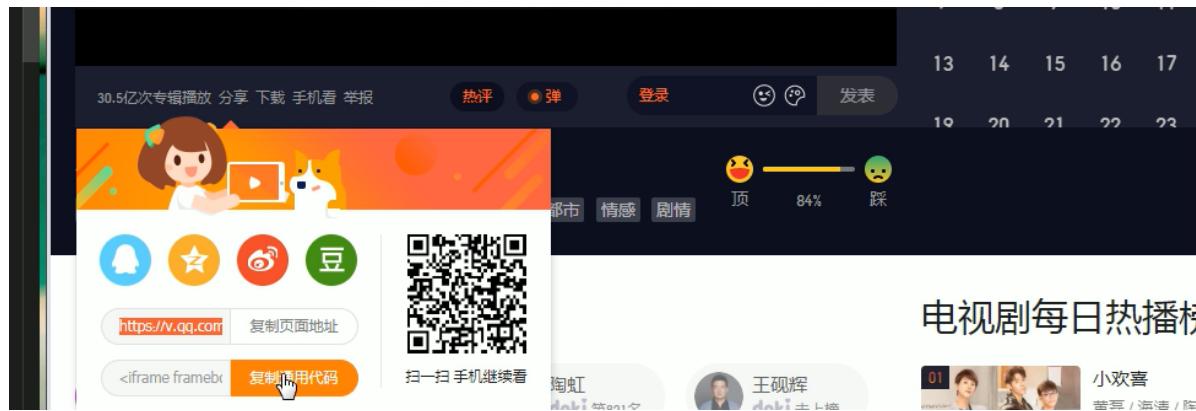
也可以使用source标签完成兼容问题

```
<video controls>
  <source src="./source/flower.webm">
  <source src="./source/flower.mp4">
  <embed src="./source/flower.mp4" type="video/mp4">
</video>
```

**ps:**

一般视频文件不会放到本地服务器，src不会相对路径，这对带宽要求很大，成本很高，所以一般租用**第三方的服务器**，放到第三方的服务器上面进行托管

再或者放到腾讯、优酷等视频网站上面，然后**复制相应视频的通用代码**，此代码是一个**iframe内联框架**复制到video标签中去



效果这样：



此时使用的是腾讯的服务器，对自己的服务器不会产生压力

## CSS

css修改元素样式的方式

**第一种：**内联样式/行内样式

通过在标签内部修改style属性来设置元素的样式



问题：

使用内联样式，样式只能对一个标签生效

如果希望影响到多个元素必须在每一个元素中都复制一遍

样式发生改变的时候需要一个一个修改，不方便维护

### 非常不推荐使用

#### 第二种：内部样式表

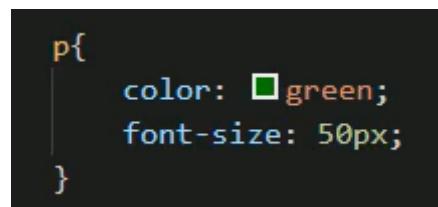
将样式编写到**head**标签中的 **style**标签里

然后通过**CSS的选择器**来选中元素并为其设置各种样式

可以为多个标签设置样式，并且修改时只需要修改一处即可全部应

内部样式表更加方便对样式进行复用

内部样式表只能对一个网页起作用，不能跨页面进行复用



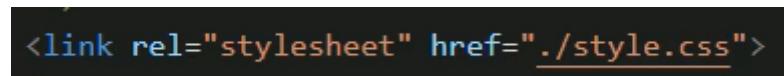
#### 第三种：外部样式表 最佳实践

可以将样式表编写到一个外部的CSS文件

然后通过**link标签**来引入外部的CSS文件

意味着只要想使用这些样式的网页都可以对其进行引用

使样式可以在不同页面之间进行复用



## CSS语法

CSS中的注释：/\* \*/

注释中的内容会自动被浏览器所忽略

CSS的基本语法：

### 选择器：

通过选择器可以选中页面中的指定元素

比如p的作用就是选中页面中所有的p元素

### 声明块：

通过声明块来指定要为元素设置的样式

声明是一个名值对结构

一个**样式名**对应一个**样式值**，名和值之间以**:**连接，以**;**结尾

## 常用选择器

### 元素选择器

作用：根据**标签名**来选中指定的元素

语法：标签名{}

例子：p{} h1{} div{}

```
p{
    color: red;
}

h1{
    color: green;
}
```

### id选择器

作用：根据元素的**id属性值**选中一个元素

语法：#id属性值{}

例子：#box{} #red{}

```
34
35     #red{
36         color: red;
37     }

38
39
40     </style>
41 </head>
42 <body>
43     <h1>我是标题</h1>
44     <p>少小离家老大回</p>
45     <p>乡音无改鬓毛衰</p>
46     <p id="red">儿童相见不相识</p>
47     <p>笑问客从何处来</p>
```

### 类选择器

作用：根据元素的**class属性值**选中一组元素

语法：.class属性值{}

```
45         .blue{  
46             color: blue;  
47         }  
48  
49  
50     </style>  
51 </head>  
52 <body>  
53     <h1>我是标题</h1>  
54     <p>少小离家老大回</p>  
55     <p>乡音无改鬓毛衰</p>  
56     <p id="red">儿童相见不相识</p>  
57     <p>笑问客从何处来</p>  
58     <!--  
59         class 是一个标签的属性，它和id类似，不同的是class可以重复使用  
60         可以通过class属性来为元素分组  
61     -->  
62     <p class="blue">秋水共长天一色</p>  
63     <p class="blue">落霞与孤鹜齐飞</p>  
64 
```

class是一个标签的属性，和id类似，但是可以重复使用

通过class可以给元素分组

可以同时为**一个元素分配多个class属性值**，写到一个class中**用空格隔开**就行

```
.blue{  
    color: blue;  
}  
  
.abc{  
    font-size: 20px;  
}  
  
</style>  
</head>  
<body>  
    <h1 class="blue abc">我是标题</h1>
```

## 通配选择器

作用：选中页面中的所有元素

语法：\*{}

```
/*  
 *{  
 |   color: red;  
 }  
*/
```

## 复合选择器

### 交集选择器

作用：选中同时复合多个条件的元素

语法：选择器1选择器2选择器3选择器n{}

注意点：

交集选择器中如果有元素选择器，必须使用元素选择器开头

```
/*  
div.red{  
  font-size: 30px;  
}  
  
.a.b.c{  
  color: blue  
}  
*/
```

### 分组选择器（并集选择器）

作用：同时选择多个选择器对应的元素

语法：选择器1,选择器2,选择器3,选择器n{}

注意点：并集选择器可以跟交集选择器一块用

如：`#b1,.p1,h1,span,div.red{}`

```
/*  
#b1,.p1,h1,span,div.red{}  
*/  
h1, span{  
  color: green  
}
```

## 关系选择器

## 父元素

- 直接包含子元素的元素叫做父元素

## 子元素

- 直接被父元素包含的元素是子元素

## 祖先元素

- 直接或间接包含后代元素的元素叫做祖先元素

- 一个元素的父元素也是它的祖先元素

## 后代元素

- 直接或间接被祖先元素包含的元素叫做后代元素

- 子元素也是后代元素

## 兄弟元素

- 拥有相同父元素的元素是兄弟元素

## 子元素选择器

作用：选中指定父元素的**指定子元素**，子元素都符合就选中多个元素

语法：**父元素>子元素**

```
div.box > span{  
    color: orange;  
}
```

可以与之前的**复合选择器**配合使用，可以使用多个>

```
div > p > span{  
    color: red;  
}
```

## 后代元素选择器

作用：选中指定元素内的**指定后代元素**，后代符合的有多个就选择多个

语法：**祖先 后代**（中间隔着空格）

```
div span{  
    color: skyblue  
}
```

同样也可以多个空格连起来或者与其他配合使用

## 选择下一个兄弟

语法：前一个+下一个

```
p + span{  
    color: red;  
}
```

**这两个兄弟元素必须紧挨着，中间不能隔着任何标签**

### 选择下面所有兄弟

语法：兄 ~ 弟

```
p ~ span{  
    | color: red;  
}
```

### 属性选择器

[属性名] 选择含有指定属性的元素  
[属性名=属性值] 选择含有指定属性和属性值的元素  
[属性名^=属性值] 选择属性值以指定值开头的元素  
[属性名\$=属性值] 选择属性值以指定值结尾的元素  
[属性名\*=属性值] 选择属性值中含有某值的元素

```
/* p[title]{ */  
/* p[title=abc]{ */  
/* p[title^=abc]{ */  
/* p[title$=abc]{ */
```

此例前面有p，是一个复合选择器，也可以不加

### 伪类选择器

**伪类** (不存在的类，特殊的类)

伪类用来描述一个元素的**特殊状态**

比如：第一个子元素、被点击的元素、鼠标移入的元素

伪类一般情况下都是使冒号开头，**冒号前面要有选择器，表示在这个选择器下，伪类这种情况**

:first-child 第一个子元素

:last-child 最后一个子元素

:nth-child() 选中第n个子元素，

**括号里面写几，就是第几个子元素，写n的话，就代表从0到正无穷，2n或even就代表选中偶数位的元素，2n+1或odd就代表选中奇数的元素**

```
/* ul > li:first-child{
    color: red;
} */

/* ul > li:last-child{
    color: red;
} */

/* ul > li:nth-child(2n+1){
    color: red;
} */

/* ul > li:nth-child(even){
    color: red;
} */
```

以上这些伪类都是根据**所有的子元素进行排序**,意思是ul元素名为li的子元素,并且它在所有的子元素中处于第一个位置,就选择它,不是处于第一个位置就不符合要求

如下并没有效果

```
ul > li:first-child{
    color: red;
```

- <ul>
- <span>我是一个span</span>
- <li>第〇个</li>
- <li>第一个</li>
- <li>第二个</li>
- <li>第三个</li>
- <li>第四个</li>
- <li>第五个</li>
- </ul>

我是一个span

- 第〇个
- 第一个
- 第二个
- 第三个
- 第四个
- 第五个

相对应有如下几个伪类:

**:first-of-type**

**:last-of-type**

**:nth-of-type**

这几个伪类的功能和上述的类似，不通点是他们是在同类型元素中进行排序，如下就没有上面那种问题

```
ul > li:first-of-type{  
    color: red;  
}
```

- 我是一个span
- 第〇个
  - 第一个
  - 第二个
  - 第三个
  - 第四个
  - 第五个

:not() 否定伪类

括号里面放伪类

将符合条件的伪类从前面的选择器中去除

```
ul > li:not(:nth-child(3)){  
    color: yellowgreen;  
}
```

- 我是一个span
- 第〇个
  - 第一个
  - 第二个
  - 第三个
  - 第四个
  - 第五个

```
ul > li:not(:nth-of-type(3)){  
    color: yellowgreen;  
}
```

- 我是一个span
- 第〇个
  - 第一个
  - 第二个
  - 第三个
  - 第四个
  - 第五个

补充：

:focus 获取焦点伪类（比如点击一下文本输入框，文本框就获取了焦点）

## a元素的伪类

**:link** (链接独有的伪类)

用来表示没访问过的链接 (所有正常的链接)

```
a:link{  
    color: red;  
}
```

**:visited** (链接独有的伪类)

用来表示访问过的链接

由于隐私的原因，这个visited伪类只能改变连接的颜色

```
a:visited{  
    color: orange;  
    /* font-size: 50px; */  
}
```

**:hover** (所有元素都有的伪类)

用来表示鼠标移入的状态

```
a:hover{  
    color: aqua;  
    font-size: 50px;  
}
```

例子：

```
.location:hover .city-list{  
    display: block;  
}
```

表示鼠标移入location父元素时，设置子元素的值

**:active** (所有元素都有的伪类)

用来表示鼠标点击的状态

```
a:active{  
    color: yellowgreen;  
}
```

## 伪元素选择器

伪元素，表示页面中一些特殊的并不真实的存在的元素（特殊的位置）

伪元素使用**两个冒号::**来开头

**::first-letter**表示第一个字母

```
p::first-letter{  
    font-size: 50px;  
}
```

**::first-line**表示第一行

```
p::first-line{  
    background-color: yellow;  
}
```

**::selection**表示选中的内容

```
p::selection{  
    background-color: greenyellow;  
}
```

**::before**元素的开始，元素内容的最前面，并不是指元素内容本身

**::after**元素的最后

**before**和**after**必须结合**content属性**来使用，before和after区域用鼠标是选不中的

例子：

```
div::before{  
    content: 'abc';  
    color: red;  
}  
  
div::after{  
    content: 'haha';  
    color: red;  
}
```

abcHello Hello How are youhaha

例子：

```
.news-list li::before{  
    content: '■';  
}
```



ps: p标签，该段文字会用双引号引起来，这个双引号就是用before和after来设置的

## 样式的继承

我们为一个元素设置的样式同时也会应用到它的后代元素上

继承是发生在祖先和后代之间的

继承的设计是为了方便我们的开发，利用继承我们可以将一些通用的样式统一设置到共同的祖先元素上

**注意：并不是所有的样式都会被继承**

**如背景、布局相关等的这些样式都不会被继承**

## 选择器的权重

样式的冲突：当我们通过不同的选择器，选中相同的元素，并且为相同的样式设置不同的值时，此时就发生了样式的冲突

发生样式冲突时，应用哪个样式由选择器的权重（优先级）决定

选择器的权重：

**内联样式 1000**

**id选择器 100**

**类和伪类选择器 10**

**元素选择器 1**

**通配选择器 0**

## 继承的样式 没有优先级

比较优先级时，需要将所有的选择器的优先级进行**相加计算**，最后优先级越高，则越优先显示（**分组选择器是单独计算的**）

**但是选择器的累加不会超过其最大的数量级**，比如，几个类选择器再高也不会超过id选择器

如如果优先级计算后相同，此时则优先使用**靠下的样式**

可以在某一个样式的后边添加**!important**，则此时该样式会获取到最高的优先级，甚至超过内联样式

```
.d1{  
    background-color: purple !important;  
}
```

注意：在开发中这个玩意一定要慎用，**能不用就不用**

## 单位

### 长度单位

**像素**

px

屏幕（显示器）实际上是由一个个的小点点构成的

不同屏幕的像素大小是不同的，像素越小的屏幕显示的效果越清晰

所以同样的200px，在不同的设备下显示效果不一样

### 百分比

可以将属性值设置为**相对于其父元素属性的百分比**

设置百分比可以使子元素**跟随父元素的改变而改变**

**em**

em是相对于**元素的字体大小**来计算的

**1em =1font-size**

em会根据**字体大小**的改变而改变

**浏览器默认字体大小为16px**

如下图，这是font-size字体大小设置的就为30px，换算后，长宽就为300px了

```
.box3{  
    font-size: 30px;  
    width: 10em;  
    height: 10em;  
    background-color: greenyellow;  
}
```

## rem

rem是相对于根元素（HTML）的字体大小来计算的

根元素字体大小默认的是16px

## 颜色单位

### 颜色单位：

在css中可以直接使用颜色名来设置各种颜色

比如：red、orange、yellow、blue、green ... ...

但是在css中直接使用颜色名是非常的不方便

### RGB值：

- RGB通过三种颜色的不同浓度来调配出不同的颜色
- R red, G green , B blue
- 每一种颜色的范围在 0 - 255 (0% - 100%) 之间
- 语法：RGB(红色,绿色,蓝色)

### RGBA：

- 就是在rgb的基础上增加了一个a表示不透明度
- 需要四个值，前三个和rgb一样，第四个表示不透明度  
1表示完全不透明 0表示完全透明 .5半透明

### 十六进制的RGB值：

- 语法：#红色绿色蓝色
- 颜色浓度通过 00-ff
- 如果颜色两位两位重复可以进行简写  
#aabbc --> #abc

```
background-color: red;  
background-color: rgb(255, 0, 0);  
background-color: rgb(0, 255, 0);  
background-color: rgb(0, 0, 255);  
background-color: rgb(255,255,255);  
background-color: rgb(106,153,85);  
background-color: rgba(106,153,85,.5);  
background-color: #ff0000;  
background-color: #ffff00;  
background-color: #ff0;  
background-color: #bfa;
```

hsl值用的不多

HSL值	HSLA值
H 色相(0 - 360)	
S 饱和度, 颜色的浓度 0% - 100%	
L 亮度, 颜色的亮度 0% - 100%	

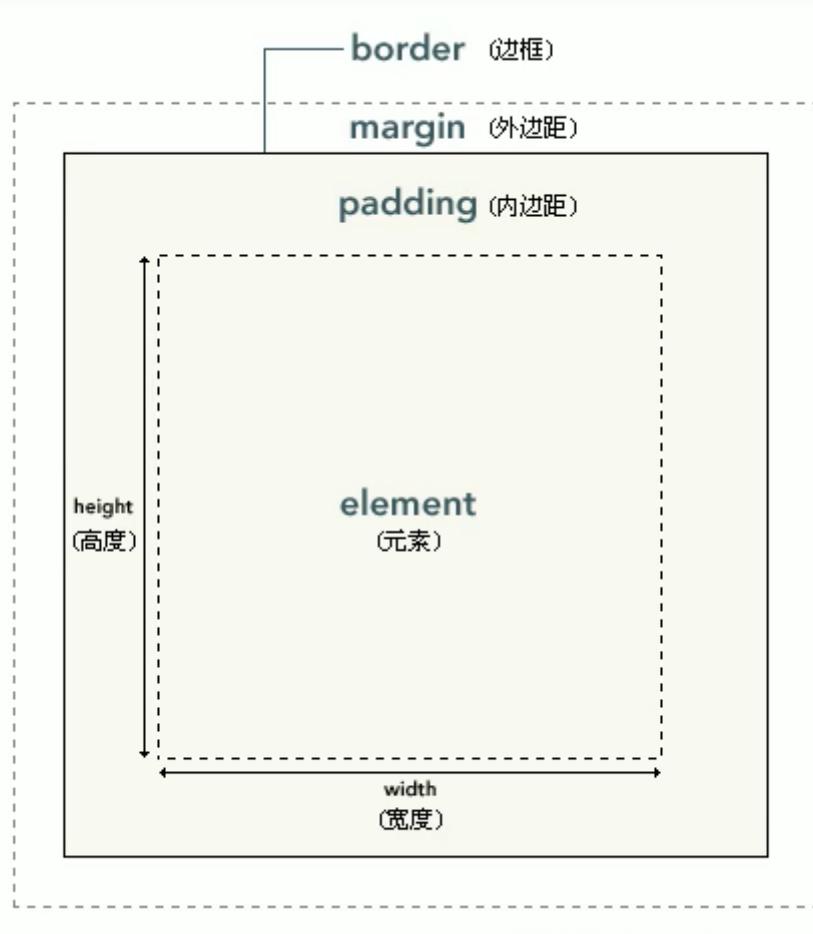
```
background-color: hsla(98, 48%, 40%, 0.658);
```

文档流

### 文档流 (normal flow)

- 网页是一个多层的结构，一层摞着一层
- 通过css可以分别为每一层来设置样式
- 作为用户来讲只能看到最顶上一层
- 这些层中，最底下的一层称为文档流 文档流是网页的基础  
我们所创建的元素默认都是在文档流中进行排列
- 对于我们来元素主要有两个状态
  - 在文档流中
  - 不在文档流中（脱离文档流）
- 元素在文档流中有什么特点：
  - 块元素
    - 块元素会在页面中独占一行(自上向下垂直排列)
    - 默认宽度是父元素的全部 (会把父元素撑满)
    - 默认高度是被内容撑开 (子元素)
  - 行内元素
    - 行内元素不会独占页面的一行，只占自身的大小
    - 行内元素在页面中左向右水平排列，如果一行之中不能放下，则元素会换到第二行继续自左向右排列 (书写习惯一致)
    - 行内元素的默认宽度和高度都是被内容撑开

## 盒子模型box model



内容区 ( content)

内边距 ( padding)

边框 ( border)

外边距 ( margin)

共盒子的**可见框的大小**, 由**内容区、内边距、边框**共同决定, 所以在计算盒子大小时, 需要将这三个区域加到一起计算

实际占地大小还要考虑外边距

**内容区 ( content)**

元素中的所有的子元素和文本内容都在内容区中排列

内容区的大小由 width和 height两个属性来设置

**width**设置内容区的宽度

**height**设置内容区的高度

**边框 ( border)**

边框属于**盒子边**, 边框里边属于盒子内部, 出了边框都是盒子的外部, **边框的大小会影响到整个盒子的大小**

要设置边框, 需要**至少**设置以下三个样式: **border-width**, **border-color**, **border-style**, 缺一不可

## 边框的宽度 border-width

只有这一个值不写的话也可以，宽度也是存在的，**默认值为3px**

可以最多用四个值指定四个方向的宽度，值与值之间用**空格隔开**

四个值：上 右 下 左  
三个值：上 左右 下  
两个值：上下 左右  
一个值：上下左右 I

```
border-width: 10px 20px 30px;
```

除了用border-width，可以用四个属性代替它，分别是**border-\*\*\*-width**， \*\*\*代表的是**top**、**right**、**bottom**、**left**

这四个代替值同样适用于边框颜色和边框样式

## 边框的颜色 border-color

也可以不写，不写的话就会**默认使用color**的值，color代表的是前景色，并不只是代表字体颜色

## 边框的样式 border-style

solid 实线

dotted 点状虚线

dashed 虚线

double 双线

边框样式不设置就默认为**none**，就没有边框了

**border属性：**简写**border**的各种样式，就不用写三个了，一个属性三个值，且**没有顺序**

```
border简写属性，通过该属性可以同时设置边框所有的相关样式，并且没有顺序要求
*/
border: solid 10px orange;
```

同样为了方便设置四个边框，有四个分开写的属性：

除了**border**以外还有四个 **border-xxx**  
**border-top**  
**border-right**  
**border-bottom**  
**border-left**

## 内边距 ( padding)

内容区和边框之间的距离

内边距的设置会影响到盒子的大小

有四个方向的内边距属性：

**padding-top**

**padding-right**

**padding-bottom**

**padding-left**

```
padding-top: 100px;
padding-left: 100px;
padding-right: 100px;
padding-bottom: 100px;
```

### 简写属性: padding

```
' padding 内边距的简写属性，可以同时指定四个方向的内边距
|   规则和border-width 一样
*/
```

```
padding: 10px 20px 30px 40px;
```

## 外边距 ( margin)

```
/*
外边距 (margin)
- 外边距不会影响盒子可见框的大小
- 但是外边距会影响盒子的位置
- 一共有四个方向的外边距:
  margin-top
    - 上外边距, 设置一个正值, 元素会向下移动
  margin-right
    - 默认情况下设置margin-right不会产生任何效果
  margin-bottom
    - 下外边距, 设置一个正值, 其下边的元素会向下移动
  margin-left
    - 左外边距, 设置一个正值, 元素会向右移动

    - margin也可以设置负值, 如果是负值则元素会向相反的方向移位

- 元素在页面中是按照自左向右的顺序排列的,
  所以默认情况下如果我们设置的左和上外边距则会移动元素自身
  而设置下和右外边距会移动其他元素

- margin的简写属性
  margin 可以同时设置四个方向的外边距, 用法和padding一样

- margin会影响到盒子实际占用空间 */

*/
```

## 水平方向的布局

```
元素的水平方向的布局:
元素在其父元素中水平方向的位置由以下几个属性共同决定“
  margin-left
  border-left
  padding-left
  width
  padding-right
  border-right
  margin-right

一个元素在其父元素中, 水平布局必须要满足以下的等式
margin-left+border-left+padding-left+width+padding-right+border-right+margin-right = 其父元素内容块的宽度 (必须满足)
*/
```

```
0 + 0 + 0 + 200 + 0 + 0 + 0 = 800
0 + 0 + 0 + 200 + 0 + 0 + 600 = 800
- 以上等式必须满足, 如果相加结果使等式不成立, 则称为过度约束, 则等式会自动调整
  - 调整的情况:
    - 如果这七个值中没有为 auto 的情况, 则浏览器会自动调整margin-right值以使等式满足
*/
```

一下这七个属性决定子元素在父元素水平方向的位置

### margin-left

**border-left**  
**padding-left**  
**width**  
**padding-right**  
**border-right**  
**margin-right**

- ① 其中**width、 margin-left、 margin-right**可以设定为auto，谁是auto就调整谁  
② 如果都不设置auto的话，**width**会首先默认为auto，能调整为最大就调整为最大，其他的先默认为0

如果都设置为auto的话，**width**会首先能调整为最大就调整为最大，其他的先默认为0

③ 如果都设置了具体的值的话，**margin-right**无论设置成多少，都会自动被修改成符合等式的值，哪怕margin-left 和width的值超过了父元素，此时margin-right强项修改为负数（但看上去子元素是将父元素顶出来了）。所以margin-right的值设置的是没有意义的

- ④ 两个外边距设置为auto,宽度固定值，则会将外边距设置为相同的值

这七个值中有三个值和设置为auto

**width**  
**margin-left**  
**margin-right**  
- 如果某个值为auto，则会自动调整为auto的那个值以使等式成立  
|  $0 + 0 + 0 + \text{auto} + 0 + 0 + 0 = 800$  auto = 800

有时会利用上面第④条的特点来让一个元素在其父元素居中，width设置为一个具体的值，左右margin设置为auto就行

示例：  
width:xxxxpx;  
margin:0 auto;

## 垂直方向的布局

父元素不设置大小的话，默认会被子元素撑开

父元素最下面的子元素的margin-bottom 不会将父元素的大小撑开

竖直方向和水平方向如果子元素的大小超过了父元素，则子元素会从父元素中溢出

## overflow属性

使用 overflow属性来设置父元素如何处理溢出的子元素

该属性有三个可选值：

**visible**: 默认值子元素会从父元素中溢出，在父元素外部的位置显示

**hidden**: 溢出内容将被裁剪不会显示

**scroll**: 生成两个滚动条，通过滚动条来查看完整的内容

**auto**: 根据水平和竖直方向需要自动生成滚动条，不需要就不用生成

**overflow-x**、**overflow-y**这两个属性可以用上述可选值单独处理水平竖直方向的滚动条

*ps:hidden 和 auto 可以来开启元素的BFC*

## 外边距的折叠

相邻的垂直方向的外边距的会发生折叠现象，指margin-bottom和margin-top这两个属性

## 兄弟元素

两个都是正值

### - 兄弟元素

- 兄弟元素间的相邻垂直外边距会取两者之间的较大值（两者都是正值）

两个值中存在负值

#### 特殊情况：

如果相邻的外边距一正一负，则取两者的和

如果相邻的外边距都是负值，则取两者中绝对值较大的

## 父元素

### - 父子关系

- 父子元素间相邻外边距，子元素的会传递给父元素（上外边距）

- 父子外边距的折叠会影响到页面的布局，必须要进行处理

第一行的意思是，子元素在父元素中设置margin-top后，调整margin-top的值，相当于也在调整父元素的margin-top，这叫做传递，子元素和父元素margin-top成了一个

## 行内元素的盒模型

行内元素不支持设置宽度width和高度height

行内元素可以设置 padding,但是垂直方向 padding,不会影响页面的布局

行内元素可以设置 border,垂直方向的 border不会影响页面的布局

行内元素可以设置 margin,垂直方向的 margin/不会影响布局，水平方向的margin就是简单地相加

**display属性**用来设置元素显示的类型

可选值：

**inline**: 将元素设置为行内元素

**block**: 将元素设置为块元素

**inline-block**: 将元素设置为行内块元素。既可以设置宽度和高度又不会独占一行

**table**: 将元素设置为一个表格(在解决高度塌陷的时候会用到)

table-cell

**none**: 元素不在页面中显示，也不占地方

**visibility属性**用来设置元素的显示状态

可选值：

visible: 默认值，元素在页面中正常显示

hidden: 元素在页面中隐藏不显示，但是依然占据页面的位置

## 浏览器的默认样式

默认样式

通常情况，浏览器都会为元素设置一些默认样式

比如：body默认有外边距

p默认有行间距

ul默认有左内边距，且会有左边会有黑点

通常情况下编写网页时必须要**去除浏览器的默认样式**(PC端的页面)

可以自己手动去除浏览器的默认样式

```
body{  
    margin: 0;  
}  
  
p{  
    margin: 0;  
}  
  
ul{  
    margin: 0;  
    padding: 0;  
    /* 去除项目符号 */  
    list-style:none;  
}  
  
.box1{  
    width: 100px;  
    height: 100px;  
    border: 1px solid black;  
}
```

其中ul标签的list-style:none 可以将默认的那个左边黑点去掉

**最常用的方式**

```
*{  
    margin: 0;  
    padding: 0;  
}
```

## 真正做项目常用的方法

或者直接上网搜现成的写好的去除默认样式的css文件，直接link标签引入

## 盒子的大小

默认情况下，盒子可见框的大小由内容区、内边距和边框共同决定

**box-sizing属性**：用来设置**盒子尺寸的计算方式**(设置width和height的作用)

可选值：

**content-box**：**默认值**，宽度和高度用来设置内容区的大小，可见框的大小还要加上padding和border

**border-box**：**宽度和高度用来设置整个可见框的大小**，width和height指的是内容区和内边距和边框的总大小

## 轮廓和圆角

### outline属性

用来设置元素的轮廓线，用法和 border 模一样

轮廓和边框不同的点，就是轮廓不会影响到可见框的大小

就相当于描了一个边，只是显示了一个效果，不影响元素的盒子，不影响页面的布局

例子：

```
outline: 10px red solid;
```

### box-shadow属性

用来设置元素的阴影效果，也就是一个效果，不影响元素的盒子和页面的布局

属性由四个值设定

①水平偏移量

设置阴影的水平位置，正值向右移动，负值向左

②垂直偏移

设置阴影的水平位置，正值向下移动，负值向上

③阴影的模糊半径

④阴影的颜色，一般制成可透明的颜色

```
background-color: #008000;  
  
/* box-shadow 用来设置元素的阴影效果，阴影不会影响页面布局  
第一个值 水平偏移量 设置阴影的水平位置 正值向右移动 负值  
第二个值 垂直偏移量 设置阴影的水平位置 正值向下移动 负值  
第三个值 阴影的模糊半径  
第四个值 阴影的颜色  
*/
```

```
box-shadow: 20px 20px 50px 0px rgba(0, 0, 0, .5);
```



### border-radius属性

可以分别指定四个角的圆角，也可设置成椭圆

属性的值是角的半径大小，或者椭圆的两个半径

```
border-radius 可以分别指定四个角的圆角  
四个值 左上 右上 右下 左下  
三个值 左上 右上/左下 右下  
两个个值 左上/右下 右上/左下  
*/
```

指定椭圆角用/隔开一个角的两个椭圆半径，四个角之间的值用空格隔开

```
/*  
border-radius: 20px / 40px;  
  
border-radius: 10px 20px 30px;
```

也可以用以下四个值分别去指定四个角的大小

```
/* border-top-left-radius: */  
/* border-top-right-radius */  
/* border-bottom-left-radius: */  
/* border-bottom-right-radius: */
```

分别指定的话，椭圆半径角用两个值来表示，用空格隔开，就不用/了

直接将border-radius 设置成50%，元素就成了一个圆

```
/* 将元素设置为一个圆形 */  
border-radius: 50%;
```



## 浮动的简介

通过浮动可以使一个元素向其父元素的左侧或右侧移动

使用**float属性**，即将float设置成非none的值，就相当于开启了浮动属性

可选值：

**none**: 默认值，元素不浮动

**left**: 元素向左浮动

**right**: 元素向右浮动

**注意：元素设置浮动以后，水平布局的等式便不需要强制成立**

**元素设置浮动以后，会完全从文档流中脱离，不再占用文档流的位置**

**所以元素下边的还在文档流中的元素会自动向上移动**

**一行占不下会自动排列到下一行**

浮动的特点：

- 1、浮动元素会完全脱离文档流，不再占据文档流中的位置
- 2、设置浮动以后元素会向父元素的左侧或右侧移动，
- 3、浮动元素默认不会从父元素中移出
- 4、浮动元素向左或向右移动时，不会超过它前边的其他浮动元素
- 5、如果浮动元素的上边是一个没有浮动的块元素，则浮动元素无法上移
- 6、浮动元素不会超过它上边的浮动的兄弟元素，最多就是和它一样高

**第六点是，浮动元素不会超过他代码中的上一个兄弟元素**

例子：

下面这三个div都设置了浮动，都是兄弟元素，代码中的先后顺序是绿、橙、黄，绿、橙向左浮动，黄向右浮动，也页面占不下，橙和黄就换到了下一行，但是这时候**黄不会向上移动**，因为上面讲到的第六点



### 浮动对于文字有一个特点

浮动元素**不会盖住文字**，文字会自动环绕在浮动元素的周围，如果是两个div就互相盖住了  
所以我们可以利用浮动来设置文字环绕图片的效果

### 脱离文档流的特点

元素设置浮动以后，将会从文档流中脱离，，元素的一些特点也会发生改变

#### 脱离文档流的特点

块元素：

- 1、块元素**不再独占页面的一行**
- 2、脱离文档流以后，块元素的**宽度和高度默认都被内容撑开**

行内元素：

行内元素脱离文档流以后会**变成块元素**，特点和块元素一样，就可以进行width和height等属性的设定了

**脱离文档流以后，不需要再区分块和行内了**

### 高度塌陷问题和BFC

#### 高度塌陷问题

在浮动布局中，很多情况下，页面布局时，**父元素的高度会故意默认被子元素撑开**，这是为了方便让父元素中的子元素进行增减，并且做到整体布局合理

但是当**子元素浮动后**，子元素从文档流中脱离，将会无法撑起父元素的高度，**导致父元素的高度丢失**

父元素高度丢失以后，其下的元素会自动上移，导致页面的布局混乱

## BFC

BFC (Block Formatting Context) 块级格式化环境

BFC是一个CSS中的一个隐含的属性，可以为一个元素开启BFC 开启BFC该元素会变成一个**独立的布局区域**

元素开启BFC后的

1.开启BFC的**元素不会被浮动元素所覆盖**

2.开启BFC的元素**子元素和父元素外边距不会重叠**

3.开启BFC的元素**可以包含浮动的子元素**

可以通过一些**特殊方式**来开启元素的BFC:

1、设置元素的浮动（不推荐，有一些副作用）

2、将元素设置为行内块元素（不推荐，有一些副作用）

使用**display:inline-block;**

3、常用的方式为元素设置 **overflow:hidden**开启其BFC以使其可以包含浮动元素，**也可以用overflow中的auto**（副作用较小）

## clear

作用：**清除浮动元素对当前元素所产生的影响**

可选值

**left**: 清除左侧浮动元素对当前元素的影响

**right**: 清除右侧浮动元素对当前元素的影响

**both**: 清除两侧中**影响最大的那侧**

原理：设置清除浮动以后，浏览器会自动为元素添加一个上外边距，以使其位置不受其他元素的影响

**将它写在需要消除影响的那个元素中，不是写在浮动的元素中**

```
clear: both;
```

## clear解决高度塌陷

一种添加一个div标签的方式来解决高度塌陷问题，但不算太好，应该用css的方法来解决css的问题，不能依靠HTML

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .box1{
            border: 10px solid red;
            /* overflow: hidden; */
        }

        .box2{
            width: 100px;
            height: 300px;
            background-color: #bfa;
            float: left;
        }

        .box3{
            clear: both;
        }

    </style>
</head>
<body>

    <div class="box1">
        <div class="box2"></div>
        <div class="box3">
            aa
        </div>
    </div>

</body>

```

## 最好方案

```

04_Float > 07.高度塌陷的最终解决方案.html > HTML > head > style > .box1
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Document</title>
8     <style>
9         .box1{
10             border: 10px solid red;
11             /* overflow: hidden; */
12         }

13         .box2{
14             width: 100px;
15             height: 100px;
16             background-color: #bfa;
17             float: left;
18         }

19         .box3{
20             clear: both;
21         }

22         .box1::after{
23             content: '';
24             display: block;
25             clear: both;
26         }
27     </style>
28 </head>
29 <body>

30     <div class="box1">
31         <div class="box2"></div>
32         <!-- <div class="box3"></div> -->
33     </div>
34 </body>
35 </html>

```

标准格式：

```

.box1::after{
    content: '';
    display: block;
    clear: both;
}

```

在高度塌陷的父元素中，写一个**伪元素选择器**，为父元素后面的**after**设置属性

after默认为行元素，用display:block转化为块元素

content不能少，after和before必须结合content来使用，只不过不用往content里面写内容

然后在after的伪元素选择器中设定clear:both

这样父元素的下方有一个不受浮动元素影响的块元素，并且始终在父元素的最后面，父元素里面的所有浮动元素，为了不影响after，就会消除高度塌陷问题

## 终极方案

```
.clearfix::before,  
.clearfix::after{  
    content: '';  
    display: table;  
    clear: both;  
}
```

这是一个很经典的代码，就可以将父子顶部外边距重叠的问题解决，也可以将高度塌陷问题解决

自定义一个类放到需要解决高度塌陷的父元素，然后设置这个类的after和before伪元素选择器，内容原理和上面基本一样，只不过display值是table，用table不用block是因为table既可以解决高度塌陷，又可以解决外边距重叠的问题

```
.clearfix::before,  
.clearfix::after{  
    content: '';  
    display: table;  
    clear: both;  

```

## 定位

定位( position): 定位是一种更加高级的布局手段，通过定位可以将元素摆放到页面的任意位置

使用 **position属性** 来设置定位

可选值：

**static**: 默认值，元素是静止的没有开启定位

**relative**: 开启元素的**相对定位**

**absolute**: 开启元素的**绝对定位**

**fixed**: 开启元素的**固定定位**

**sticky**: 开启元素的**粘滞定位**

### 相对定位

相对定位

position属性值为 **relative**

相对定位的特点

1.元素开启相对定位以后，如果不设置偏移量**元素不会发生任何的变化**，任何属性都没变

2. 相对定位是参照与元素最一开始在文档流中的位置，也就是定位位置

3. 相对定位会提升元素的层级

4. 相对定位不会使元素脱离文档流

5. 相对定位不会改变元素的性质，块还是块，行内还是行内

### 偏移量( offset)

当元素开启了定位以后，可以通过偏移量来设置元素的位置

**top**: 定位元素和定位位置上边的距离

**bottom**: 定位元素和定位位置下边的距离

**left**: 定位元素和定位位置的左侧距离

**right**: 定位元素和定位位置的右侧距离

top/bottom、left/right这两组分别用一个就行了

## 绝对定位

position属性值为 **absolute**

绝对定位的特点

1. 开启绝对定位后，如果不设置偏移量，元素的暂且位置不会发生变化，但此时不代表原位就是偏移量为0的位置

2. 绝对定位后，元素从文档流中脱离，符合脱离文档流的一切特点（详见上方脱离文档流的特点）

3. 绝对定位会改变元素的性质，行内元素变成块，块的宽高被内容撑开，不在独占一行（实际上就是第二点）

4. 绝对定位会使元素提升一个层级

5. 绝对定位是相对于其包含块来进行定位的

### 包含块( containing block)

正常情况下：

包含块就是离当前元素最近的祖先块元素

绝对定位的包含块：

包含块就是离它最近的开启了定位 (position属性的值非static) 的祖先元素

如果所有的祖先元素都没有开启定位，则根元素HTML标签就是它的包含块

## 固定定位

**position属性为 fixed**

固定定位也是一种绝对定位，所以固定定位大部分的特点都和绝对定位一样

**唯一不同的是固定定位永远参照于浏览器的视口进行定位（也就是屏幕左上角，并不是页面左上角，滚动滑轮位置也不会变），不管包含块，也就是固定的意思**

## 粘滞定位

粘滞定位

**position属性为 sticky**

粘滞定位和相对定位的特点基本一致，定位是按照相对定位的方式

**不同的是粘滞定位可以在元素到达某个位置时将其固定**

页面向下滚动时，元素到达指定的位置就会不动了，再滚回去就恢复正常

但是常见的粘滞导航栏一般不用粘滞定位，因为**兼容性不好（IE就不可用）**，一般都结合js来达到目的

## 绝对定位的元素布局

### 水平方向

之前讲的水平布局由7个属性来进行设置，如下：

**margin-left**

**border-left**

**padding-left**

**width**

**padding-right**

**border-right**

**margin-right**

开启绝对定位后，在加上**right**和**left**这两个属性一共9个

这9个值加起来前置等于父元素的宽度

如果9个属性中没有值为auto时，则自动调整**right**值以使等式满足（之前一直是margin-right）

如果有auto，则自动调整auto的值以使等式满足

**可设置auto值的有以下属性**

**margin width left right**

因为left和right的值默认是auto，所以如果不知道left和right

像之前讲的，想设置子元素居中，让margin-left和margin-right和auto就行了，但是现在不行，必须写上right:0和left:0，这两个值如果不指定的话，会自动调整这两个值

则等式不满足时，会自动调整这两个值

## 垂直方向

跟之前的不一样，开启绝对定位后，垂直方向也必须满足

top

margin-top

margin-bottom

padding-top

padding-bottom

border-top

border-bottom

height

bottom

这些属性的值加起来也必须满足等于包含块的高度

利用这个同样可以实现垂直居中

设定好高和宽，让top:0;和bottom:0;，并且将margin-top:0和margin-bottom:0，就能垂直居中，原理类似。当然也能垂直水平双居中

## 元素的层级

对于开启了定位元素，可以通过z-index属性来指定元素的层级

z-index需要一个整数作为参数，值越大元素的层级越高

元素的层级越高越优先显示

如果元素的层级一样，则优先显示靠下的元素

祖先的元素的层级再高也不会盖住后代元素

只要是定位，默认层级就一样高，无论是什么定位，只凭借z-index来比较层级

## 字体

字体相关的样式属性

color用来设置字体颜色

font-size字体大小

和font-size相关的单位：

**em**相当于当前元素的一个font-size

**rem**相对于根元素的一个font-size

## 字体族

**font-family**字体族 (字体的格式类别)

可选值：

serif衬线字体

sans-serif非衬线字体

monospace等宽字体

.....

**指定字体的类别，浏览器会自动使用该类别下的字体**

font-family可以同时指定多个字体，多个字体间使用**逗号隔开**

字体生效时优先使用第一个，第一个无法使用则使用第二个以此类推

一般前几个是具体的字体，最后一个**是字体族**，前面几个没有了，浏览器就在最后的字体族中随便挑一个就行了

```
font-family: 'Courier New', Courier, monospace;
```

有的字体之间有空格，**建议将有空格的字体用引号引起来**，避免发生错误

## font-face

**@font-face属性**

语法：

有两个属性：

**font-family**: 指定字体的名字 (和之前的font-family不同，这个是自己指定名字)

**src:url("")** :服务器中字体的路径

实例：

```
@font-face {
    /* 指定字体的名字 */
    font-family: 'myfont' ;
    /* 服务器中字体的路径 */
    src: url('./font/ZCOOLKuaiLe-Regular.ttf') format("truetype");
}
```

可以将服务器中的属性直接提供给用户去使用

但是由于是用户从服务器上面去下载的字体，**会影响字体的加载速度**

次属性涉及到版权的问题，必须使用已授权的字体，而前面讲的文字样式里面设定的**font-family不考虑版权**，因为那个只是提供一个可选项，真正选哪个由用户的浏览器去选择

实例中的ttf文件是最常用的一个字体文件格式，

为了防止有的文件格式在浏览器中加载不出来，会指定多个字体文件格式

```
@font-face {  
    font-family: "Open Sans";  
    src: url("/fonts/OpenSans-Regular-webfont.woff2") format("woff2"),  
         url("/fonts/OpenSans-Regular-webfont.woff") format("woff");  
}
```

format一般不用写，是用来告诉浏览器这个字体文件是什么格式的，不同字体文件对应不同的格式，一般不写，如下

The available types are: "woff", "woff2", "truetype", "opentype", "embedded-opentype", and "svg".

## 图标字体

图标字体

在网页中经常需要使用一些图标，可以通过图片来引入图标

但是图片大小本身比较大，并且非常的不灵活

所以在使用图标时，我们还可以将图标直接设置为字体

然后通过@font-face的形式来对字体进行引入

这样我们就可以通过使用字体的形式来使用图标

## fontawsesome

fontawesome使用

1.下载下来后，一般就将css和 webfonts移动到项目中就可以

2..将all.css引入到网页中

```
<link rel="stylesheet" href="./fa/css/all.css">
```

3..使用图标字体：直接通过类名来使用图标字体

下载下来的css里面已经定义好类名了，css里面也写入了@font-face，并且设定好样式了，我们只需要为标签绑定类名就行，一般常用的就是*标签*，

类名有指定格式：fas/fab 字体图标的名字

一般就fas和fab这两个免费的类型

```
<i class="fas fa-bell"></i>  
<i class="fas fa-bell-slash"></i>  
<i class="fab fa-accessible-icon"></i>
```

## 字体图标其他使用方式

①

通过伪元素来设置图标字体

1. 找到要设置图标的元素通过 before 或 after 伪元素选择器选中

```
li::before{
```

2. 在伪元素选择器中的 content 中设置字体的编码，需要加上反斜杠（官网上，或者文档中有每个图标对应的编码）

```
li::before{  
    content: '\f1b0';
```

3. 在伪元素选择器中指定字体的 font-family，有时也要设置字体的样式，基本上就分为两种情况，fab 和 fas 类型的不同，下面的格式直接记住就行

```
/* font-family: 'Font Awesome 5 Brands'; */  
font-family: 'Font Awesome 5 Free';  
font-weight: 900;
```

**fab:**

```
font-family: 'Font Awesome 5 Brands';
```

**fas:**

```
font-family: 'Font Awesome 5 Free';
```

```
font-weight: 900;
```

完整例子

```
front&background > 03.图标字体.html > html > body > ul > li
```

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Document</title>
8      <link rel="stylesheet" href="./fa/css/all.css">
9      <style>
10         li{
11             list-style: none;
12         }
13
14         li::before{
15             content: '\f1b0';
16             /* font-family: 'Font Awesome 5 Brands'; */
17             font-family: 'Font Awesome 5 Free';
18             font-weight: 900;
19             color: blue;
20             margin-right: 10px;
21         }
22     </style>
23 </head>
24 <body>
25
26     <!-- <i class="fas fa-cat"></i> -->
27
28     <ul>
29         <li>锄禾日当午</li>
30         <li>汗滴禾下土</li>
31         <li>谁知盘中餐</li>
32         <li>粒粒皆辛苦</li>
33         <li>粒粒皆辛苦</li>
34         <li>粒粒皆辛苦</li>
35         <li>粒粒皆辛苦</li>
36         <li>粒粒皆辛苦</li>
37     </ul>
38
39 </body>
40 </html>
```

② 锄禾日当午  
汗滴禾下土  
谁知盘中餐  
粒粒皆辛苦  
粒粒皆辛苦  
粒粒皆辛苦  
粒粒皆辛苦  
粒粒皆辛苦

②

再或者，用之前讲过的**字符实体**来设置字体图标

**字符实体语法：&实体的名字**

在这里语法：**&#x图标编码**

但类名**只用写fab或者fas**就行了

```
<span class="fas">&#xf0f3;</span>
```

## iconfont

使用方法：

1.网上打包下载字体项目文件

2.用link标签引入iconfont.css的css文件

```
<link rel="stylesheet" href="./iconfont/iconfont.css">
```

3.为需要的元素添加名为iconfont的类名

```
<i class="iconfont">&#xe61c;</i>
```

4.在元素中写入文档中指定的图标编码

```
<i class="iconfont">&#xe61c;</i>
<i class="iconfont">&#xe622;</i>
<i class="iconfont">&#xe623;</i>
```

5.设置图标的样式需要指明iconfont的类名，选择器只说明标签名不行，因为css文件里面设定了iconfont类的样式，就会覆盖元素本身设置的样式

```
style>
  i.iconfont{
    font-size: 100px;
  }
```

ps：图标库中还为每一个图标设置了指定的类名，使用具体的类名，就不用写编码来指定了，如下就可以

```
<i class="iconfont icon-qitalaji"></i>
```

同样也可以使用伪元素选择器来设置样式，同之前的方法

## 字体的简写属性

**font属性**可以设置字体的所有相关属性

之前**字体大小**和**字体族**分开写

```
font-size: 50px;  
font-family: 'Times New Roman', Times, serif;
```

现在可以写在一起了

不同内容的值用**空格隔开**，各自的语法还是不变

**语法：**

**font:字体大小/行高 字体族**

```
font:50px/2 'Times New Roman', Times, serif;
```

```
font:50px 'Times New Roman', Times, serif;
```

ps:

①/**并不是或**的意思

②行高是可选值，可写可不写，**但是不写不意味着没有，有一个具体的默认值**，之前如果设置了行高的话，**会被覆盖掉**

font属性中在字体大小和字体族前面，还可以选写**italic (斜体)** 和**bold (加粗)**，顺序无所谓，不写就是默认值

```
font: italic bold 50px/2 'Times New Roman', Times, serif;
```

## font-weight和font-style

**font-weight属性**：字重（用来设置字体的加粗）

可选值：

**normal**：默认值不加粗

**bold**：加粗

100-900九个级别（**没什么用**）

**font-style属性**: 字体的风格 (用来设置字体的倾斜)

可选值:

**normal**: 正常的

**italic**: 斜体

## 行高

**行高(line height)**

行高指的是文字占有的实际高度

可以通过**line-height属性**来设置行高

行高可以直接指定一个大小(px、em)

也可以直接为行高设置一个整数

**如果是一个整数的话，行高将会是字体的指定的倍数**

多行的行高不会进行重叠

**字体框**

字体框就是字体存在的格子，使用**font-size属性**来设置字体框的高度，即**在调整字体的大小**

如果父元素不设置height高度的话，多余的行高会在**字体框**的上下平均分配，这也就是之前讲到的

**垂直居中设置方法**: 将行高 (line-height) 和父元素高度 (height) 设置成一样的，再将行高和**字体框** (font-size) 的大小设置一样，或者直接不设置父元素的height，直接由行高来撑起来就行

可以经过换算后得，**行间距=行高 - 字体大小**

## 文本的水平和垂直对齐

### 水平对齐

**text-align属性**: 文本的水平对齐

可选值:

left: 左侧对齐, 默认值

right: 右对齐

center: 居中对齐

justify: 两端对齐

### 垂直对齐

一般适用于一行中不同元素中的文字, 这样垂直方向才会有偏差, 如图:

```
<div>
|   今天天气 Hello<span>真不错 Hello</span>!
</div>
```

**vertical-align属性**: 设置元素垂直对齐的方式

可选值:

baseline: 默认值, 基线对齐, 并非元素的最底部

top: 顶部对齐, 元素的顶部

bottom: 底部对齐, 元素的底部

middle: 居中对齐

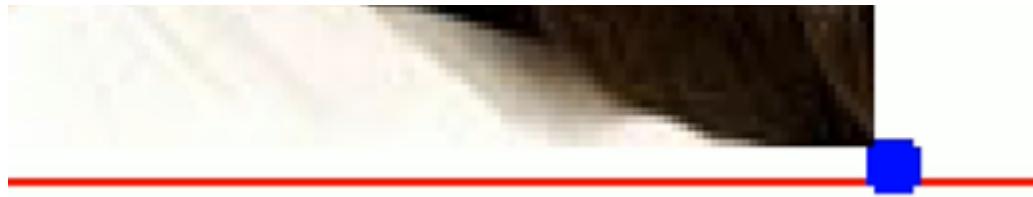
指定数值, 如: 100px, 他会往基线的上方移动100px的距离

例子:

```
span{
    font-size: 20px;
    border: 1px solid blue;

    /*
        vertical-align 设置元素垂直对齐的方式
        可选值:
            baseline 默认值 基线对齐
            top 顶部对齐
            bottom 底部对齐
            middle 居中对齐
    */
    vertical-align: middle;
}
```

元素引入一个图片，这个图片同样也是默认基线对齐，所以图片并没有对准元素的底部，会有一个缝隙



同样可以用**垂直对齐的属性vertical-align**来消除，在图片的样式中添加vertical-align使vertical-align:bottom或者top，就可以消除缝隙

## 文本的样式

**text-decoration属性**: 设置文本修饰

可选值:

**none**: 什么都没有

**underline**: 下划线

**line-through**: 删除线

**overline**: 上划线

```
text-decoration: line-through;
```

在上面这些值的后面还可以指定**颜色**和**线的样式**，用**空格**隔开，但是颜色和线的样式**IE不支持**

```
text-decoration: underline red dotted;
```

**white-space属性**: 设置网页如何处理空白

可选值:

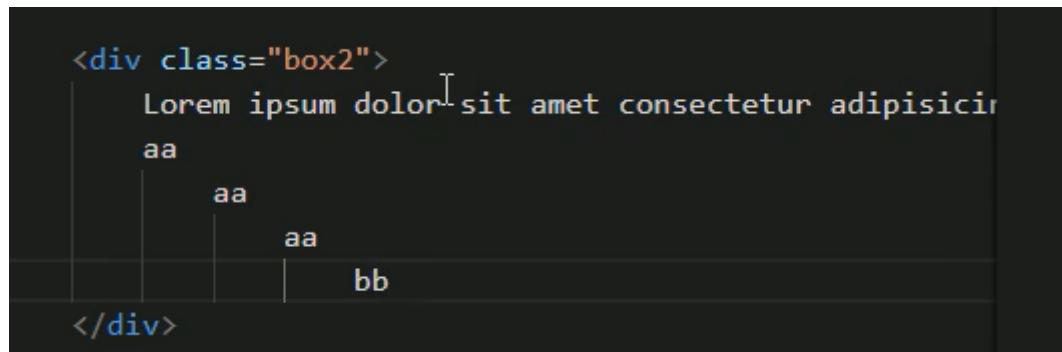
**normal**: 正常

**nowrap**: 不换行

**pre**: 保留空白 (会保留元素内文本的原来格式, 换行和空格都会保留, 不会改变)

pre效果:

```
        white-space: pre;
```



```
<div class="box2">
    Lorem ipsum dolor sit amet consectetur adipisicing
    aa
        aa
        aa
            bb
</div>
```

```
Lorem ipsum dolor sit amet consectetur
aa
aa
aa
bb
```

**text-overflow属性**: 设置文本溢出的样式

text-overflow:ellipsis 将溢出的边缘使用省略号

前提是: white-space属性的值是nowrap (不换行)

overflow属性值是hidden

```
        white-space: nowrap;
        overflow: hidden;
        text-overflow: ellipsis;
```

```
Lorem ipsum dolor sit a...
```

## 背景

background-color: 背景颜色

**background-image**: 设置背景图片

同时设置背景颜色和背景图片，背景颜色将会成为图片的背景色

语法: **background-image:url("图片地址");**

```
background-image: url("./img/1.png")
```

如果背景的图片小于元素，则背景图片会自动在元素中平铺将元素铺满，不拉伸

如果背景的图片大于元素，将会一个部分背景无法完全显示

如果背景图片和元素一样大，则会直接正常显示

**background-repeat**: 用来设置背景图片的重复方式

可选值:

repeat: 默认值，背景会沿着x在y双方向重复

repeat-x: 沿着x方向重复

repeat-y: 沿着y方向重复

no-repeat: 背景图片不重复

**background-position**: 用来设置图片在父元素中的位置

通过 top、left、right、bottom、center几个表示方位的值来组合表示父元素九宫格的九个位置

值	描述
top left top center top right center left center center center right bottom left bottom center bottom right	如果您仅规定了一个关键词，那么第二个值将是"center"。 默认值: 0% 0%。
x% y%	第一个值是水平位置，第二个值是垂直位置。 左上角是 0% 0%。右下角是 100% 100%。 如果您仅规定了一个值，另一个值将是 50%。
xpos ypos	第一个值是水平位置，第二个值是垂直位置。 如果您仅规定了一个值，另一个值将是50%。

background-color

background-image

background-repeat

background-position

background-size

**background-clip**: 设置背景的范围

可选值:

**border-box**: 默认值, 背景会出现在边框的下边

**padding-box**: 背景不会出现在边框, 只出现在内容区和内边距

**content-box**: 背景只会出现在内容区

**background-origin**: 背景图片的偏移量计算的原点

可选值:

**padding-box**: 默认值, **background-position**从内边距处开始计算

**content-box**: 背景图片的偏移量从内容区处计算

**border-box**: 背景图片的变量从边框处开始计算

**background-size**: 设置背景图片的大小

可选值:

第一个值表示宽度

第二个值表示高度

如果只写一个, 则第二个值默认是auto

或者写以下两个值:

**cover**: 图片的比例不变, 将元素铺满, 能做到铺满就行, 不用将图片还原成原来的大小

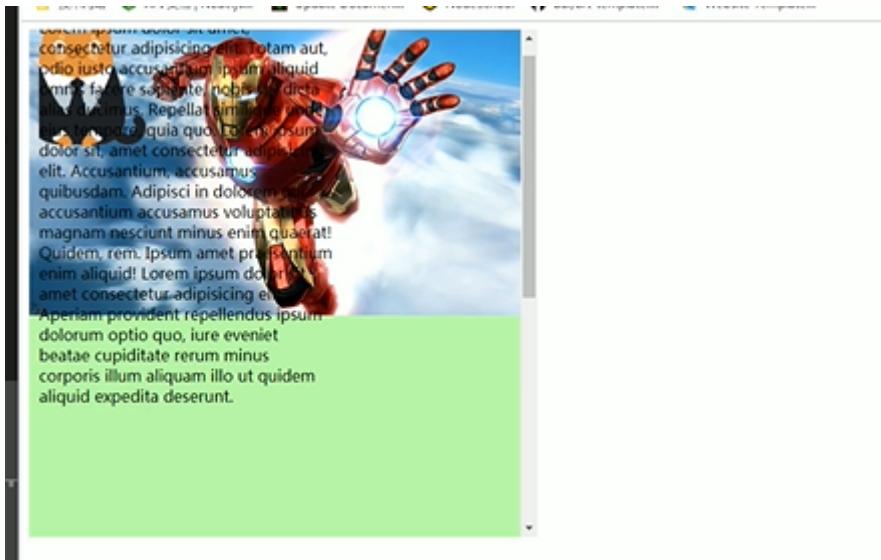
**contain**: 图片比例不变, 将图片在元素中完整展示, 不用将图片过于缩小, 能展示完整就可

**background-attachment**: 背景图片是否跟随元素移动

可选值:

**scroll**: 默认值, 背景图片会跟随元素移动

**fixed**: 背景会固定在页面中, 不会随元素移动



注意，在这里面父元素的背景不会移动，因为父元素本身就没有动，移动的是子元素

**background**: 背景相关的简写属性，所有背景相关的样式都可以通过background来设置

注意：

**background-size**必须写在 **background- positione**的后边，并且使用/隔开：**background-position/background-size**

background- origin和background-clip两个样式，orging要在clip的前边，因为这两个属性的值一样，这么做为了区分

```
.box3{  
    width: 500px;  
    height: 500px;  
    background: url('./img/2.jpg') #bfa center center/contain no-repeat ;  
}
```

## 雪碧图

**网页背景加载的一个实际问题：**

一个按钮点击效果，点击前，点击时，点击后三种状态，三个图片

```
<style>
    a:link{
        display: block;
        width: 93px;
        height: 29px;
        background-image: url('./img/08/link.png')
    }

    a:hover{
        background-image: url('./img/08/hover.png')
    }

    a:active{
        background-image: url('./img/08/active.png')
    }

```



图片属于网页中的外部资源，外部资源都需要浏览器单独发送请求加载

浏览器加载外部资源时是按需加载的，用则加载，不用则不加载

像我们上边的练习link会首先加载，而 hover和 active会在指定状态触发时才会加载，用户第一次点击按钮时会有**卡顿**的体验

## 解决方案

解决图片闪烁的是题

可以将多个小图片统一保存到一个大图片中，然后通过调整 background-position来显示相应的图片

这样图片会同时加载到网页中就可以有效的避免出现烁的问题

**这个技术在css中应用十分广泛，被称为CSS-Sprite（雪碧图）**

```
a:link{  
    display: block;  
    width: 93px;  
    height: 29px;  
    background-image: url('./img/09/btn.png')  
}  
  
a:hover{  
    background-position: -93px 0;  
}  
  
a:active{  
    background-position: -186px 0;  
}
```

雪碧图的使用步骤：

先确定要使用的图标

测量图标的大小

根据测量结果创建一个元素

将雪碧图设置为元素的背景图片

设置一个偏移量以显示正确的图片

雪碧图的特点：

一次性将多个图片加载进页面，降低请求的次数，加快访问速度，提升用户的体验

## 渐变

通过渐变可以设置一些复杂的背景颜色，可以实现从一个颜色向其他颜色过度的效果

**渐变是图片**，需要通过**background-image属性**来设置，对老版本兼容性不好

**linear-gradient()**: 线性渐变，颜色沿着一条直线发生变化

linear-gradient(red, yellow)，红色在开头，黄色在结尾，中间是过渡区域

```
background-image: linear-gradient( red, yellow)
```



线性渐变的开头，我们可以指定一个渐变的方向

to left

to right

to bottom

to top

to top left

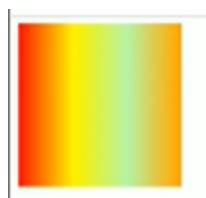
xxxdeg deg表示度数

```
background-image: linear-gradient(45deg, red, yellow)
```

```
background-image: linear-gradient(to top left, red, yellow)
```

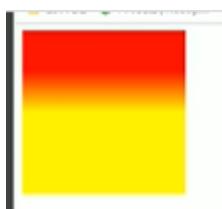
颜色不止两种，也可以指定多种，多种颜色平均分配

```
background-image: linear-gradient(to right, red, yellow, #bfa, orange)
```



可以手动指定渐变的分布情况，在每个颜色的后面指定纯色开始的位置，例如，红色从50px开始发生渐变，黄色从100px完成渐变，后面全是纯色

```
background-image: linear-gradient(red 50px, yellow 100px);
```



**repeating-linear-gradient()**可以平铺的线性渐变

```
background-image: repeating-linear-gradient(red 0px, yellow 50px)
```



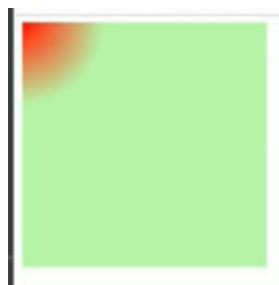
设置完repeating-linear-gradient之后，再设置background-repeat:no-repeat就没有用了

## 径向渐变

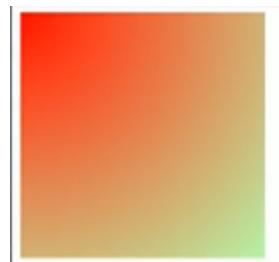
radial-gradient()径向渐变（放射性效果）

设置径向渐变圆的大小，设置径向渐变中心点的位置，语法如下：

```
background-image: radial-gradient(100px 100px at 0 0, red, #bfa)
```



```
background-image: radial-gradient(at top left, red, #bfa)
```



具体某部分语法：

```
radial-gradient() 径向渐变(放射性的效果) /*  
*  
    默认情况下径向渐变的形状根据元素的形状来计算的  
    正方形 --> 圆形  
    长方形 --> 椭圆形  
    - 我们也可以手动指定径向渐变的大小  
        circle  
        ellipse  
  
    - 也可以指定渐变的位置  
    - 语法:  
        radial-gradient(大小 at 位置, 颜色 位置 ,颜色 位置 [颜色 位置])  
        大小:  
            circle 圆形  
            ellipse 椭圆  
            closest-side 近边  
            closest-corner 近角  
            farthest-side 远边  
            farthest-corner 远角  
  
        位置:  
            top right left center bottom
```

# 表格

通过

table标签中使用tr标签代表行， tr标签中使用td标签代表列

```
<table border="1" width='50%' align="center">
    <!-- 在table中使用tr表示表格中的一行，有几个tr就有几行 -->
    <tr>
        <!-- 在tr中使用td表示一个单元格，有几个td就有几个单元格 -->
        <td>A1</td>
        <td>B1</td>
        <td>C1</td>
        <td>D1</td>
    </tr>
    <tr>
        <td>A2</td>
        <td>B2</td>
        <td>C2</td>
        <td>D2</td>
    </tr>
</table>
```

td标签中的colspan属性：横向合并单元格

```
<tr>
    <td>A4</td>
    <td>B4</td>
    <!--
        colspan 横向的合并单元格
    -->
    <td colspan="2">C4</td>
</tr>
```

td标签中的rowspan属性：纵向合并单元格

```
<tr>
    <td>A2</td>
    <td>B2</td>
    <td>C2</td>
    <!-- rowspan 纵向的合并单元格 -->
    <td rowspan="2">D2</td>
</tr>
<tr>
    <td>A3</td>
    <td>B3</td>
    <td>C3</td>
</tr>
```

A1	B1	C1	D1
A2	B2	C2	
A3	B3	C3	D2
A4	B4	C4	

## 长表格

table里面还可以有三个有语义的标签

**thead标签**: 语义是表格的头部

**tbody标签**: 语义是表格的主体

**tfoot标签**: 语义是表格的尾部

这三个可以在表格的第一行、中间所有、最后一行的**tr标签外面加上**

加上后，无论thead、tbody、tfoot的先后顺序怎么样，三个部分的位置永远都是头、主体、尾部，**位置定死了**。

**thead中的tr标签中的td标签可以用th来代替**，代替后头部的文本有**加粗居中**的效果

```

<body>
  <table border="1" width='50%' align="center">
    <!--
      可以将一个表格分成三个部分：
      头部 thead
      主体 tbody
      底部 tfoot
    -->
    <thead>
      <tr>
        <th>日期</th>
        <th>收入</th>
        <th>支出</th>
        <th>合计</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>2000.1.1</td>
        <td>500</td>
        <td>200</td>
        <td>300</td>
      </tr>
      <tr>
        <td>2000.1.1</td>
        <td>500</td>
        <td>200</td>
        <td>300</td>
      </tr>
    </tbody>
  </table>

```

日期	收入	支出	合计
2000.1.1	500	200	300
2000.1.1	500	200	300
2000.1.1	500	200	300
2000.1.1	500	200	300
		合计	300

## 表格的样式

table也是块元素，只不过比较特殊，他默认不是占全屏，而是被内容撑开

## 注意：

①

在内联样式中的table中设置border属性，设置边框宽度时，**内部外部全部都设置了**  
但是在**样式表**里面设置时，在table中设置只会为**表格外层添加边框**

②

如果表格中没有使用 tbody而是直接使用tr  
么**浏览器会自动创建一个 tbody**，并且将tr全都放到 tbody中  
tr不是 table的子元素，**table中通过子元素选择器也选不中tr**

以下是在**样式表**中设置表格属性：

可以按块元素的样式来设置表格的样式

**table**的border设置表格的**外围边框**

**td**的border设置**内部的边框**

**table**中的属性：

**border-spacing**属性：指定边框之间的距离

```
border-spacing: 0px;
```

**border-collapse:collapse** 设置边框的合并

border-collapse:collapse设置之后**border-spacing**属性就没有作用了

```
border-collapse: collapse;
```

**tr**中的属性：

让奇数行变颜色：使用**伪类选择器**让奇数行变颜色

**tr:nth-child(2n+1/even/odd)**

```
tr:nth-child(2n+1){  
    background-color: #bfa;  
}
```

学号	姓名	性别	年龄	地址
1	孙悟空	男	18	花果山
2	猪八戒	男	28	高老庄
3	沙和尚	男	38	流沙河
4	唐僧	女	16	女儿国

td中的属性：

默认情况下元素在td中是垂直居中的

在td中可以设置vertical-align (文本的垂直对齐属性)、和text-align (文本的水平对齐属性) 等属性来设置文本水平和垂直的对齐方式

补充：

在div中可以将元素的元素显示的类型display设置成table-cell，将div设置成一个单元格，这样能够使用vertical-align:middle (这个之前只能在文本和表格中使用) 来设置子元素的垂直对齐方式

元素变成table-cell后，比较特殊，他只能跟同是table-cell的元素在一行，其他元素不能跟其同行

```
.box1{
    width: 300px;
    height: 300px;
    background-color: orange;

    /* 将元素设置为单元格 td */
    display: table-cell;
    vertical-align: middle;
}
```

```
.box2{
    width: 100px;
    height: 100px;
    background-color: yellow;
    margin: 0 auto;
}
```

# 表单

网页中的表单用于将本地的数据提交给远程的服务器

form标签，form来创建一个表单

action属性：表单要提交的服务器的地址

```
<form action="target.html">  
/</form>
```

form中有以下几种表单项：

都是input标签（行内元素），只是type属性值不同

但是，这几个表单中的数据要提交到服务器，必须要为元素指定一个name属性

文本框：

type属性为text，value属性可以为文本框指定默认值

```
<input type="text" name="username" value="hello">
```



提交按钮：

type的值为submit

value属性可以指定提交按钮内的值

```
<input type="submit" value="注册">
```

密码框：

type属性值为password

```
<input type="password" name="password">
```

单选按钮框：

type属性值为radio

```
<input type="radio" name="hello" value="a">  
<input type="radio" name="hello" value="b">
```

## 单选按钮

几个按钮想要共属于一组，必须要有一个相同的name属性，这几个一组的按钮之间只能选中一个，否则相互之间没有关系

像这种选择按钮，还必须指定value属性，作为按钮的数据值传递给服务器

按钮框还可以为一个按钮设置默认选中属性，在input中加上checked即可，相应按钮一开始就是被选中的状态

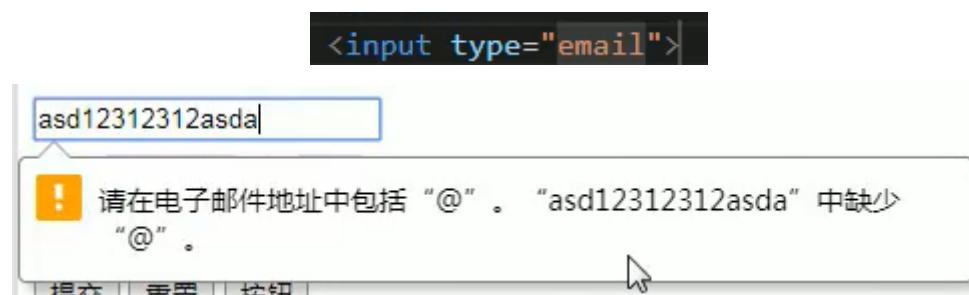
```
<input type="radio" name="hello" value="a">
<input type="radio" name="hello" value="b" checked>
```

## 颜色选择框

type属性值为color，可以选择颜色，兼容性不太好，IE不支持

## 邮件输入框

type值为email，和文本输入框相似，只不过是用来输入邮件的，输入的格式不符合邮件，会有提示，但提示的内容在不同的浏览器中有时候不一样



type还有一个tel值，是用来输入电话号码的，但是email和tel在pc端使用的不多，多用在移动端，移动端会有相应的不同键盘来配合使用

## 多选按钮框

type属性值是checkbox

```
多选框 <input type="checkbox" name="test" value="1">
<input type="checkbox" name="test" value="2">
<input type="checkbox" name="test" value="3">
```

## 多选框

可以多选按钮，其他性质和单选框一样，也可以设置checked属性

## 重置按钮

type属性的值为**reset**, 可以将表单里面写的内容重置, 要么清空, 要么变成默认的

```
<input type="reset">
```

## 普通按钮

type属性的值为**button**, value属性指定按钮的内容

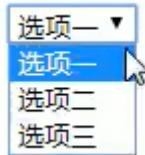
```
<input type="button" value="按钮">
```

## 下拉列表

下拉列表就不是input标签了, 为**select标签**

select标签中还有**option子标签**, 用来表示下拉选项

```
<!-- 下拉列表 -->
<select name="haha">
    <option value="i">选项一</option>
    <option value="ii">选项二</option>
    <option value="iii">选项三</option>
</select>
```



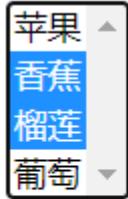
可以为某一个option标签设置**selected属性**, 表示默认选中

```
<select name="haha">
    <option value="i">选项一</option>
    <option selected value="ii">选项二</option>
    <option value="iii">选项三</option>
</select>
```

还有一个属性: **multiple属性**

用来表示下拉列表可以多选, 按住**ctrl键**进行多选

```
<select name="abc" v-model="fruits" multiple>
    <option value="苹果">苹果</option>
    <option value="香蕉">香蕉</option>
    <option value="榴莲">榴莲</option>
    <option value="葡萄">葡萄</option>
</select>
```



## 您选择的水果是[ "香蕉", "榴莲" ]

**数字框** input的type="number"

```
<input type="number" /> <!-- 数字框 -->
```

只能输入数字，可以是小数，右边的小箭头是用来加减整数的

与几个input按钮（提交按钮，重置按钮，普通按钮）相对的也有三个应有**button标签**

button标签也有**type属性**，可选值也分别由submit（提交）、reset（重置）、button（普通按钮）

作用跟上面那三个按钮一样，只是写法不一样，**button不是自结束标签，所以可以写一些复杂的结构**

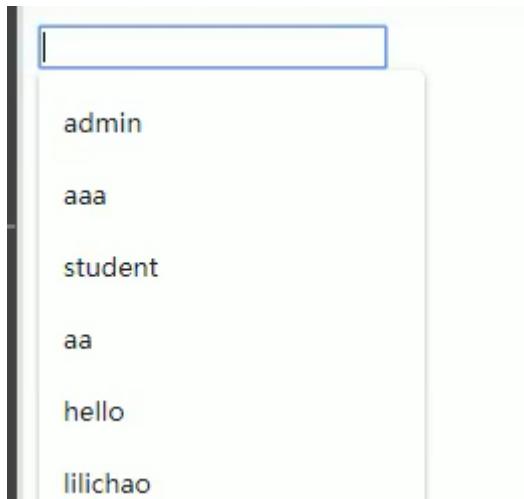
```
<button type="submit">提交</button>
<button type="reset">重置</button>
<button type="button">按钮</button>
```

某些属性的补充

**autocomplete属性**：用来控制输入框表单自动补全的功能，之前输入过的内容，可以选择展示，也可以关闭

on默认开启，off关闭

```
<input type="text" name="username" autocomplete="on">
```



如果将autocomplete属性放在input的样式中，也要一个一个设置，放在form标签的属性中，所有输入框的自动补全提示就全都关闭了

**readonly属性**: 将表单项设置为只读，不可修改，但数据会提交到服务器中

**disable属性**: 将表单项设置为禁用，也用不了，表单项直接变成了灰色，数据也不能提交了

**autofocus属性**: 自动获取焦点，页面一加载，焦点就在这个表单项中

## 网站的图标

设置**网站的图标**（在标题栏和收藏栏） 网站的图标一般都存在网站的根目录下面，名字一般都叫做 favicon.ico

**在head标签中，用link标签引入，rel属性值为icon**

```
<!-- 设置网站的图标（在标题栏和收藏栏）
-->
<link rel="icon" href="./favicon.ico">
```

## 项目代码的压缩

平时写完项目时，为了项目的上线用户体验流畅，应当将项目代码尽可能的压缩，将换行、空格和注释等等全都删去

使用**插件或者工具**，来将代码进行压缩，自动删除注释、换行、空格等等

# animation

## 过渡

**transition属性:** 过渡效果，哪个元素需要添加过渡效果就在那个元素样式中添加

两个可选值：

第一个：需要添加过渡效果的属性的名称。或者，all：所有的属性都添加过渡效果

第二个：过渡效果的时间，需要加上时间单位

```
.box2{  
    background-color: #bfa;  
    transition:height 2s;  
}
```

```
.box2{  
    background-color: #bfa;  
    transition:all 2s;  
}
```

### transition-property:

指定要执行过渡的属性

多个属性间使用逗号隔开

如果所有属性都需要过渡，则使用all关键字

大部分属性都支持过渡效果，只要是可计算的值基本都可以，颜色、角度等等都可以过渡

注意过渡时必须是从一个有效数值向另外一个有效数值进行过渡，不能有auto

### transition-duration:

指定过渡的持续时间

多个属性也可以分别指定时间，如下图：长宽变化时间不一样

```
transition-property: height , width;  
  
/*  
transition-duration: 指定过渡效果的持续  
时间单位: s 和 ms 1s = 1000ms  
*/  
transition-duration: 100ms, 2s;
```

### transition-timing-function:

指定过渡方式

可选值：

ease: 默认值，慢速开始，先加速，再减速

linear: 匀速运动

ease-in: 加速运动

ease-out: 减速运动

ease-in-out: 先加速后减速（和ease有一点点差别，速度上看加减速比ease猛）

cubic-bezier() 来指定时序函数，参数是四个值，上面所有的可选值其实都是由Bezier曲线生成的（一般不用这个）

生成bezier曲线的网站：<https://cubic-bezier.com>

steps() 分步执行过渡效果

两个可选值：

第一个为步数，分几步执行，将指定的过度时间除以这个步数，然后分段执行（效果是突变的）

第二个可选可不选：

end: 在时间结束时执行过渡（默认值）

start: 在时间开始时执行过渡

```
/*
 * transition-timing-function: cubic-bezier(.24,.95,.82,-0.88); */
/* transition-timing-function: steps(2, start); */
```

### transition-delay:

过渡效果的延迟，等待一段时间后在执行

### transition简写属性

可以同时设置过渡相关的所有属性，只有一个要求，如果要写延迟，则两个时间中第一个是持续时间，第二个是延迟

```
transition:2s margin-left 1s;
transition:2s margin-left 1s cubic-bezier(.24,.95,.82,-0.88);
```

## 动画

动画和过渡类似，都是可以实现一些动态的效果

过渡需要在某个属性发生变化时才会触发

动画可以自动触发动态效果

设置动画效果，必须先要设置一个关键帧，关键帧设置了动画执行每一个步骤

## 关键帧

**关键帧:** (写在css中)

语法:

```
@keyframes 关键帧自定义的名字{  
from{初始属性值};  
.....  
to{结束属性值};  
}
```

from是动画开始的位置，同时动画结束的位置，**from可以写成0%，to可以写成100%**，同样  
0~100%之间可以设置任意关键帧

```
@keyframes test {  
    /* [from表示动画的开始位置 也可以使用 0% */  
    from{  
        margin-left: 0;  
    }  
  
    /* to动画的结束位置 也可以使用 100%*/  
    to{  
        margin-left: 700px;  
    }  
}
```

设置好关键帧后，可以在元素的样式中指定动画属性，元素的样式位置和关键帧的位置**没有先后顺序要求**

动画有以下几个属性：

**animation-name:** 要对当前元素生效的关键帧的名字

**animation-duration:** 动画的持续时间

**animation-delay:** 动画的延时执行

**animation-timing-function:** 指定过渡方式，和过渡属性设置的一样

**animation-iteration-count:** 动画执行的次数，每次都从from到to

可选值：

次数：1/2、...

infinite：无限执行

**animation-direction:** 指定动画运行的方向

可选值：

normal: 默认值，从from向to运行每次都是这样

reverse: 从to向from运行每次都是这样

alternate: 从from向to运行，**重复执行动画时反向执行**，来回正放到放

alternate-reverse: 从to向from运行，重复执行动画时反向执行

**animation-play-state**: 设置动画的执行状态，就是用来控制动画的运行与暂停

可选值:

running: 默认值，动画执行

paused: 动画暂停

**animation-fill-mode**: 动画的填充模式

可选值:

none: 默认值，动画执行完毕元素回到原来位置

forwards: 动画执行完毕元素会停止在动画结束的位置

backwards: 动画延时等待时，元素就会处于开始位置，(**元素原来的样式可能跟from设定的样式不一样，这里是回到from设定的状态，none是元素原来的样式**)

both: 结合了 forwards 和 backwards

**animation**: 简写属性

以上所有的属性都可以写到简写属性中去，**持续时间放在延迟时间前面**

```
animation: test 2s 2 1s alternate;
```

**关键帧的补充：**

当设置多个关键帧时，可以在**关键帧中**修改最一开始在元素样式设置的过渡方式，让几个过程有不同的过渡方式

注意：在关键帧设置的过渡方式是**从这个位置到下一个关键帧的位置之间的过渡方式**，而不是过渡到此位置的过渡方式

例子：

```
.outer{
    height: 500px;
    border-bottom: 10px solid black;
    margin: 50px auto;
    overflow: hidden;
}
.box1{
    width: 100px;
    height: 100px;
    border-radius: 50%;
    background-color: #bfa;

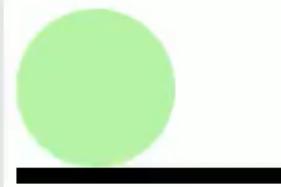
    animation: ball 1s forwards ease-in;
}

/* 创建小球下落的动画 */
@keyframes ball {
    from{
        margin-top: 0;
    }

    33%{
        margin-top: 400px;
        animation-timing-function: ease-in;
    }

    66%{
        margin-top: 100px;
    }

    to{
        margin-top: 400px;
        animation-timing-function: ease-in;
    }
}
```



还可以将几个关键帧整合，将具有相同属性的关键帧进行合并，类似于选择器分组

```
@keyframes ball {
    from{
        margin-top: 0;
    }

    33%, to{
        margin-top: 400px;
        animation-timing-function: ease-in;
    }

    66%{
        margin-top: 100px;
    }
}
```

## 变形

变形就是指通过CSS来改变元素的**形状或位置**

变形**不会影响到页面的布局**

**transform属性**, 用来设置元素的变形效果, 可以同时设置多个变形效果, 之间用**空格隔开**。

**一个元素只能有一个transform属性**, 一次性将所有需要变形的属性写完, 不能写多个, 多个会产生覆盖, 上一个写的所有属性完全不起作用

**此属性是可以作为transition属性的一个值, 作为有变化的属性**

有以下几种变形效果

### 平移

translateX(): 沿着x方向平移

translateY(): 沿着y方向平移

translateZ(): 沿着z轴方向平移

括号里面写平移的位置或者百分比, **百分比是相对于自身计算的**

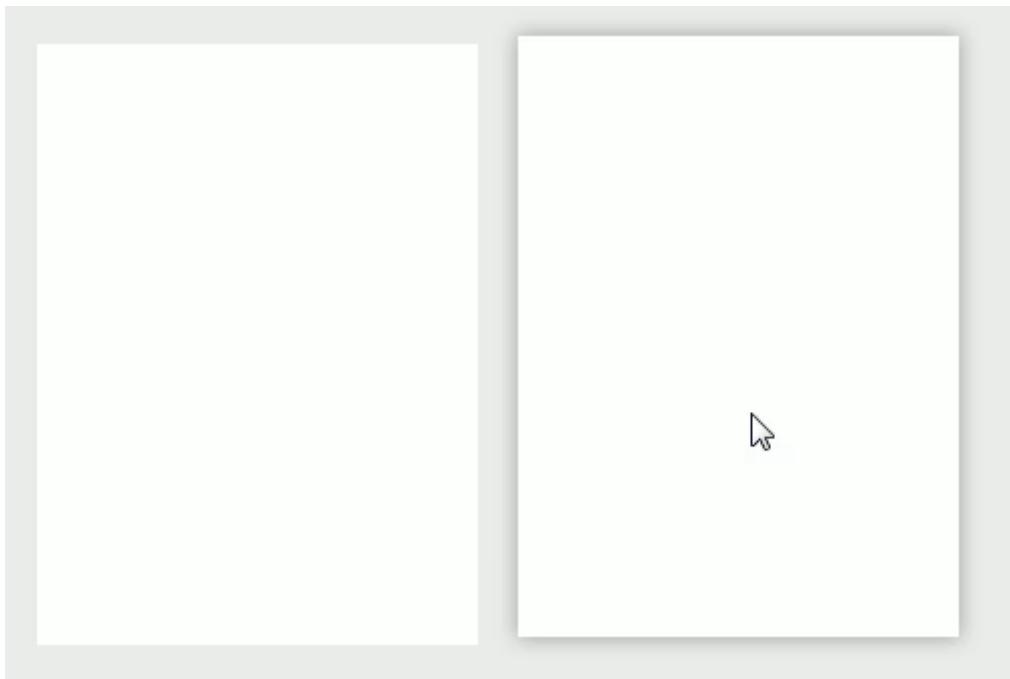
可以利用此特性来将**宽度高度不确定的元素**在页面实现**居中**, 如图:

```
.box3{
    background-color: orange;
    position: absolute;
    /*
     * 这种居中方式, 只适用于元素的大小确定
     */
    top: 0;
    left: 0;
    bottom: 0;
    right: 0;
    margin: auto; */

    left: 50%;  
    transform: translateX(-50%);  
}
```

box3是个div, 由于元素被内容撑开, 长度不确定, 页面居中的话可以使用**left相对包含块的居中**, 在用**transform**进行左侧移动元素自身的50%

此属性可以完成页面中的点击浮动的效果



## z轴平移

轴平移，调整元素在z轴的位置，正常情况就是调整元素和人眼之间的距离

轴平移属于立体效果（近大远小），**默认情况下网页是不支持透视**，如果需要看见效果，必须要设置**网页的视距**

网页的透视一般都设置给html标签，用**perspective属性**来设置

```
html{  
    /* 设置当前网页的视距为 800px，人眼距离网页的距离 */  
    perspective: 800px;  
}
```

设置之后就能开启了透视

```
body:hover .box1{  
    transform: translateZ(100px);  
}
```

设置了之后，translateZ()参数值为正，视觉上元素就会放大，为负，元素就会缩小，当元素旋转后，就能看到z轴的变化

## 旋转

通过旋转可以使元素沿着x、y或z旋转指定的角度，同样用**transform属性**来设置，有以下几个可选函数：

rotateX()  
rotateY()  
rotateZ()

参数是45deg、90deg等角度，或者圈数：0.25turn、1turn

开启透视也会有近大远小的效果

```
/* transform: rotateZ(.25turn); */
transform: rotateY(45deg);
```

transform属性里面表示变形的函数可以写多个，但是会有变形的先后顺序（**这个指的不是时间上的，而是效果上的**）

如图：

```
body:hover .box1{
    /* z轴先平移，然后y轴旋转，元素视觉上离我进了 */
    transform:translateZ(300px) rotateY(180deg);
    /* y轴先旋转，然后z轴平移，元素视觉上离我远了 */
    transform:rotateY(180deg) translateZ(300px);
}
```

旋转后，整个立体坐标轴也发生了旋转，**坐标轴的旋转会影响z轴上的形变**

与旋转有关的一个**backface-visibility属性**

可选值：

visible：**默认值**，元素的背面显示，相当于镜像反过来

hidden：元素的背面隐藏

## 缩放

对元素进行的缩放

transform属性中可选的函数：

scaleX()水平方向缩放

scaleY()垂直方向缩放

scale()双方向的缩放

函数里面的参数都是放大的倍数，大于1是放大，小于1是缩小

```
transform:scale(2)
```

**transform-origin属性**：用来设置变形的原点

可选值：

center：默认值，中心

相对于元素左上角的定位，例如20px 20px，中间用空格隔开

```
/* 变形的原点 默认值 center */
/* transform-origin:20px 20px; | */
```

## less

less是一门css的预处理语言

less是一个css的增强版，通过less可以编写更少的代码实现更强大的样式

在原生的css中也可以用一些less中常用到的语法，如下

比如设置变量

在HTML标签中设置**变量**：

语法：--变量名:变量值

调用：**var(--变量名)**

```
html{
    /* css原生也支持变量的设置 */
    --color: #ff0;
    --length:200px;
}

.box1{
    /* calc()计算函数 */
    width: calc(200px*2);
    height: var(--length);
    background-color: var(--color);
}
```

比如一些函数，如**计算函数calc()函数**，函数值可以是一个计算式

```
/* calc()计算函数 */
width: calc(200px*2);
```

但是在原生的css中写这些功能兼容性不好，于是有一门新的语言就出现了，不只是这些功能可以兼容，语法也更简化了，增添了许多对css的扩展

**浏览器不能直接执行less代码，要转化为css再由浏览器支持，转化的过程一般由插件来完成**

## 注释

//：单行注释，这种注释内容不会被解析到css中去

/\*\*/：css中的注释，注释的内容会被解析到css中去

## 变量

在变量中可以存储一个任意的值

并且我们可以在需要时，任意的修改变量中的值

变量的语法：@变量名:变量值

变量的引用：

①如果是直接使用，直接使用@变量名就行

```
@a: 200px;
@b: #bfa;

.box5{
    width: @a;
    color:@b;
}
```

②如果作为类名，或者一部分值时必须以@{变量名}的形式使用

```
.{@c}{
    width: @a;
    background-image: url("@{c}/1.png");
}
```

变量的声明和使用没有先后顺序，但一般都先声明，后声明的话也能用，同一个变量，后面的值会覆盖之前定义的

## \$的使用

如果想引用别的属性里面的值，使用\$属性名，就可以作为变量引用别的属性的属性值

```
div{
    color: red;
    background-color: $color;
}
```

## 父元素

后代元素直接在父元素的括号里面新建新的选择器（.class名、标签名）就行

如果想指定子元素，则应当在选择器之前加上>

```
LESS > SyntaxLESS > .box1
.box1{
  .box2{
    color: red;
  }

  >.box3{
    color: red;
  }
}
```

### &号的使用

less的选择器可以嵌套写表示祖先后代关系，用&可以代表当前这个选择器外面那层的父元素，用&替换此父元素，且不会产生其他的嵌套

例子：

```
LESS > SyntaxLESS > .box1 > div &
.box1{
  .box2{
    color: red;
  }

  >.box3{
    color: red;
    &:hover{
      color: blue;
    }
  }

  //为box1设置一个hover
  //& 就表示外层的父元素
  &:hover{
    color: orange;
  }

  div &{
    width: 100px;
  }
}
```

第一个&代表的是.box1>.box3:hover

第二个&代表的是box1:hover

第三个&代表的是div box1

## 样式的扩展

①

**extend**

**:extend()**

**对括号里面的选择器的属性进行扩展**，既有括号里面的选择器的属性，也可以新增属性，新增的属性写在新建的这个选择器中

括号里面的选择器可以是子元素选择器、分组选择器等等等

**语法：**

选择器2:extend(选择器1){ }

```
.p2:extend(.p1){  
|   color: red;  
|}  
|}
```

生成的css文件里面的样式，生成的**选择器分组**：

```
.p1,  
.p2 {  
|   width: 100px;  
|   height: 200px;  
|}  
.p2 {  
|   color: red;  
|}
```

**多处使用同一个选择器作为括号里的参数，相关的样式都会写到一个选择器分组里面，，提高效率，节省空间**

②

**混合mixins**

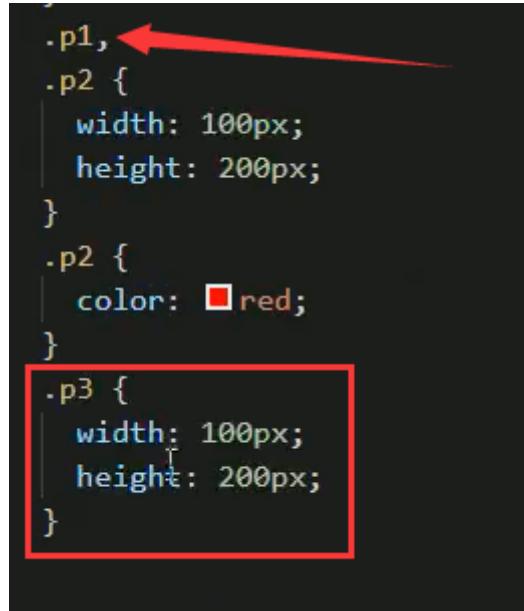
另一种方法直接将样式进行引用，相当于将样式进行了复制

这种方法叫样式的**mixins混合**

语法：在新的选择器2中添加**选择器1();**就可以

```
.p3{  
    //直接对指定的样式进行引用，这里就相当于将p1的样式在这里进行了复制  
    //mixin 混合  
    .p1();|  
}
```

生成的css样式：



```
.p1,  
.p2 {  
    width: 100px;  
    height: 200px;  
}  
.p2 {  
    color: red;  
}  
.p3 {  
    width: 100px;  
    height: 200px;  
}
```

样式是直接复制过来的，不会生成选择器分组

所以这么用效率会比较低

一般应用于以下的场景：

使用类选择器时可以在选择器后边添加一个括号，这时我们实际上就创建了一个 mixins混合函数，函数中可以加入形参

但是这个mixins不会在css中生成，这个mixins是专门用来给别的选择器用的，谁需要，谁就将这个带括号的mixins写入属性中

例子

```
.p4(){  
    width: 100px;  
    height: 100px;  
    background-color: #bfa;  
}  
  
.p5{  
    .p4();  
}
```

引用时，如果没有形参，()括号可以不写

```
.p3(){
    color: red;
    background-color: $color;
}

.box{
    .p3;
}
```

**括号中可以加入形参**，在调用这个mixins函数的时候在指定一些属性的值，形参用**@变量**的形式写到括号里面

```
//混合函数 在混合函数中可以直接设置变量
.test(@w,@h){
    width: @w;
    height: 100px;
    border: 1px solid red;
}

div{
    .test(200px,300px);
}
```

```
//混合函数 在混合函数中可以直接设置变量
.test(@w,@h,@bg-color){
    width: @w;
    height: @h;
    border: 1px solid @bg-color;
}

div{
    //调用混合函数，按顺序传递参数
    // .test(200px,300px,#bfa);
    .test(@bg-color:red, @h:100px, @w:300px);
}
```

**定义形参变量的时候，还可以指定默认值**

在写实参的时候，没写的实参就用默认值

不写默认值，有几个形参就写几个实参

```
//混合函数 在混合函数中可以直接设置变量
.test(@w:100px,@h:200px,@bg-color:#red){
    width: @w;
    height: @h;
    border: 1px solid @bg-color;
}

div{
    //调用混合函数，按顺序传递参数
    // .test(200px,300px,#bfa);
    .test([300px]);
    // .test(@bg-color:red, @h:100px, @w:300px);
}
```

less还定义了一些**官方的函数库**，可以直接调用

## less的补充

less中的所有数值都可进行直接的运算，像属性里面的数值都可以进行直接的运算

```
.box1{
    // 在less中所有的数值都可以直接进行运算
    // + - * /
    width: 100px + 100px;
    height: 100px/2;
    background-color: #bfa;
}
```

less还可以**用@import来将其他的less文件来导入**，其他文件中的less样式，在当前的less文件中生效

```
//import用来将其他的less引入到当前的less
@import "syntax2.less";
```

这样方便对less文件做模块化的处理

讲不同种类的less写到不同文件中，如mixins、变量、页面布局、动画效果……然后进行整合

# 弹性盒

flex(弹性盒、伸缩盒)

是css中又一种布局手段，它主要用来代替浮动来完成页面的布局

flex使元素具有弹性，让元素可以跟随页面的大小的改变而改变

## 弹性容器

使用弹性盒，必须先将一个元素设置为弹性容器

我们通过 **display属性** 来设置弹性容器

**display:flex**: 设为**块级弹性容器**，独占一行

**display:inline-flex**: 设置为**行内的弹性容器**，不会独占一行

设置弹性盒后，子元素如果浮动后，就不会有高度塌陷的问题，盒子随盒内元素的浮动而变化高度

## 弹性元素

弹性容器的直接子元素就会自动变成弹性元素

一个元素可以同时是弹性容器和弹性元素

## 弹性容器的属性

### flex-direction

指定容器中弹性元素的排列方式

可选值

row: **默认值**，弹性元素在容器中水平排列（左向右）

row-reverse: 弹性元素在容器中反向水平排列（右向左）

column: 弹性元素纵向排列（自上向下）

column-reverse: 弹性元素方向纵向排列（自下向上）

主轴：

**弹性元素的排列方向**称为主轴

侧轴（辅轴）：

与主轴垂直方向的称为侧轴（辅轴）

## **flex-wrap属性**

设置弹性元素是否在弹性容器中自动换行

可选值

**nowrap**: 默认值，元素不会自动换行

**wrap**: 元素沿着辅轴方向自动换行

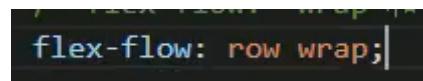


**wrap-reverse**: 元素沿着辅轴反方向换行



## **flex-flow属性**

上面两个属性**flex-direction**和**flex-wrap**的简写属性



相当于横向排列，并且换行

## **justify-content属性**

如何分配**主轴上的空白空间** (主轴上的元素如何排列)

可选值

**flex-start**: 元素沿着主轴起边排列

**flex-end**: 元素沿着主轴终边排列



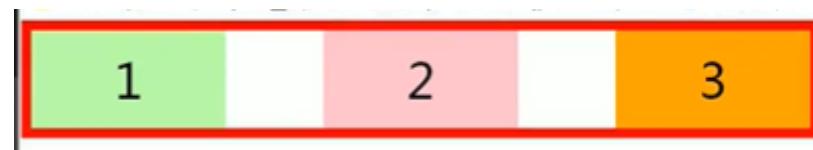
**center**: 元素居中排列



space-around: 空白分布到元素两侧



space-between: 空白均匀分布到元素间，最两遍不会分布



space-evenly: 空白分布到元素的单侧

**ps:** 有justify一般都是主轴上的属性，有align一般都是辅轴上面的属性

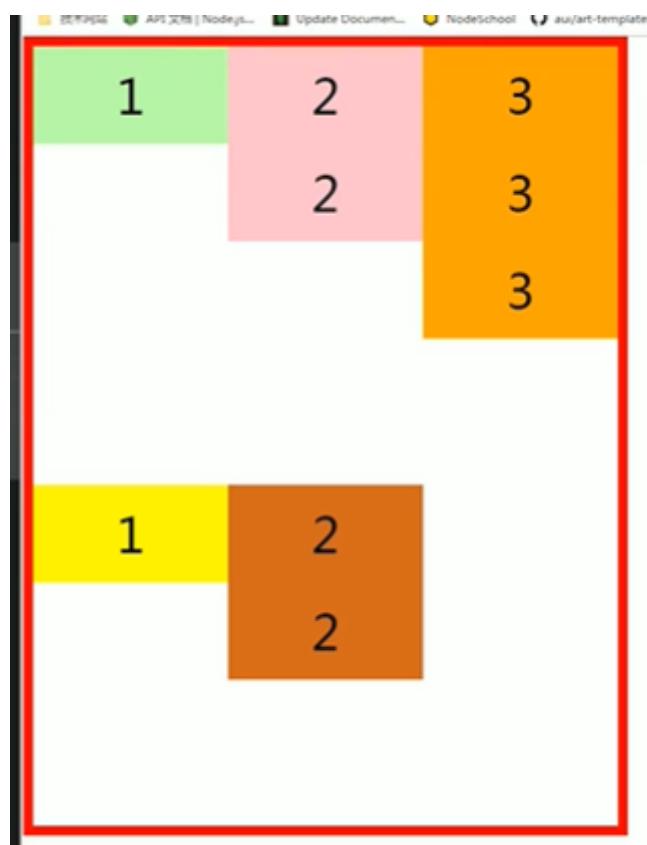
#### align-items属性：

##### 元素在辅轴上面如何对齐

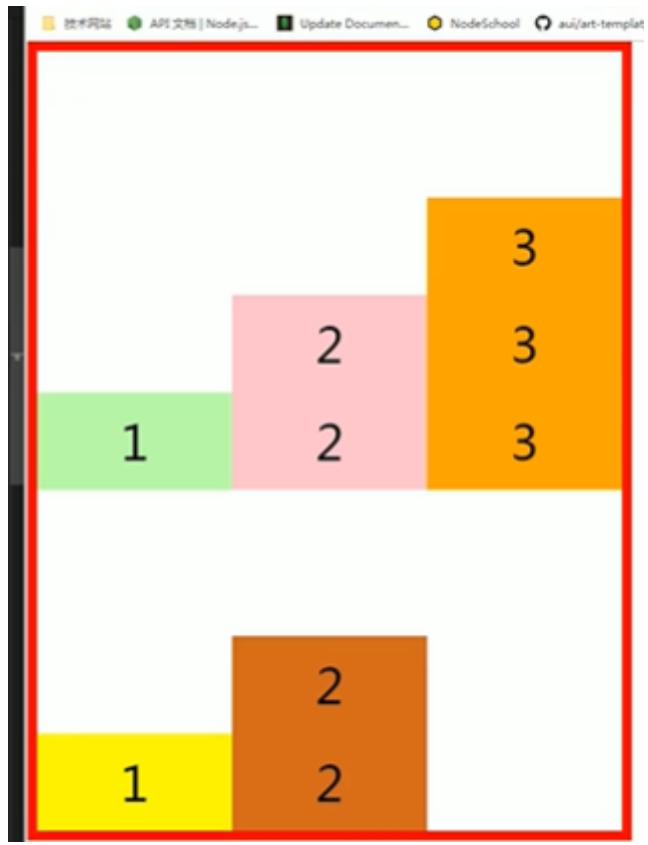
可选值：

stretch: 默认值，将元素的长度设置为相同的值

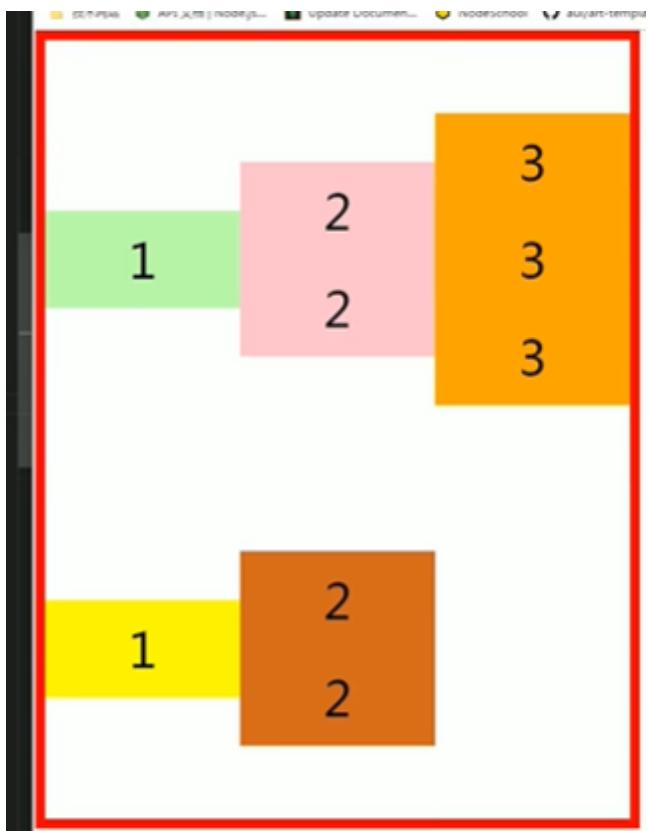
flex-start: 元素不会拉伸，沿着辅轴起边对齐



flex-end: 沿着辅轴的终边对齐



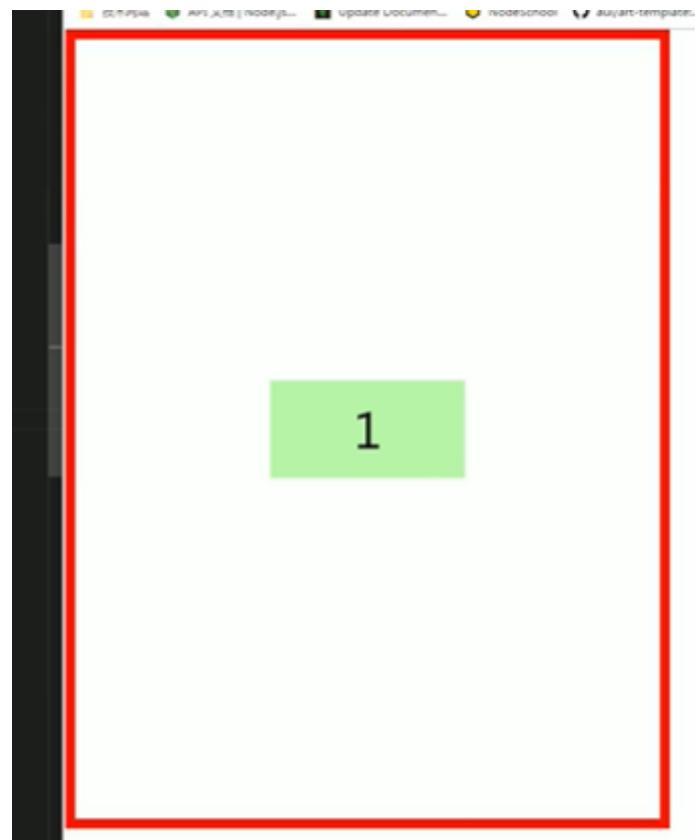
center: 居中对齐



baseline: 基线对齐

可以利用上面这两个属性：justify-content和align-items，来做到**垂直水平双居中**

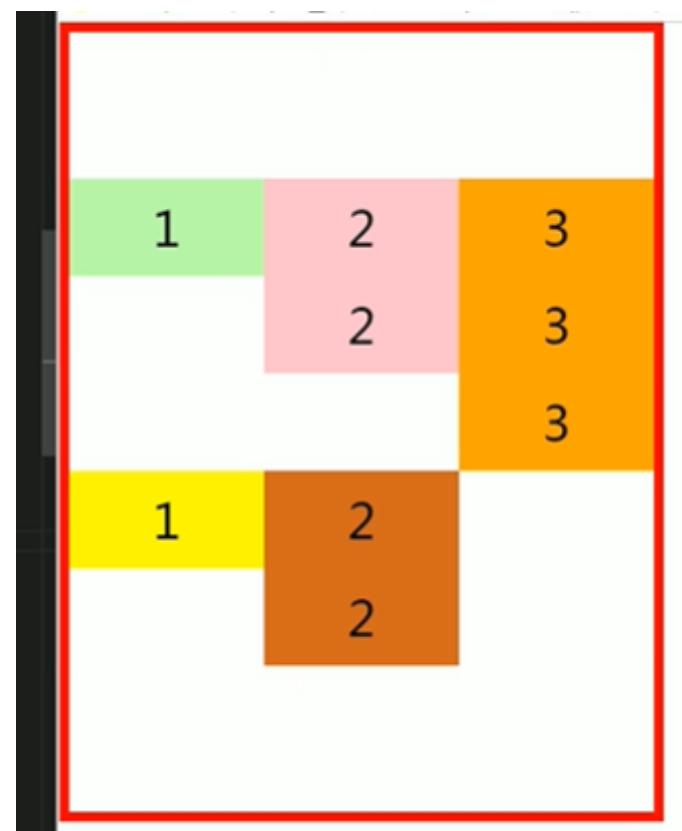
```
justify-content: center;  
align-items: center;
```



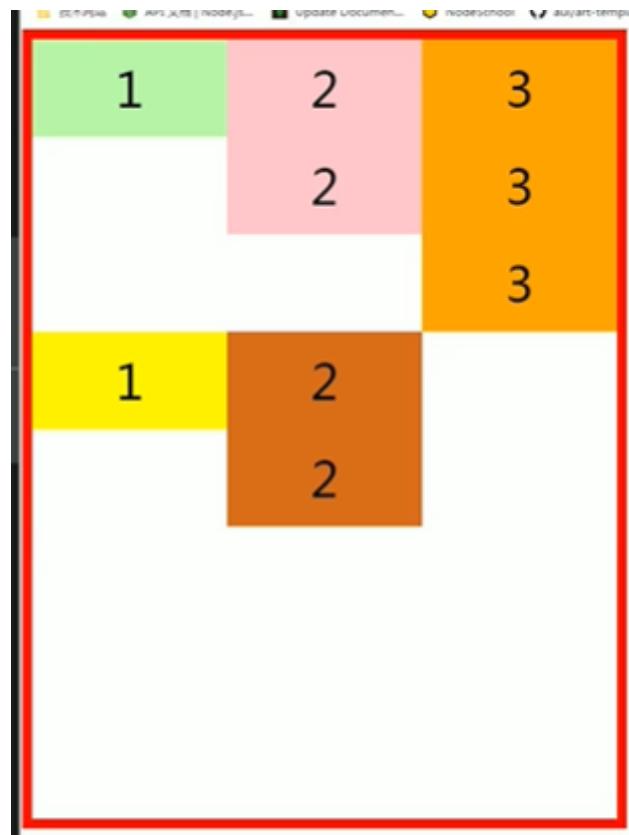
### align-content属性

辅轴空白空间的分布，跟justify-content的属性、用法一样

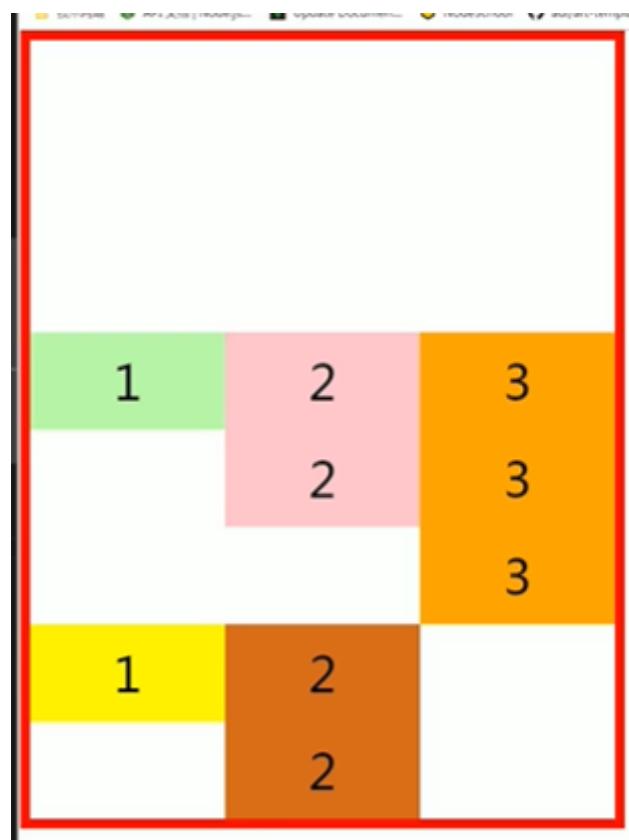
center: 元素居中排列



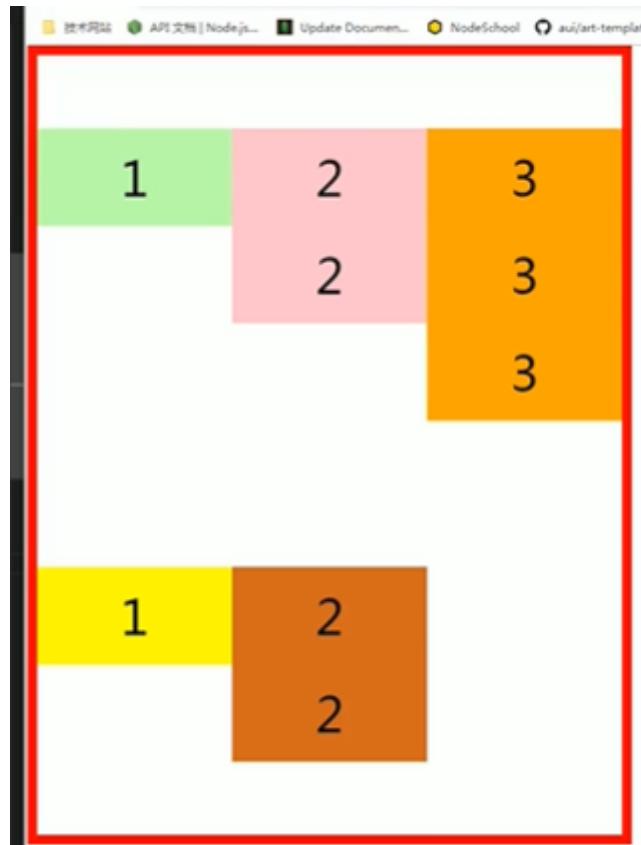
flex-start: 元素沿着辅轴起边排列



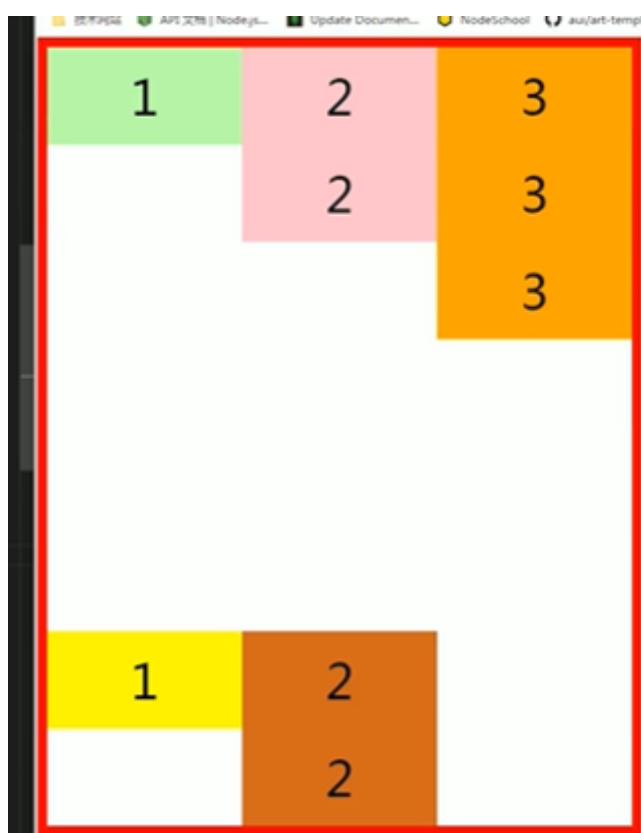
flex-end: 元素沿着辅轴**终边**排列



space-around: 空白分布到元素两侧



space-between: 空白均匀分布到元素间，最两遍不会分布



## 弹性元素的属性

### flex-grow属性 (弹簧拉长)

指定弹性元素的伸展系数

设定的是当父元素有多余空间的时，子元素如何伸展

父元素的剩余空间，会按照比例进行分配

语法：**flex-grow:比例数;**

**flex-grow默认为0**，表示弹性元素不会分配弹性盒的剩余空间

可以为每个元素分设置flex-grow属性，也可以统一设置

**统一设置为1**，表示父元素的剩余空间平均分配

如果每个子元素都设置了flex-grow，制定了不同的比例数，剩余空间会**按照比例分配**

例子：弹性容器中，弹性容器分别设置

```
li:nth-child(1){  
    flex-grow: 0;  
}  
  
li:nth-child(2){  
    background-color: #pink;  
    flex-grow: 2;  
}  
  
li:nth-child(3){  
    background-color: #orange;  
    flex-grow: 3;  
}
```



统一设置：

```
li{  
    width: 200px;  
    height: 100px;  
    background-color: #bfa;  
    font-size: 50px;  
    text-align: center;  
    line-height: 100px;  
  
    /*  
        弹性元素的属性：  
            flex-grow 指定弹性元素的伸展的系数  
            - 当父元素有多余空间时，子元素如何伸展  
    */  
    flex-grow: 1;  
}
```



## flex-shrink属性 (弹簧压缩)

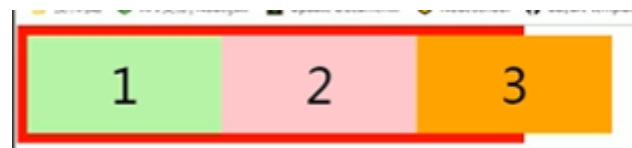
指定弹性元素的收缩系数

当父元素中的空间不足以容纳所有的子元素时，如何对子元素进行收缩

用法和flex-grow类似

比例数统一设置成1时，等比例收缩

当比例系数统一设置成0，就不会收缩



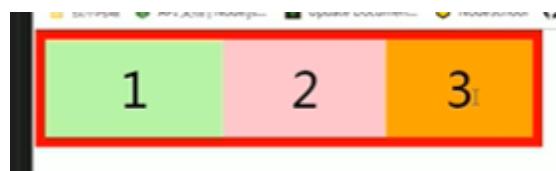
分别指定比例系数，系数大的收缩的多，弹性元素会比较小

```
}

li:nth-child(1){
    flex-grow: 0;
    flex-shrink: 1;
}

li:nth-child(2){
    background-color: pink;
    /* flex-grow: 2; */
    flex-shrink: 2;
}

li:nth-child(3){
    background-color: orange;
    /* flex-grow: 3; */
    flex-shrink: 3;
}
```



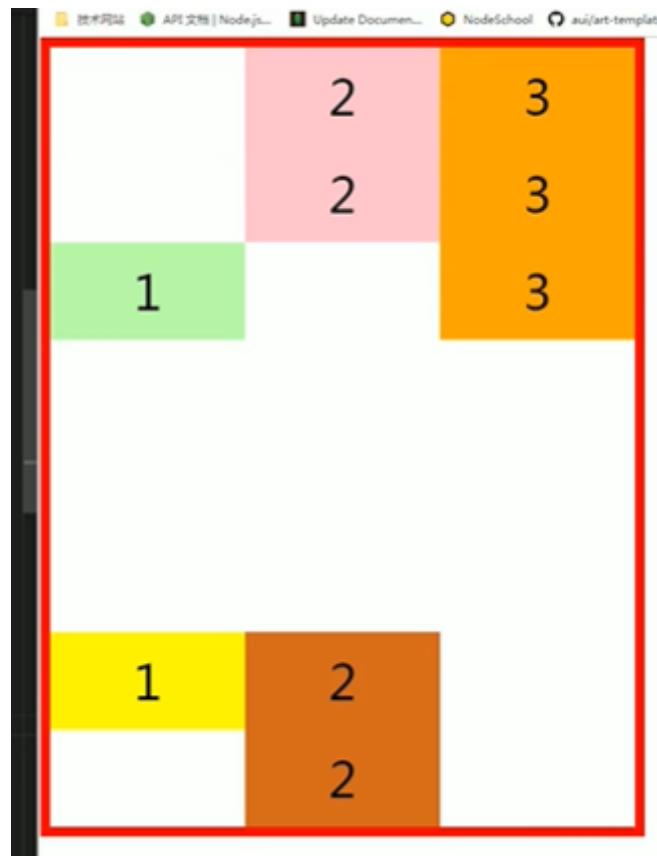
## align-self属性

用来覆盖当前弹性元素上的align-items属性

align-items之前是弹性容器统一设定的，这里单独为每一个弹性元素单独设置，会覆盖之前统一设置的

```
}

li:nth-child(1){
    /* align-self: 用来覆盖当前弹性元素上的align-items */
    align-self: flex-end;
}
```



### flex-basis属性 (弹簧原长)

flex-basis指定的是元素在**主轴**上的基础长度，**指定具体长度后，会覆盖之前指定的元素的大小**

如如果**主轴是横向**的则该值指定的就是元素的宽度

如果**主轴是纵向**的则该值指定的是就是元素的高度

**默认值是auto**，表明参考元素之前设定的自身高度或宽度

### flex属性

弹性元素属性的**简写样式**，可以同时设定弹性元素的**三个属性（拉伸、压缩、原长）**

可选值：**增长 缩减 基础**，三个值顺序不能变，中间用**空格隔开**

同时还提供三个参数值：

**initial**相当于flex: 0 1 auto

**auto**相当于flex: 1 1 auto

**none**相当于flex:0 0 auto"弹性元素没有弹性"

### order属性

通过为弹性元素设定order值，来**控制元素的排列顺序**

可选值就是数字

通过一个属性来改变元素的顺序布局

```
        li:nth-child(1){  
            /* order 决定弹性元素的排列顺序 */  
            order: 2;  
        }  
  
        li:nth-child(2){  
            background-color: #pink;  
            /* flex-grow: 2; */  
            order: 3;  
        }  
  
        li:nth-child(3){  
            background-color: #orange;  
            /* flex-grow: 3; */  
            order: 1;  
        }
```



像素

## 像素：

- 屏幕是由一个一个发光的小点构成，这一个个的小点就是像素
- 分辨率：**1920 × 1080** 说的是屏幕中小点的数量
- 在前端开发中像素要分成两种情况讨论：**CSS像素** 和 **物理像素**
- 物理像素，上述所说的小点点就属于物理像素
- **CSS像素**，编写网页时，我们所用像素都是**CSS像素**
  - 浏览器在显示网页时，需要将**CSS像素**转换为物理像素然后再呈现
  - 一个**css像素**最终由几个物理像素显示，由浏览器决定：  
默认情况下在**pc端**，一个**css像素** = 一个物理像素

## 视口 (**viewport**)

- 视口就是屏幕上用来显示网页的区域
- 可以通过查看视口的大小，来观察**CSS像素**和**物理像素**的比值
- 默认情况下：
  - 视口宽度 **1920px (CSS像素)**
  - 1920px (物理像素)**
  - 此时，**css像素**和**物理像素**的比是 **1:1**
- 放大两倍的情况：
  - 视口宽度 **960px (CSS像素)**
  - 1920px (物理像素)**
  - 此时，**css像素**和**物理像素**的比是 **1:2**
- 我们可以通过改变视口的大小，来改变**CSS像素**和**物理像素**的比值

## 手机端像素

在不通的屏幕，单位像素的大小是不同的，像素越小屏幕会越清晰  
**24寸 1920×1080**  
**16 4.7寸 750 × 1334**

智能手机的像素点远远小于计算机的像素点

问题：一个宽度为**900px**的网页在**iphone6**中要如何显示呢？

默认情况下，移动端的网页都会将视口设置为**980像素 (css像素)**  
以确保**pc端**网页可以在移动端正常访问，但是如果网页的宽度超过了**980**，  
移动端的浏览器会自动对网页缩放以完整显示网页

默认情况下，**移动端**的网页都会将视口设置为**980像素 (css像素)**

当网页的**css像素**超过了**980px**时，为确保移动端的网页能正常显示，**移动端的浏览器会自动将网页进行缩放**来完整显示网页（在网页没有专门进行移动端的适配开发时）

所以基本大部分的pc端网站都可以在移动端中正常浏览，但是往往都不会有一个好的体验

为了解决这个题，**大部分网站都会专门为移动端设计网页**

## 完美视口

移动端默认的视口大小是980px(css像素)，

默认情况下，移动端的像素比就是  $980/\text{移动端宽度}$  ( $980/750$ )

如果我们直接在网页中编写移动端代码，这样在980的视口下，像素比是非常不好，

导致网页中的内容非常非常的小

编写移动页面时，必须要确保有一个比较合理的像素比：

1css像素 对应 2个物理像素

1css像素 对应 3个物理像素

每一款移动设备设计时，都会有一个**最佳的像素比**

般我们只需要将像素比设置为该值即可得到一个最佳效果

将像素比设置为最佳像素比的视口大小我们称其为**完美视口**

可以通过**meta标签**来**设置视口大小**

建立一个meta标签，**name属性为viewport，content属性为width=像素值**，可以为网页指定，在网页占满手机全屏时，页面宽度为多少像素，相当于把网页放大，一个css像素由几个物理像素来呈现

当meta标签，**name属性为viewport，content属性为width=device-width**

**这样就是自动适配所有移动端最佳像素比，不用再指定了**

```
<!-- 设置视口大小 device-width表示设备的宽度（完美视口） -->
<meta name="viewport" content="width=device-width">
```

此外还有一个initial-scale属性，这是设置网页的初始化缩放，完美视口的初始化缩放设置为一

所以**一般完美视口使用以下的标签**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

## vw单位

不同的设备完美视口的大小是不一样的

iphone6 -- 375

iphone6plus -- 414

由于不同设备视口和像素比不同，所以同样的375个像素在不同的设备下意义是不一样

比如在 iphone6中375就是全屏，而到了plus中375就会缺一块

**所以在移动端开发时，就不能再使用px来进行布局了**

vw表示的是视口的宽度( viewport width)

100vw = 1个视口的宽度

1vw = 1%视口宽度

vw这个单位永远**相对于视口宽度**进行计算

**设计图的宽度**一般都是750px、1125px、**375的2倍3倍**....

**换算：**

使用vw作为单位，创建一个48px\*35px大小的元素

100vw=750px (设计图的像素)  $0.1333333333 \text{vw} = 1 \text{px}$

$6.4 \text{vw} = 48 \text{px}$  (设计图像素)

## vw的适配

可以用之前讲过的rem来做适配

**1rem=html中font-size的值**

font-size中设定值为vw单位，之后网页中的宽高用rem来作为单位

**但是大部分浏览器网页中现在规定字体大小最小为12px**

比如面那个例子，所以还不能用font-size:  $0.1333333333 \text{vw}$

此时font-size的值为1px，太小了

所以有一种常见的解决方案时将**font-size设置为5.33333vw=40px**

之后就就可以将设计图上的长度/40得到rem为单位的值

```
html{
    /*
        网页中字体大小最小是12px，不能设置一个比12像素还小的字体
        如果我们设置了一个小于12px的字体，则字体自动设置为12
        0.1333333vw = 1px
        5.33333vw = 40px
    */
    font-size: 5.33333vw;
}

.box1{
    /*
        rem
        - 1 rem = 1 html的字体大小
    */
    width: 4.8rem;
    height: 3.5rem;
    background-color: #bfa;
}
```

跟上面那个直接换算没有本质区别，只是方便了换算，可以乘除整数了

## 响应式布局

### 响应式布局

网页可以根据不通的设备或窗口大小呈现出不同的效果

使用响应式布局，可以使一个网页适用于所有设备

### 响应布局的关键就是媒体查询

通过媒体查询，可以为不同的设备，或设备不同状态来分别设置样式

## 媒体查询

使用媒体查询

**语法：**@media 查询类型{}

### 查询类型：

①

### 媒体类型：

all：所有设备

print：打印设备

screen：带屏幕的设备

speech：屏幕阅读器

可以使用逗号连接多个媒体类型

```
@media print,screen{  
    body{  
        background-color: #bfa;  
    }  
}
```

ps：也可以在媒体类型前面加上only，这是为了兼容老版本的浏览器

```
@media only screen {  
    body{  
        background-color: #bfa;  
    }  
}
```

②

### 媒体特性

**width**: 视口的宽度（视口等于指定宽度时生效）

**height**: 视口的高度（视口等于指定高度时生效）

**min-width**: 视口的最小宽度（视口大于等于指定宽度时生效）

**max-width**: 视口的最大宽度（视口小于等于指定宽度时生效）

上面说的生效的内容是后面大括号里面的符合完整css语法规则的样式

语法：

```
@media(媒体特性){ css样式}
```

```
@media (max-width: 500px){  
    body{  
        background-color: #bfa;  
    }  
}
```

样式切换的分界点，我们称其为**断点**，也就是网页的样式会在这个点时发生变化

### 一般比较常用的断点

小于768**超小屏幕** max-width=768px

大于768**小屏幕** min-width=768px

大于992**中型屏幕** min-width=992px

大于1260**大屏幕** min-width=1260px

可以多个媒体特性一起用

多个媒体特性中间用**逗号**隔开，表示或

```
@media (min-width: 500px), (max-width: 300px){  
    body{  
        background-color: #bfa;  
    }  
}
```

多个媒体特性中间用**and**相连，表示与

```
@media (min-width: 500px) and (max-width: 700px){  
    body{  
        background-color: #bfa;  
    }  
}
```

同时还可以指定媒体类型，也用and连接起来

```
@media only screen and (min-width: 500px) and (max-width: 700px){  
    body{  
        background-color: #bfa;  
    }  
}
```

## PS

### title属性

用于将某一段文字设置一个标签，鼠标停留在文字上就会出现文字提示

```
<p title="abc">少小离家老大回</p>
```



**快捷键：** ul>li，然后点击tab键，直接就生成了如下格式代码

```
<ul>
  <li></li>
</ul>
```

ul>li\*3，就会生成带有三行的一个列表

写完一段字符串，按tab键，就会生成名为这段字符串的一个标签，无论这个标签是否存在

**快捷键：** db按tab，会直接得到display:block

```
display: block;
```

**快捷键：**

.box就创建了

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

.box\$\*3就创建了

```
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
```

**快捷键：** 输入**.box1**,然后按tab键，就自动生成了如下结构

```
<div class="box1"></div>
```

**.box1+.box2+.box3**然后按tab键，就自动生成了三个div块

**快捷键：** shift+ctrl+R代表当前行向下复制一行

**快捷键：** **w190**，然后点击tab，就直接生成了width: 190px;

h190，然后点击tab，就直接生成了height: 190px;

## line-height 属性

设置行间的距离（行高）。

注意：不允许使用负值。

### 该属性可以被继承

多用于以下这种情况：

要让一个**文字在父元素中垂直居中**，只需要将父元素的line-height和height设置成相等就行

```
/* 设置菜单内部的item */
.left-nav .item{
    height: 25px;
    /* 要让一个文字在父元素中垂直居中，只需将父元素的line-height设置为一个和父元素height相等的值 */
    line-height: 25px;
}
```

去除超链接的下划线

**text-decoration属性**，将他设置为none就能去掉下划线

```
/* 去除下划线 */
text-decoration: none;
```

## font-weight:bold

给font-weight属性设置bold值，代表将文字加粗

## background-clip属性

```
/* 将背景颜色值设置到内容区，边框和内边距不在有背景颜色 */
background-clip: content-box;
```

用fontawesome字体图标，将图标放到li标签中

li原来是块元素，fontawesome中预设的css的样式就把li改成了行内元素，图标就会跟后面的在一  
行

在div中可以将元素的元素显示的类型**display设置成table-cell**, 蒋奇设置成一个单元格, 这样能够使用**vertical-align:middle (这个之前只能在文本和表格中使用)** 来设置子元素的垂直对齐方式

元素变成table-cell后, 比较特殊, 他**只能跟同是table-cell的元素在一行, 其他元素不能跟其同行**

```
.box1{  
    width: 300px;  
    height: 300px;  
    background-color: orange;  
  
    /* 将元素设置为单元格 td */  
    display: table-cell;  
    vertical-align: middle;  
}  
  
.box2{  
    width: 100px;  
    height: 100px;  
    background-color: yellow;  
    margin: 0 auto;  
}
```

**min-width属性**: 表示元素的最小宽度

一般用于body标签, 当浏览器边缘缩放时, **防止body过小导致子元素出现问题, 设置的最小宽度**, 再往小了缩小浏览器, body的大小就不跟着变化了

transition: 为元素设置过渡效果

编写网页时，为元素命名样式，如果是制作广告栏，有时候**class命名ad**，里面的图片有时候会被浏览器自动拦截，可能是当做广告了，**可能跟浏览器的某些插件有关**

---

设置**网站的图标**（在标题栏和收藏栏） 网站的图标一般都存在网站的根目录下面，名字一般都叫做 favicon.ico

**在head标签中，用link标签引入，rel属性值为icon**

```
<!-- 设置网站的图标（在标题栏和收藏栏）
| 网站的图标一般都存在网站的根目录下面，名字一般都叫做favicon.ico
| -->
<link rel="icon" href="./favicon.ico">
```

## 错误记录

报错parsing error: invalid version tag

和Failed to decode downloaded font

是因为没有吧webfonts导入到项目文件中！！！

