链滴

HTML 和 CSS 基础学习

作者：sean77

原文链接：https://ld246.com/article/1609679943018

来源网站：链滴

HTML提供了大量元素，没一个元素都有特殊的用途，保证网页的丰富多样性：

- 区块：div
- 区分：span
- 文本：p、h1~h6、em、dt、dd
- 表格：table、tbody、thead、tr、td、th、tfoot、caption
- 表单：form、input、label、textarea、select
- 链接：a
- 图片：img
- 文档：html、head、title、body、meta
- 列表：ul、ol、li、dlside、footer、nav
- 其他：br、hr、iframe
- 结构：header、section、a、strong、pre、adderss、q、blcokquote、cite、code

文档声明

<!DOCTYPE html>

HTML的基本元素

html元素

是HTML文档的根元素，一个文档中只有一个，其他元素都是它的后代元素

W3C标准建议为html元素增加lang属性

<html lang="en"></html>

作用是：

- 帮助语音合成工具确定要使用的发音
- 帮助翻译工具确定要使用的翻译规则

head元素

head元素里面的内容是一些元数据（描述数据的数据）

一般用于描述网页的各种信息，如字符编码、网页标题。网页图标

- title
- meta
- style
- link

h、p、strong、code、br、hr

字符实体

HTML中有一些字符是预留出来作特殊用途的，比如

- 小于号(<)
- 大于号（>）

想要在网页中正确地显示这些预留字符，必须使用字符实体，书写格式一般有两种

- &entity_name: (空格) &lt(小于号)
- &#entity_number： (空格) &#60（小于号）

span元素

默认情况下，跟普通文本几乎没差别

用于区分特殊文本和普通文本，比如用来显示一些关键字。

对普通的文本进行归类。

div元素

一般作为其他元素的容器，把其他元素包住，代表一个整体。

用于把网页分割为多个独立的部分。

img元素

img元素是单标签

img元素的属性：

- src：设置图片的路径，可以使用绝对路径或相对路径。
- alt
- width
- height：使用很少

a元素

定义超链接，用于打开新的URL

属性：

- href：链接地址。如果不写href属性，会被识别为普通文本。
- target：打开方式

- _self
    - _blank
    - _parent：与iframe配合使用
    - _top：与iframe一起使用

a元素和base元素可以结合使用：

如果页面中要访问的url的域名或ip相同，可以抽离到base中。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <base href="https://www.baidu.com" target="_blank">
</head>
<body>
    <a href="/">百度一下</a>
    <a href="/img/bd_logo1.png">百度logo</a>
</body>
</html>
```

锚点

```
<!-- 只有#号的锚点会跳转到页面顶部 -->
<a href="#">顶部</a>
<a href="#title1">标题1</a>
<a href="#title2">标题2</a>
<a href="#title3">标题3</a>

<h2 id="title1">我是内容1</h2>
<h2 id="title2">我是内容2</h2>
<h2 id="title3">我是内容3</h2>
```

伪链接

有时候点击链接的时候不希望打开新的URL，而是希望做点别的事情，这时候就可以使用伪链接。

伪链接：没有指明具体链接地址的链接

点击伪链接之后具体来做什么事情，要编写对应的js代码。

如果暂时不需要做任何事情，可以用下面的形式

```
<a href="#" onclick="return false;">伪链接1</a>
<a href="javascript:">伪链接2</a>
```

iframe

现在使用很少。

在网页中嵌套网页。

标签语义化的重要性

标签语义化指的是选择标签的时候尽量让每一个标签都有正确的语义。

虽然很多标签之间互换之后也能实现功能，但还要遵守标签语义化原则。

比如自己可以用一个div去加上相应样式模仿h1标签，但是不推荐这么做，标签语义化很重要。

列表

HTMl提供了3组常用的用来展示列表的元素：

- 有序列表 ol、li
- 无序列表 ul、li
- 定义列表： dl、dt、dd

有序列表

ol (ordered list) 有序列表，直接子元素只能是li(list item)

```
<ol>
  <li>海王</li>
  <li>海贼王</li>
  <li>上海堡垒</li>
  <li>星际穿越</li>
</ol>
```

对于列表，Chorme浏览器默认会加上如下样式元素。

```
/* user agent stylesheet */
ol {
  display: block;
  list-style-type: decimal;
  margin-block-start: 1em;
  margin-block-end: 1em;
  margin-inline-start: 0px;
  margin-inline-end: 0px;
  padding-inline-start: 40px;
}

/* user agent stylesheet */
li {
  display: list-item;
  text-align: -webkit-match-parent;
}
```

浏览器默认加上了边距的样式，加边距的时候，Chrome使用了margin-block-**，而不是使用margi
-left或其他的，是因为有些国家或地区在阅读时，可能是从右向左，这样设置更容易进行适配。

无序列表

ul (unordered list) 有序列表，直接子元素只能是li(list item)。

默认样式等，与有序列表基本相同。

定义列表

dl(definition list)定义列表，直接子元素只能是dt(definition term)、dd(definition description)。

dt：列表中每一项的项目名

dd：列表中每一项的具体描述，是对dt的描述、解释、补充。

- 一个 dt后面一般紧跟着一个或多个dd。

dt、dd常见的组合是：

- 事务的名称、事务的描述
- 问题、答案
- 类别名、归属于这类的各种事务

```
<dl>
  <dt>红饮料</dt>
  <dd>西瓜汁</dd>

  <dt>黑饮料</dt>
  <dd>咖啡</dd>

  <dt>白饮料</dt>
  <dd>牛奶</dd>
 </dl>
```

表格

table: 表格

tr：表格中的行

td：行中的单元格

table相关属性

table的常用属性

属性	说明
border	边框的宽度
cellpadding	单元格内部的间距
cellspacing	单元格之间的间距

width	表格的宽度
align	表格的水平对齐方式 left、center、right

```
<table border="1" cellspacing="10" width="500" align="center">
  <tr>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
  </tr>
  <tr>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
  </tr>
  <tr>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
  </tr>
  <tr>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
    <td>内容</td>
  </tr>
</table>
```

tr和td相关属性

tr常见属性

属性	说明
valign	单元格的垂直对齐方式 top、middle、bottom、baseline
align	单元格的水平对齐方式 left、center、right

th和td常见属性

属性	说明
valign	单元格的垂直对齐方式 top、middle、bottom、baseline

align	单元格的水平对齐方式 left、center、right
width	单元格的宽度
height	单元格的高度
rowspan	单元格可横跨的行数
colspan	单元格可横跨的列数

细线表格

使用cellspacing = 0边框会合在一起，但是边框宽度是border*2。

可以使用border-collapse将边框合并。

```
table {
  border: 1px solid #000;
  border-collapse: collapse;
}
```

表格的其他元素

thead 表格的表头

tbody 表格的主体 内容放在tbody中

tfoot 表格的页脚

caption 表格的标题

th 表格的表头单元格

单元格的合并

合并方向是向右、向下。

colspan 跨列合并

rowspan 跨行合并

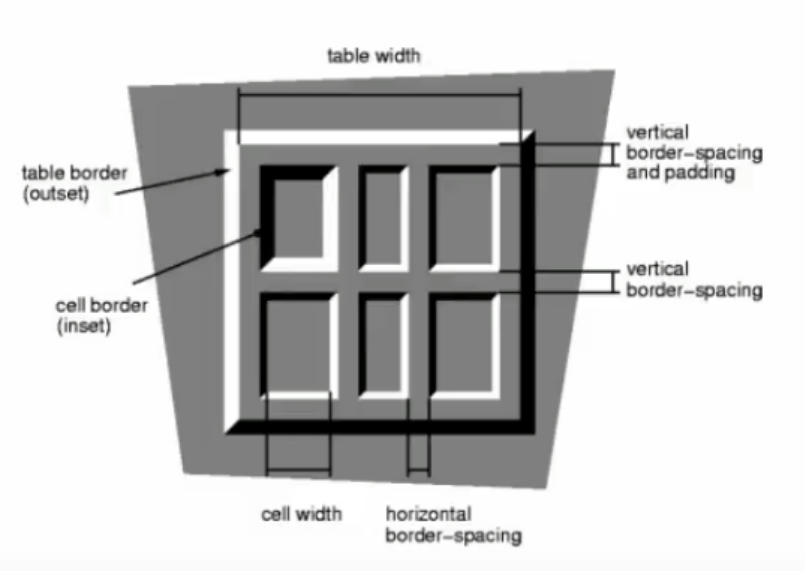
表格CSS属性

设置单元格之间的间距，使用这个，尽量不要使用上面的

border-spacing用于设置单元格之间的水平、垂直间距。

- 2个值分别是cell之间的水平、垂直间距
- 如果只设置一个值，同时代表水平、垂直间距

```
table {
  border-sapcing: 10px 20px;
}
```



表单

表单常用元素：

- form 表单 一般情况下，其他表单元素都是它的后代元素
- input 单行文本框、单选框、复选框、按钮等元素
- textarea 多行文本框
- select、option 下拉选择框
- button 按钮
- label 表单元素的标题
- fieldset 表单元素组
- legend fieldset的标题

input元素

input也是行内元素，准确地说是行内替换元素。与img元素相似，页面最终展示的不是img元素而是img中对应的那张图片，使用图片替换掉了img元素。input也是，不展示编写的input，而是展成一个框。基本上所有的替换元素都是行内元素。

input的type类型：

- text 文本输入框（明文输入）
- password 文本输入框（密文输入）
- radio 单选框
- checkbox 复选框
- button 按钮
- reset 重置
- submit 提交表单数据给服务器
- file 文件上传

input其他属性：

- maxlength 允许输入的最大字数
- placeholder 占位文字
- readonly 只读
- disabled 禁用
- checked 默认被选中
- selected 下拉框默认选中选项
- multiple 下拉框多选
- autofocus 当页面加载时，自动聚焦
- name 名字
- value 取值
- tableindex
- form 设置所属的form元素（填写form元素的id），一旦使用了此属性，input元素即使不在for元素内部，它的数据也能提交给服务器。

布尔属性：

布尔属性可以没有属性值，写上属性就代表使用了这个属性。常见的布尔属性有，diasbled、checked、eadonly、multiple、autofocus、selected。

如果要给这些布尔属性设置值，值就是属性名本身。

```
<!-- 以下两种写法是等价的，建议采用第一种 -->
<input type="text" readonly disabled>
<input type="radio" checked>

<input type="text" readonly="readonly" disabled="disabled">
<input type="radio" checked="checked">
```

按钮的两种实现方式，在form中button按钮如果不设置任何属性，默认的作用是重置。但如果form置了action属性，那么默认行为是提交

```
 <!-- reset不写value，浏览器会默认赋值 -->
<input type="reset" value="重置">
<button>重置</button>
```

label可以跟某个input绑定，点击label就可以激活对应的input；有两种写法

```
<div>
  <label for="phone">手机</label>
  <input type="text" value="" id="phone">
</div>

<label for="phone">
  手机
  <input type="text" value="" id="phone">
</label>
```

```
<div>
  <span>性别</span>
  <label for="male">男</label>
  <input type="radio" name="sex" id="male">
  <label for="female">女</label>
  <input type="radio" name="sex" id="female">
</div>
```

input去除边框，设置CSS属性outline: none;

例子:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <form action="" method="post">
    <fieldset>
      <legend>必填信息</legend>
      <div>
        <label for="phone">手机</label>
        <input type="text" value="" id="phone">
      </div>

      <div>
        <span>密码</span>
        <input type="password">
      </div>

      <div>
        <span>验证码</span>
        <input type="text">
        <input type="submit" value="获取验证码">
      </div>
    </fieldset>
    <fieldset>
      <legend>选填信息</legend>
      <div>
        <span>照片</span>
        <input type="file">
      </div>
      <div>
        <span>性别</span>
        <label for="male">男</label>
        <input type="radio" name="sex" id="male">
        <label for="female">女</label>
        <input type="radio" name="sex" id="female">
      </div>
```

```
    <div>
      <span>爱好</span>
      唱<input type="checkbox" name="hobby">
      跳<input type="checkbox" name="hobby">
      rap<input type="checkbox" name="hobby">
      篮球<input type="checkbox" name="hobby">
    </div>
    <div>
      <span>学历</span>
      <select name="" id="">
        <option value="0">小学</option>
        <option value="1">初中</option>
        <option value="2">高中</option>
      </select>
    </div>
    <div>
      <span>简介</span>
      <textarea name="" id="" cols="10" rows="3"></textarea>
    </div>
  </fieldset>
  <!-- reset不写value，浏览器会默认赋值 -->
  <input type="reset" value="重置">
  <!-- button按钮有一个默认的reset属性 -->
  <button>重置</button>
  <input type="submit" value="提交">
 </form>
</body>

</html>
```

textarea元素

常用属性：

- cols 列数
- rows 行数

缩放的css设置

- 禁止缩放　resize: none
- 水平缩放　resize: horizontal
- 垂直缩放　resize: vertical
- 水平垂直缩放（默认）　resize: both

select和option

option是select的子元素，一个option代表一个选项

select常用属性：

- multiple 可以多选

- size 同时显示多少项

option常用属性

- selected 默认被选中

form元素

常用属性：

- action 提交地址
- method 提交方法
- target 跳转方式
  - _blank：打开新页面跳转
  - _self在本页面跳转
- enctype 规定了在向服务器发送表单数据之前如何对数据进行编码
  - application/x-www-form-urlencoded 默认的编码方式
  - multipart/form-data 文件上传时必须为这个值，并且method为post
  - text/plain 普通文本传输
- accept 规定表单提交时使用的字符编码（默认UNKNOW，和文档相同的编码）

Emmet语法

！ 和 html:x

使用!或html[:x]，x可以不写，或5或xml，能够直接生成html模板。

> 子代 和 + 兄弟

```
<!-- 结构 div>p>span>strong -->
<div>
  <p><span><strong></strong></span></p>
</div>

<!-- 结构 h2+div+p+ -->
<h2></h2>
<div></div>
<p></p>

<!-- div>h2+a+p>span -->
<div>
  <h2></h2>
  <a href=""></a>
  <p><span></span></p>
</div>
```

* 多个 和 ^ 上一级

```
<!-- div>p*3 -->
<div>
  <p></p>
  <p></p>
  <p></p>
</div>

<!-- div>span^div>p -->
<div><span></span></div>
<div>
  <p></p>
</div>

<!-- div>p*2>span^^h1+strong -->
<div>
  <p><span></span></p>
  <p><span></span></p>
</div>
<h1></h1>
<strong></strong>
```

() 分组

```
<!-- div>(p>span)+h2+strong -->
<p><span></span></p>
<h2></h2>
<strong></strong>

<!-- 转换 div>p*2>span^^h1+strong -->
<!-- 为 (div>(p*2>span))+h1+strong -->
<div>
  <p><span></span></p>
  <p><span></span></p>
</div>
<h1></h1>
<strong></strong>
```

感觉还是找上一层^好用。

属性(id属性、class属性、普通属性) 和 数字

```
<!-- div#main -->
<div id="main"></div>

<!-- div.box  -->
<div class="box"></div>

<!-- div[title] -->
<div title=""></div>

<!-- div[title="哈哈"] -->
```

```
<div title="哈哈"></div>

<!-- 添加多个属性 div#main.box1.box2[title="test"]-->
  <div id="main" class="box1 box2" title="test"></div>

<!-- 练习div#main>div.box+p.p1+span.title^div#footer>div.box2 -->
  <div id="main">
    <div class="box"></div>
    <p class="p1"></p>
    <span class="title"></span>
  </div>
  <div id="footer">
    <div class="box2"></div>
  </div>
```

{} 内容

```
<!-- div.box{我是内容} -->
<div class="box">我是内容</div>
```

$ 数字

```
<!-- 属性数字 div.box$*4 -->
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
<div class="box4"></div>

<!-- ul>li.item$$$*5 -->
<ul>
  <li class="item001"></li>
  <li class="item002"></li>
  <li class="item003"></li>
  <li class="item004"></li>
  <li class="item005"></li>
</ul>


<!--  内容数字 div.box{我是内容$}*3 -->
<div class="box">我是内容1</div>
<div class="box">我是内容2</div>
<div class="box">我是内容3</div>
```

隐式标签

一些约定俗成，挡在不指定元素的时候，会根据情况生成元素。

一般情况下，隐式标签是代表div的，不过也可以代表li或table

```
<!-- #main -->
<div id="main"></div>
```

```
<!-- .box -->
<div class="box"></div>

<!-- div>.wrap>.content -->
<div>
  <div class="wrap">
    <div class="content"></div>
  </div>
</div>

<!-- ul>.item${列表元素$}*3 -->
<ul>
  <li class="item1">列表元素1</li>
  <li class="item2">列表元素2</li>
  <li class="item3">列表元素3</li>
</ul>


<!-- table>#row$*4>[colspan=2] -->
<table>
  <tr id="row1">
    <td colspan="2"></td>
  </tr>
  <tr id="row2">
    <td colspan="2"></td>
  </tr>
  <tr id="row3">
    <td colspan="2"></td>
  </tr>
  <tr id="row4">
    <td colspan="2"></td>
  </tr>
</table>
```

CSS的Emmet语法

部分举例。

```
.box {
  /* w200+h150+m20+p30 */
  /* w200 */
  width: 200px;
  /* h200 */
  height: 150px;
  /* m20  */
  margin: 20px;
  /* p30  */
  padding: 30px;
}

.box2 {
  /* m20-20-40-50 */
  margin: 20px 20px 40px 50px;
}
```

```
.box3 {
  /* p-10-20--30 */
  padding: -10px 20px -30px;
}

.box4 {
  /* m10px20px */
  margin: 10px 20px;
}

.box5 {
  /* m10px-20 */
  margin: 10px -20px;
}
```

字体的

```
.box{
 /* fz20 */
 font-size: 20px;

  /* fz1.5 */
  font-size: 1.5em;

  /* fw700 */
  font-weight: 700;
  /* lh40 */
  line-height: 40;
  /* lh40px */
  line-height: 40px;

  /* bgc#333 */
  background-color: #333333;
}
```

CSS

CSS全称是Cascading Style Sheets。

CSS3：是CSS2.x以后对某一些CSS模块进行升级更新后的称呼，比如CSS Color Module Level 3、Select Level 3、CSS Namespaces Module Level 3。目前并不存在真正意义的CSS 3。

常见的CSS属性的具体用途，大致可以分类为：

- 文本：color、direction、letter-spacing、word-spacing、line-height、text-align、text-indent、text-transform、text-decoration、white-space
- 字体：font、font-family、font-style、font-variant、font-weight
- 背景：background、background-color、background-image、background-repeat、background-attachment、background-position
- 盒子模型：width、height、border、margin、padding
- 列表：list-style

- 表格：border-collapse
  - 显示：display、visibility、overflow、opacity、filter
  - 定位：vertical-align、position、left、top、right、bottom、float、clear

CSS元素类型

元素类型可以用两种方式进行划分

块级、行内级元素

根据元素的显示类型，即能否在同一行显示，HTML元素可以主要分为两大类。

- 块级元素　block-level elements
  - **独占父元素一行。**即==块级元素的宽度独占父级元素一整行==。高度会由内容撑起来。
  - 如div、p、pre、h1~h6、ul、ol、li、dl、dt、dd、table、form、article、aside、footer header、hgroup、main、nav、section、blockquote、hr等
- 行内级元素　inline-level elements
  - **多个行内级元素可以在父元素的同一行中显示**
  - 如a、img、span、strong、code、iframe、label、input、button、canvas、embed、object、video、audio等

例子

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      width: 500px;
      height: 500px;
      background-color: orange;
    }

    .inner {
      height: 30px;
      background-color: red;
    }

    p {
      background-color: palegreen;
    }
  </style>
</head>
<body>
  <!-- 块级元素 -->
```

```
  <div class="box">
    <div class="inner"></div>
    <p>我是段落</p>
  </div>

  <!-- 行内级元素 -->
  <span>我是span元素</span>
  <strong>我是strong元素</strong>
  <input type="text">
</body>
</html>
```

替换、非替换元素

根据元素的内容类型，即是否浏览器会替换掉元素，HTML元素可以主要分为2大类

- 替换元素　replaced elements
  - 元素本身没有实际内容，浏览器会根据元素的类型和属性，来决定元素的具体显示内容
  - 如img、input、iframe、video、embed、canvas、audio、object等
- 非替换元素　non-replaced elements
  - 和替换元素相反，元素本身是有实际内容的，浏览器会直接将其内容显示出来，而不需要根元素类型和属性来判断到底显示什么内容
  - 如div、p、pre、h1~h6、ul、ol、li、dl、dt、dd、table、form、article、aside、footer header、hgroup、main、nav、section、blockquote、hr、a、strong、span、code、label等

行内非替换元素注意点

以下属性对行内非替换元素不起作用

- width
- height
- margin-top
- margin-bottom

下面例子中margin-left生效，而margin-top是无效的。width和height也是无效的。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    span {
      background-color: red;
      width: 100px;
      height: 100px;
      margin-left: 20px;
```

```
    margin-top: 100px;
  }
 </style>
</head>
<body>
 <span>span1</span>
 <span>span2</span>
</body>
</html>
```

以下属性对行内元素比较特殊：

- padding-top\padding-bottom
- border-top\border-bottom

这几个属性设置后上下会多出来区域，但不会占据空间。给它们设置成行内块级元素display: inline-block可以解决此问题。

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Document</title>
 <style>
  .outer {
   margin-bottom: 100px;
  }

  span {
   background-color: red;
  }

  .span1 {
   padding-left: 20px;
   padding-bottom: 10px;
   padding-top: 20px;
   /* display: inline-block; */
  }

  .box1 {
   background-color: green;
   width: 100px;
   height: 50px;
  }

  .span2 {
   border-right: 10px solid purple;
   border-top: 10px solid blue;
   border-bottom: 10px solid yellow;
   /* display: inline-block; */
  }
 </style>
```

```
</head>

<body>
  <div class="outer">
    <span class="span1">span1</span>
    <span class="span1">span2</span>
    <div class="box1">div</div>
    <span class="span1">span3</span>
  </div>

  <div class="outer">
    <span class="span2">span1</span>
    <span class="span2">span2</span>
    <div class="box1">div</div>
    <span class="span2">span3</span>
  </div>
</body>

</html>
```

元素分类总结

```
<table>
    <tr>
      <td colspan="2">元素分类</td>
      <td>具体元素</td>
      <td>默认特性</td>
    </tr>
    <tr>
      <td rowspan="2">块级元素<br /> (block-level elements) </td>
      <td>替换元素<br />(replaced elements)</td>
      <td></td>
      <td></td>
    </tr>
    <tr>
      <td>非替换元素(non-replaced elements)</td>
      <td>div、p、pre、h1~h6、ul、ol、li、dl、dt、dd、table、form、article、aside、footer、
eader、hgroup、main、nav、section、blockquote、hr
      </td>
      <td>独占父元素一行<br />可以随意设置宽高<br />高度默认由内容决定</td>
    </tr>
    <tr>
      <td rowspan="2">行内级元素<br /> (inline-level elements) </td>
      <td>替换元素<br />(replaced elements)</td>
      <td>img、input、iframe、video、embed、canvas、audio、object等</td>
      <td>跟其他行内级元素在同一行显示<br />可以随意设置宽高</td>
    </tr>
    <tr>
      <td>非替换元素(non-replaced elements)</td>
      <td>a、strong、span、code、label等</td>
      <td>跟其他行内元素在同一行显示<br />不可以随意设置宽高<br />宽高由内容决定</td>
    </tr>
  </table>
```

元素之间的空格

行内级元素（包括inline-block元素）的代码之间如果有空格，会被解析显示为空格。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    /* 方法3 */
    body {
      font-size: 0;
    }
    span, strong, div {
      font-size: 16px;

      float: left; /* 方法4 */
    }


    span {
      background-color: blue;
    }
    strong {
      background-color: coral;
    }
    div {
      display: inline-block;
      background-color: red;
    }
  </style>
</head>
<body>
  <div>
    <span>span</span>
    <strong>strong</strong>
    <div>我是div元素</div>
  </div>

  <!-- 方法1 -->
  <div>
    <span>span</span><strong>strong</strong><div>我是div元素</div>
  </div>

  <!-- 方法2 -->
  <div>
    <span>span</span><!--
--><strong>strong</strong><!--
--><div>我是div元素</div>
  </div>
</body>
</html>
```

为什么会产生空格，因为浏览器在解析多个空格或换行符时，会将其解析成一个空格。上面的第一个div中span、strong、div各占一行，三者之间的空格被解析成了一个空格。第二个div中三者之间没有空格，浏览器中显示也没有空格。

目前存在的解决方案：

1. 元素之间不要留空格（不建议）
2. 元素之间写注释（繁琐，不建议）
3. 设置父级元素 font-size: 0，然后在元素中重新设置自己需要的font-size（不推荐）
    - 此方法不使用Safari
4. 给元素加浮动，一般都是使用这种办法。

元素之间的嵌套关系

块级元素、行内块元素inline-block：

- 一般情况下，可以嵌套任意的元素。如块级元素、行内级元素、 inline-block元素
- 特殊情况，p元素不能包含其他块级元素

行内级元素（如span、a、strong等）

- 一般情况下，只能包含行内元素

CSS样式应用

CSS样式应用到元素上有3种方法：

- 内联样式 （inline style）
- 文档样式表（document style sheet）、内嵌样式表（embed style sheet）
- 外部样式表 （external style sheet）

在Chrome中的调试窗口可以看到有些样式右上角标注user agent stylesheet，这是用户代理样式，浏览器默认的标签样式。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="./css/style.css">
  <style>
    /* 文档样式表 */
    .document-style {
      color: blue;
      font-size: 60px;
    }
  </style>
</head>
```

```
<body>
  <!-- 内联样式 inline -->
  <h1 style="color: red; font-size: 50px">哈哈哈</h1>

  <!-- 文档样式表 document style sheet -->
  <h1 class="document-style">呵呵呵</h1>

  <!-- 外部样式表 （external style sheet） -->
  <h1 class="external-style">咯咯咯</h1>
</body>
</html>
```

css编码设置

关于外部样式表，推荐使用utf-8

这样可以避免浏览器解析样式，出现中文时的编解码错误。如：font-family: '华文宋体'，如果不设置码，可能就会出错。

```
@charset: "utf-8"
```

使用@import导入外部样式

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    @import: url('./css/style.css');
  </style>
</head>
<body>
  <h1 class="external-style">咯咯咯</h1>
</body>
</html>
```

使用@import引入和<link>进行引入都可以。

设置网页图标

```
<link rel="icon" type="image/x-icon" href="">
```

link元素可以用来设置网页的图标

- link元素的 rel属性不能省略，用来指定文档与链接资源的关系
- 一般rel若确定，响应的type也会默认确定，所以可以省略type
- 网页图标支持图片格式是 ico、png，常用大小是16*16、24*24、32*32像素

CSS选择器 （selector）

CSS选择器就是按照一定的规则，选择出符合条件的元素，为之添加CSS样式。

选择器种类大概可以分为：

- 通用选择器（universal selector）
- 元素选择器（type selectors）
- 类选择器（class selectors）
- id选择器（id selectors）
- 属性选择器（attribute selectors）
- 组合（combinators）
- 伪类（pseudo-classes）
- 伪元素（pseudo-elements）

通用选择器

选择所有的元素。

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

一般是用来给所有元素做一些通用性的设置。

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

元素选择器

选择指定的元素。

```
div {
  color: red;
}
```

class选择器

注意点：

- 一个元素可以有多个class，每个class之间用空格隔开
- class值如果由多个单词组成，`单词之间可以使用中划线

id选择器

通过id选择选择指定元素，每个id值在一个文件中应该只出现一次。

属性选择器

根据属性进行选择，比如

```
[title="test"] {
  color: red;
}

[title*="test"] {
  color: red;
}
```

后代选择器

又叫组合选择器，包含直接和间接所有的。

```
<style>
div span {
  color: red;
}
</style>

<span>文字内容1</span>
<div class="box">
  <span>文字内容2</span>
  <p>
    <span>文字内容3</span>
  </p>
  <div>
    <span>文字内容4</span>
  </div>
  <span>文字内容5</span>
</div>
<span>文字内容6</span>
```

选中div下面span元素，所以选中的是2/3/4/5。

子代选择器

选择直接的子元素，只包含子代的。

```
<style>
div > span {
  color: red;
}
</style>

<span>文字内容1</span>
<div class="box">
  <span>文字内容2</span>
  <p>
    <span>文字内容3</span>
  </p>
  <div>
    <span>文字内容4</span>
```

```
  </div>
  <span>文字内容5</span>
</div>
<span>文字内容6</span>
```

所以符合规则的是2/4/5

p标签中不可以包含div，包含的话结构会直接乱掉。

相邻兄弟选择器

div元素后面紧挨着的元素（且div、p元素必须是兄弟关系）

```
<style>
  div+p {
    color: red;
  }
</style>
<p>文字内容1</p>
<div>
  <p>文字内容2</p>
</div>
<p>文字内容3</p>
<p>文字内容4</p>
```

符合规则的是3

全体兄弟选择器

div元素后面的p元素（且div、p元素必须是兄弟关系）

```
<style>
  div~p {
    color: red;
  }
</style>
<p>文字内容1</p>
<div>
  <p>文字内容2</p>
</div>
<p>文字内容3</p>
<p>文字内容4</p>
```

符合规则的是3/4

选择器组

交集选择器

同时符合2个条件的元素：div元素、class值有one的元素

```
 <style>
```

```
    div.one[title="test"] {
      color: red;
    }
  </style>
  <div class="one">
    <p>文字内容1</p>
  </div>
  <div class="two">文字内容2</div>
  <p class="one">文字内容3</p>
    <div class="one" title="test">文字内容5</div>
</body>
```

符合规则的是5

并集选择器

所有的div元素 + 所有class值有one的元素 + 所有title属性值等于test的元素。

```
<style>
  div, .one, [title="test"] {
    color: red;
  }
</style>
<div class="one">文字内容1</div>
<span title="test">文字内容2</span>
<p class="one">文字内容3</p>
```

符合条件的有1/2/3

伪类选择器（pseudo-classes）

常见的伪类有

- 动态伪类（dynamci pseudo-classes）
  - :link、:visited、:hover、:active、:focus
- 目标伪类（target pseudo-classes）
  - :target
- 语言伪类(language pseudo-classes) 使用较少
  - :lang()
- 元素状态伪类(UI element state pseudo-classes)
  - :enabled、:disabled、:checked
- 结构伪类(structural pseudo-classes)
  - :nth-child()、:nth-last-child()、:nth-of-type()、:nth-last-of-type()
  - :first-child、:last-child、first-of-type、:last-of-type
  - :root、:only-child、:only-of-type、:empty

- 否定伪类(negation pseudo-classes)
    - :not()

目标伪类

使用较少，一般在锚点中使用。

比如点击某个锚点，让选中的标题变颜色。

```
/* 选中的锚点字体变成红色 */
:target {
  color: red;
}
```

元素状态伪类

使用较少

设置某个元素处于某种状态时设置样式

```
<style>
  button:enabled {
    color: green;
  }

  button:disabled {
    color: blue;
  }


</style>
<button>我是按钮1</button>
<button disabled>我是按钮2</button>
```

设置按钮在可用时是绿色，不可用时是蓝色。

动态伪类

a元素上的使用

使用举例，可以在a链接上设置。

- a:link：未访问的链接
- a:visited：已访问的链接
- a:hover：鼠标移动到链接上
- a:active：激活的链接（鼠标在链接上长按住未松开）

```
<style>
  /* 给a元素设置样式，会应用到所有伪类 */
```

```
 a {
    font-size: 20px
 }

 a:link {
   color: red;
 }

 a:visited {
   color: gray;
 }

 a:hover {
   color: blue;
 }

 a:active {
   color: orange;
 }
</style>
<a href="">Google一下</a>
```

如果给a元素设置样式，相当于给a元素的所有动态伪类都设置了。

要注意：

- :hover必须放在:link和:visited后面才能生效。
- :active必须放在:hover后面才能完全生效。

官方推荐的顺序是上面的顺序link visted hover acitve。

记忆小技巧：女朋友看到LV后，ha ha 大笑，哈哈哈。

其他元素上的使用

:hover、:active也可以用到其他元素上：

```
<style>
 strong:hover {
   color: blue;
 }

 strong:active {
   font-size: 30px;
 }

</style>
<strong>哈哈</strong>
```

:focus的使用

:focus指当前拥有输入焦点的元素（能接收键盘输入）

文本输入框聚焦后，北京变为灰色

```
input:foucs {
  background: gray;
}
```

因为链接a元素可以被键盘Tab键选中聚焦，所以:focus也适用于a元素

```
<style>
  a:focus {
    color: yellowgreen;
  }
</style>
<a href="">哈哈</a>
```

所以此时链接a上可以有五种动态伪类，建议的编写顺序为：

:link、:visited、:focus、:hover、:active

记忆：女朋友看到LV包包后，Feng了一样地ha ha大笑。

去掉a元素的 :focus

开发过程中，a标签很多时候不希望获得foucs获取焦点。可以使用一些css的方法：

```
a:focus {
  outline: none
}
```

这种方法其实是选中了，只是看不出来效果。

第二种办法，给a标签加上属性tabindex，这个属性是添加Tab键的选中顺序。

```
<a tabindex="-1" href="">Google一下</a>
```

设置成-1之后，Tab键是无法选中的。

结构伪类

:nth-child()

子元素选择器，选中第几个子元素改变样式。

```
<style>
  /*
     交集选择器
    * 是一个p元素
    * 同时p元素作为子元素的第三个元素
  */
  p:nth-child(3) {
    color: red;
  }
</style>
```

```
<body>
  <div>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
    <p>文字内容5</p>
  </div>
  <div>我是个div</div>
  <p>文字内容6</p>
</body>
```

文字内容3和6会变为红色。

:nth-child()根据n选择设置颜色，可以根据自己的需求，进行相关的计算。

```
<style>
  /* 所有p的子元素设置为绿色 */
  p:nth-child(n) {
    color: green;
  }

  /* 偶数p红色 */
  p:nth-child(2n) {
    color: red;
  }

  /* 偶数p红色 */
  p:nth-child(even) {
    color: red;
  }

  /* 奇数p蓝色 */
  p:nth-child(2n+1) {
    color: blue;
  }

  /* 奇数p蓝色 */
  p:nth-child(odd) {
    color: blue;
  }

  /* 选中前5个设置紫色 */
  p:nth-child(-n+5) {
    color: purple;
  }
</style>
<div>
  <p>文字内容1</p>
  <p>文字内容2</p>
  <p>文字内容3</p>
  <p>文字内容4</p>
  <p>文字内容5</p>
  <p>文字内容6</p>
```

```
  <p>文字内容7</p>
  <p>文字内容8</p>
  <p>文字内容9</p>
</div>
```

一个例子

```
<style>
  p:nth-child(4n+1) {
    color: red;
  }

  p:nth-child(4n+2) {
    color: blue;
  }

  p:nth-child(4n+3) {
    color: green;
  }

  p:nth-child(4n+4) {
    color: orange;
  }
</style>
<body>
  <div>
    <div>div</div>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
    <span>span</span>
    <p>文字内容5</p>
    <p>文字内容6</p>
  </div>
</body>
```

p标签的颜色从上至下依次是：蓝、绿色、橘色、红色、绿色、橘色。

:nth-last-child()

:nth-child()是正着数，:nth-last-child()是倒着数。

其他的特性和:nth-child()是一样的。

:nth-last-child(1)倒数第一个子元素；

:nth-last-child(2n)选中偶数的子元素；

nth-last-child(-n+2)，代表最后两个子元素。

:nth-of-type()

:nth-of-type()用法和:nth-child()类似，不同点是:nth-of-type()计数时只计算同种类型的元素。

忽略其他类型，只选择同类的子元素进行样式添加。

```
<style>
  p:nth-child(3) {
    color: blue;
  }

  p:nth-of-type(3) {
    color: red;
  }
</style>
<body>
  <div>
    <div>div</div>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
    <span>span</span>
    <p>文字内容5</p>
    <p>文字内容6</p>
  </div>
</body>
```

文字内容2变成蓝色，文字内容3变成红色。:nth-child()会在所有的子元素中进行选择，:nth-of-type()会在同类的子元素中进行选择。

nth-of-type(2n)或:nth-of-type(2n+1)表示从相同的子类中选择偶数或奇数。使用同上面的是一个理。

假如说这样写就表示找每个类型的偶数。

```
:nth-of-type(2n) {
  color:red;
}
```

再来看这例子，样式改变是什么哪些：

```
<style>
  p:nth-of-type(2) {
    color: red;
  }
</style>

<body>

  <div>
    <p>文字内容1</p>
    <div>
      <p>文字内容2</p>
    </div>
    <div>
      <div>
```

```
      <p>文字内容3</p>
    </div>
    <p>文字内容4</p>
    <p>文字内容5</p>
  </div>
  <p>文字内容6</p>
 </div>

</body>
```

变红色字体的是5和6

:nth-last-of-type()

同理nth-of-type()，唯一不同是，倒过来找。不再赘述。

其他伪类

- :first-child 等同于:nth-child(1)
- :last-child等同于:nth-last-child(1)
- :first-of-type等同于:nth-last-of-type(1)
- :only-child是父元素中唯一的子元素，选中。

```
<style>
 body :only-child {
  color: red;
 }
</style>
<!-- 文字内容2和3变红 -->
<body>
 <p>文字内容1</p>
 <div>
  <p>文字内容2</p>
 </div>
 <div>
  <div>
   <p>文字内容3</p>
  </div>
  <p>文字内容4</p>
  <p>文字内容5</p>
 </div>
 <p>文字内容6</p>

</body>
```

- only-of-type，是父元素中唯一的这种类型的子元素。

```
<style>
 body :only-of-type {
  color: red;
 }
</style>
```

```
<!-- 文字内容1、4和5变红 -->
<body>
  <div>
    <p>
      <span>文字内容1</span>
    </p>
    <div>文字内容2</div>
    <div>文字内容3</div>
  </div>
  <div>
    <strong>文字内容4</strong>
    <a href="">
      <span>文字内容5</span>
    </a>
  </div>
</body>
```

- :root根元素，也就是html元素。下面两种写法等价

```
html {

}

:root {

}
```

- :empty 选中元素内容为空的元素。

```
<style>
  :empty {
    height: 20px;
    background-color: red;
  }
</style>
<!-- 空元素p和div会被选中 -->
<body>
  <div>
    <p></p>
    <span>文字内容1</span>
  </div>
  <div>
    <strong>文字内容2</strong>
    <a href="">文字内容3</a>
    <div></div>
  </div>
</body>
```

:not() 否定伪类

:not()的格式是:not(x)，表示除x以外的元素。

x是一个选择器，可以是：元素选择器、通用选择器、属性选择器、类选择器、id选择器、伪类（除定伪类）。

```
<style>
  body :not(div) {
    color: red;
  }
</style>
<!-- 文字内容2和3变红 -->
<body>
  <div>文字内容1</div>
  <p>文字内容2</p>
  <span>文字内容3</span>
  <div>文字内容4</div>
</body>
```

:not()中的参数也可以是类名或者id名

伪元素选择器 (pseudo-elements)

伪元素可以看成是行内元素。

常用的伪元素有：

- :first-line、::first-line
- :fitst-letter、::first-letter
- :before、::before
- :after、::after

为了区分伪元素和伪类，建议伪元素使用2个冒号，如::first-line

::first-line

选中第一行，针对首行文本设置属性

```
div::first-line {
  color: blue;
  text-decoration: underline;
}
```

只有下列属性可以应用到::first-line上：

- 字体属性、颜色属性、背景属性
- word-spacing、letter-spacing、text-decoration、text-transform、line-height

::first-letter

选中第一个字母，针对首字母设置属性

```
div::first-letter {
  color: blue;
  font-size: 30px;
}
```

只有下列属性可以应用在::first-letter上：

- 字体属性、margin属性、padding属性、border属性、颜色属性、背景属性
- text-decoration、text-transform、letter-spacing、word-spacing(适当的时候)、line-height、loat、vertical-align(只有当float是none时)

::before 和 ::after

::before和::after用来在一个元素内容之前或之后插入其他内容(可以是文字、图片)。

```
  span::before {
    content: "1";
    color: red;
    margin-right: 5px;
  }

  span::after {
    content: "abc";
    color: purple;
    margin-left: 5px;
  }
</style>

<body>
  <span>我是一个span</span>
</body>
```

这个元素中，不能省略content属性，即使里面是空字符串，属性是不能删除的。

```
div::before {
  content: "";
  display: inline-block;
  width: 100px;
  height: 20px;
  background-color: red;
}

div::before {
  content: url("bg001.png")
}
```

CSS常用属性

- color 前景色，不止字体颜色，例如还可以作用border边框，或text-decoration:line-through。
- font-size 字体大小
- background-color 背景颜色
- width/height 宽高

宽度和高度不适用与非替换行内元素

颜色设置的方法

1. 基本颜色关键字red、black、yellow、blue等，只有上百种基本颜色的关键字。
2. RGB颜色十进制：rgb()
3. RGBA颜色：和十进制之前相同，a指alpha，透明度。如：rgb()
4. RGB颜色十六进制：#FFFFFF

查看网站布局的技巧

在浏览器中打开开发者工具，然后添加一个新的div全局样式属性，增加outline就可以进行布局结构查看

```
div {
    outline: 2px solid red !important;
}
```

CSS属性-文本

文本 text-transform

转换文本大小写

文本 text-decoration

text-decoration 用于设置文字的装饰线

很多时候用来去掉a标签的下划线。

```
a {
  text-decoration: none;
}
```

还有就是比如用来加删除划线，划掉原来的价格。

html中的u标签，是设置了这个属性

```
/* u标签默认样式 */
u {
    text-decoration: underline;
}
```

文本 letter-sapcing

设置之母之间的间距

文本 word-spacing

设置单词之间的间距

文本 word-break

指定非中日韩脚本的单词换行规则。

文本 text-ident

设置文本首行缩进。

可以使用但是em进行首行缩进。比如说，一段文本中的字体大小是20px，想要首行缩进2个字，可使用text-indent: 40px;或text-indet: 2em;

```
p {
  text-indent: 40px;
  text-indet: 2em;
}
```

文本 text-align

设置元素内容在元素中的水平对齐方式。

```
div {
  background-color: #0f0;
  text-align: center;
}
```

可以用户设置div中的元素，文本，图片等。

可以设置元素内容居中，但是不能设置div居中。因为div是块级元素，父级的盒子会认为它自己内部盒子本身宽度和自己是一致的，所以是不会对内部的盒子产生作用。

```
<style>
    .box {
      background-color: #0f0;
      text-align: center;
    }

    .inner {
      background-color: purple;
      width: 200px;
    }
</style>

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

如果不设置inner的宽度，看起来好现实box中设置的元素内容居中作用在了inner的div盒子上，其实是的，由于继承，inner盒子也有了文本设置属性，他只是作用在了inner盒子的内容上。

可以这样改

```
.inner {
  background-color: purple;
```

```
  width: 200px;
  display: inline-block
}
```

文本 text-shadow

text-shadow用法与box-shadow类似，是用来设置文字阴影的。

text-shadow同样适用于伪元素::first-line、::first-letter

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    h1 {
      text-shadow: 5px 5px 3px orange;
    }

    p::first-line {
      text-shadow: 3px 3px 3px green;
    }

    p::first-letter {
      font-size: 25px;
      text-shadow: 5px 5px 5px purple;
    }
  </style>
</head>

<body>
  <h1>你好啊</h1>
  <p>
    无论是灾区搜救的机器人、无人驾驶的汽车、还是监视地面影像的卫星，穿透云层和雾霾的能力都常有用。目前，科学家已开发出最先进的再成像系统LiDAR，通过配套的算法——可以测量单个光子运动。这项技术特别与众不同之处在于，它可以复原被障碍物散射和反射掉的光子。
  </p>
</body>

</html>
```

CSS属性-字体

Google浏览器设置字体，最小是12px，再小其实就看不清楚了。

浏览器默认字体的大小是浏览器自己设置的，例如chrome浏览器我们就可以自己在设置中进行字体式的相关设置。

字体 font-size

决定文字的大小。字体默认浏览器设置的大小是16px

常用设置：

具体数值+单位，如100px

也可以使用em单位，如1em代码100%，2em代表200%，0.5em代表50%。或者使用rem单位，现移动端使用很多。

```
<style>
    .box {
      font-size: 18px;
    }

    p {
      /* 相对于父级box的字体大小，所以字体大小是2*18px */
      font-size: 2em;
    }
  </style>
</head>
<body>
  <div class="box">
    <span>我是span元素</span>
    <p>我是段落，哈哈哈</p>
  </div>
</body>
```

字体 font-family

用于设置字体的类型。如微软雅黑

当设置某种字体的时候，浏览器会去读取操作系统中是否有这种字体，Windows下会去C:\Windows\onts目录下寻找设置的字体是否存在，如果有就会去使用这种字体。

如果没有这种字体，那么设置的字体将失效。为了防止设置的字体刚好操作系统不存在，一般会一次设置多个字体

```
font-family: Arial, Helvetica, sans-serif;
```

字体中间有空格，可以使用单引号包含起来。

浏览器会从左至右依次选择设置的字体，直到找到可用的字体，如果都不存在，那么浏览器会使用操系统的默认字体。

一般情况下，英文字体只适用于英文，中文字体同时适用于中文和英文。所以在开发中，中英文需要用不同的字体，可以将英文字体写在前面，中文字体写在后面。

```
div {
  font-family: "Courier New", "宋体"
}
```

字体 font-weight

设置文本的粗细。

使用某些html标签，如h1~h6、b、strong，其实是浏览器给这些标签设置了font-weight属性，显出来的字体就是加粗的。标签也是设置了css样式而已。

```
/* h1的默认样式 */
h1 {
    display: block;
    font-size: 2em;
    margin-block-start: 0.67em;
    margin-block-end: 0.67em;
    margin-inline-start: 0px;
    margin-inline-end: 0px;
    font-weight: bold;
}
/* b标签默认样式 */
b {
    font-weight: bold;
}
```

字体 font-style

用于设置文字的常规、斜体显示。

如使用i标签，是加上了这个属性。

```
i {
    font-style: italic;
}
```

我们很少使用这种标签，一般会使用字体样式来设置。

i标签使用很多，都会使用i标签和伪类配合，来设置小图标之类的。

字体 font-variant

影响小写字母的显示形式。

small-caps将小写字母替换为缩小过的大写字母规格。

字体 line-height

用于设置文本的最小行高。

简单理解

行高可以简单理解为一行文本所占据的高度。

例子：

```
<style>
  .box {
    background: green;
    font-size: 18px;
```

```
 }
</style>
<div class="box">
 我是div元素
</div>
```

设置的文本，在浏览器中显示，文本在盒子中占据了一定的高度，但要注意的是，**文本占据的高度并等于文字的高度**，我们的直观印象会认为，一行文本占据的高度等于文本的高度，但这是不一定的。

打开浏览器选中“我是div元素”文本，我们可以看到，默认情况下文本占据的高度其实是大于文字高度的，文本上下都多出来一部分区域。

上面的div我们并没有设置高度，不过我们都知道是有高度的，这个高度我们通常会说是里面的内容起来的。准确来说，撑起来div高度的就是文本的行高。

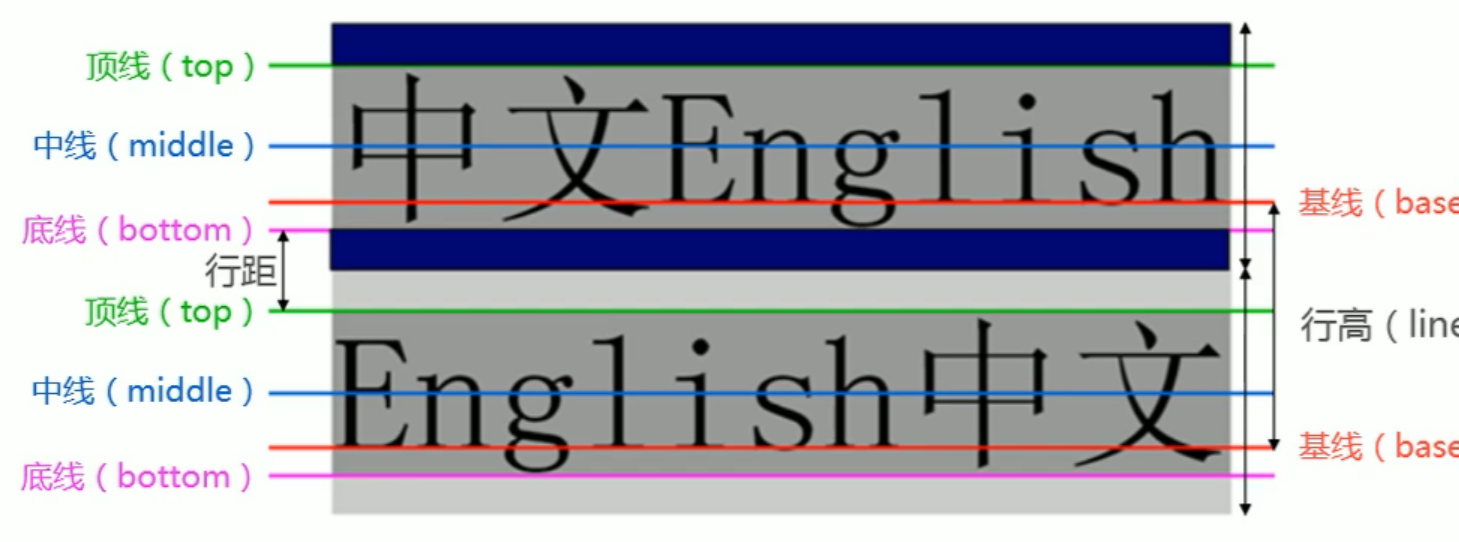
为什么要有行高呢？比如说我们在读一首诗时，这首诗是竖着读还是横着读，我们会自动根据这个诗体间的间距去决定该怎样去读。就是因为这些文字之间有间距，我们潜移默化地会去决定怎样读这段字。

严格定义

行高的严格定义是，两行文字基线(baseline)之间的间距。

baseline：与小写字母x最底部对齐的线。

行距：当前文本底线和下一行文本顶线之间的距离。



根据行高的定义，我们知道了行高就是右侧标出是行高的那根黑线即两条基线之间的高度，我们又可知道其实右侧三条黑色箭头代表的高度都是行高。

我们要注意区分height和line-height的区别：

- height：元素的整体高度
- line-height：元素中每一行文字所占据的高度

行高一个最常用的实例就是，让一行文字在div内部进行垂直居中。

```
<style>
  .box {
    font-size: 20px;
    height: 50px;
    line-height: 50px;
    background: green;
  }
</style>
<div class="box">
  我是div元素
</div>
```

上面的文字在div中是垂直居中的。

假如说当前的行高是30px，文本的高度是20px，现在行距就是10px，文本在排布的时候，上下距离5px。所以上面的例子，设置的行高应该就是div盒子的高度，那么文字正好是居中的，距离上下各15x。

所以文本为什么可以居中，就是因为行高设定之后，行距会等分。

这个时候，其实height是可以不写的，行高已经将盒子撑起来了。当然，没有文本时，行高也没有意。

字体 font 多写属性

可以同时设置多个属性

font: font-style font-variant font-weight font-size/line-height font-family

比如

```
.box {
  font-size: 18px;
  font-family: "宋体";
  font-weight: bold;
  font-style: oblique;
  font-variant: small-caps;
  line-height: 50px;
}
/* 可写成 */
.box {
  font: oblique small-caps bold 30px/50px "宋体";
}
```

其中font-style、font-variant、font-weight可以随意调换顺序，也可以省略，不设置。

/line-height可以省略，如果不省略，必须跟在font-size后面。

CSS属性-列表

列表常见的CSS属性有四个，列表属性用的很少。

- list-style-type

- list-style-image
  - list-style-position
  - list-style

它们都可以继承，所以设置给ol、ul元素，默认也会应用到li元素。

列表 list-style-type

设置li元素前面标记的样式

- **none** 去除标记样式
- dics 实心圆
- circle 空心圆
- square 实心方块
- decimal 阿拉伯数字
- lower-roman 小写罗马数字
- upper-roman 大写罗马数字

列表 list-style-image

设置某张图片为li元素前面的标记，会覆盖list-style-type的设置。

列表 list-style-position

设置li元素前面标记的位置：

- outside
- inside

列表 list-style

是list-tyle-type、list-tyle-image、list-tyle-position的缩写属性。

```
ul {
 list-style: outside url("image/dot.png");
}
```

一般最常用对还是直接设置为none，去掉前面的默认标记list-style: none。

CSS属性-表格

CSS属性-display

块级元素是浏览器给它默认加上了块级元素的属性。

display属性能够改变元素的显示类型，常见常用的：

- block 让元素显示为块级元素
- inline 让元素显示为行内级元素
- inline-block 让元素同时具备行内级、块级元素的特征。即既可以跟其他行内级元素在同一行显，也可以随意设置宽高。若没有设置宽高，宽高有内容决定。可以理解为，对外是一个行内级元素，内是一个块级元素。
- none 隐藏元素 元素不再占用空间

以下取值，等同于某些HTML元素：

- table <table> 一个block-level表格
- inline-table <table> 一个inline-level表格
- table-row <tr>
- table-row-group <tbody>
- table-header-group <thead>
- table-footer-group <tfoot>
- table-cell <td>、<th>，一个单元格
- table-caption <caption>， 表格的标题
- list-item

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    h1 {
      display: inline-block;
      width: 100px;
      height: 200px;
      background-color: red;
    }
    span {
      display: inline-block;
      width: 100px;
      height: 100px;
      background-color: blue;
    }
  </style>
</head>

<body>
  <!-- 非浮动元素是基线对齐的，看着会有点奇怪 -->
  <h1>你好</h1>
  <span>哈哈</span>
</body>
</html>
```

邮箱练习：

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    ul, li {
      list-style: none;
      margin: 0;
      padding: 0;
    }

    a {
      text-decoration: none;
      color: #000;
    }

    .email {
      width: 100px;
      border: 2px solid #999;
      text-align: center;
    }

    .email  .header {
      background-color: #999;
      color: #fff;
    }

    /* 移动到div.email上时，显示ul邮箱列表 */
    .email:hover > ul {
      display: block;
    }

    .email ul {
      display: none;
    }

    .email ul li:hover {
      background-color: rosybrown;
    }

    /* 将a标签设置成块级元素，宽度占满父级元素宽度，实现整行都显示小手光标 */
    .email ul li a {
      display: block;
    }

  </style>
</head>

<body>
```

```
  <div class="email">
    <div class="header">邮箱</div>
    <ul>
      <li><a href="">qq邮箱</a></li>
      <li><a href="">163邮箱</a></li>
      <li><a href="">gmail邮箱</a></li>
    </ul>
  </div>
</body>
</html>
```

分页练习：

CSS属性-visibility

visibility能控制元素的可见性，有2个常用值：

- visible 显示元素
- hidden 隐藏元素

visibility: hidden;和display: none; 的区别：

- visibility: hidden; 虽然元素看不见了，但元素的框依旧还留着，还会占着原来的位置
- display: none; 不仅元素看不见了，而且元素的框也会被移除，不会占着任何位置

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    div {
      width: 100px;
      height: 100px;
      background-color: red;
      display: none;
      /* visibility: hidden; */
    }
  </style>
</head>
<body>
  <div>
    我是div元素
  </div>
  <p>我是段落</p>
</body>
</html>
```

CSS属性-overflow

overflow用于控制内容溢出时的行为

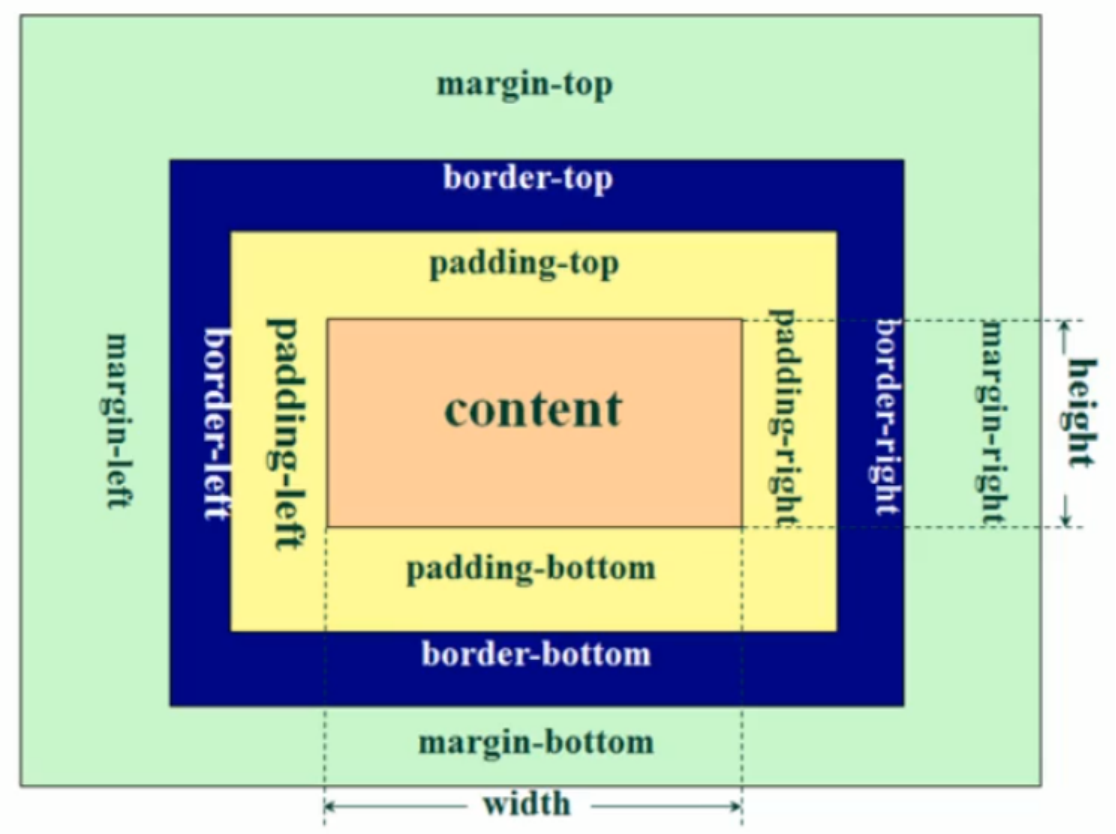
- visible 默认值，溢出的内容照样可见
  - hidden 溢出的内容直接剪裁，被隐藏
  - scroll 溢出的内容被剪裁隐藏，但可以通过滚动机制查看
  - auto 自动根据内容是否溢出来决定是否提供滚动机制

还有overflow-x和overflow-y可以分别设置水平垂直方向。建议还是直接使用overflow，因为目前overflow-x、overflow-y还没有成为标准，有些浏览器可能不支持。

CSS属性-盒子模型

HTML中每一个元素都可以看做是一个盒子。可以具备四个属性：

- 内容　content 盒子里面装的东西
- 内边距　padding 盒子边缘和里面装的东西之间的间距
- 边框　border 盒子的边框，边缘部分
- 外边距　margin 盒子和其他盒子之间的间距



内容 content

相关属性值：

- width 宽度
- min-width 最小宽度，无论内容多少，宽度都小于或等于其值。
- max-width 最大宽度，无论内容多少，宽度都大于或等于其值。常用于inline-block，用于达

一定宽度后换行。

- height 高度
- min-height 最小高度，无论内容多少，高度都小于或等于其值
- max-height 最大高度，无论内容多少，高度都大于或等于其值

内边距 padding

相关属性值：

- padding-left 左内边距
- padding-right 右内边距
- padding-top 上内边距
- padding-bottom 下内边距
- padding 上面四个的简写，顺序是上、右、下、左。

padding的规律：

- 四个值　padding: 10px 20px 30px 40px 上右下左
- 三个值　padding: 10px 20px 30px　上右下，左边跟随右边的值
- 两个值　padding:10px 20px　上和右，没有下，跟随上，没有左，跟随右
- 一个值　padding: 10px 上下左右都使用同一个值

外边距 margin

相关属性值：

- margin-left 左外边距
- margin-right 右外边距
- margin-top 上外边距
- margin-bottom 下外边距
- margin 上面四个属性的简写。规律同padding

上下margin的传递，父子关系

margin-top传递：如果块级元素的顶部线和父元素的顶部线重叠，那么这个块级元素的margin-top会传递给父元素。

下面的margin-top设置是在inner上，但是传递到了box2上

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
  <title>Document</title>
  <style>
    .box1 {
      width: 100px;
      height: 100px;
      background-color: red;
    }

    .box2 {
      width: 200px;
      height: 200px;
      background-color: orange;
    }

    .inner {
      width: 100px;
      height: 100px;
      background-color: orchid;
      margin-top: 20px;
    }
  </style>
</head>

<body>
  <div class="box1"></div>
  <div class="box2">
    <div class="inner"></div>
  </div>
</body>

</html>
```

margin-bottom传递：如果块级元素的底部线和父元素的底部线重叠，并且父元素的高度是auto，么这个块级元素的margin-bottom值会传递给父元素。

下面的margin-bottom设置是在inner上，box2的高度是auto（没有设置），inner的margin-bottom递到了box2上。

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box1 {
      width: 100px;
      height: 100px;
      background-color: red;
    }

    .box2 {
      width: 200px;
```

```
    background-color: orange;
  }

  .inner {
    width: 100px;
    height: 100px;
    background-color: orchid;
    margin-bottom: 20px;
  }
 </style>
</head>

<body>
 <div class="box2">
  <div class="inner"></div>
 </div>
 <div class="box1"></div>
</body>

</html>
```

如何防止出现传递问题?

1. 给父元素设置 padding-top\padding-bottom，让父级元素和内部元素的顶部、底部线不重，就可以防止此问题。不推荐。

```
.box2 {
 width: 200px;
 background-color: orange;
 padding-bottom: 5px;
}
```

2. 给父元素设置 border，也是让元素边界线不重叠。不推荐

```
.box2 {
 border-bottom: 5px solid yellowgreen;
 width: 200px;
 background-color: orange;
}
```

3. 触发 BFC（block format context）结界。BFC是解决此类问题最好的方法。

- 浮动可以触发
- 设置一个元素的 overflow为非visible，如hidden、auto、scroll

```
.box2 {
 overflow: hidden;
 width: 200px;
 background-color: orange;
}
```

建议：

- margin一般是用来设置兄弟元素之间的间距
- padding一般是用来设置父子元素之间的间距

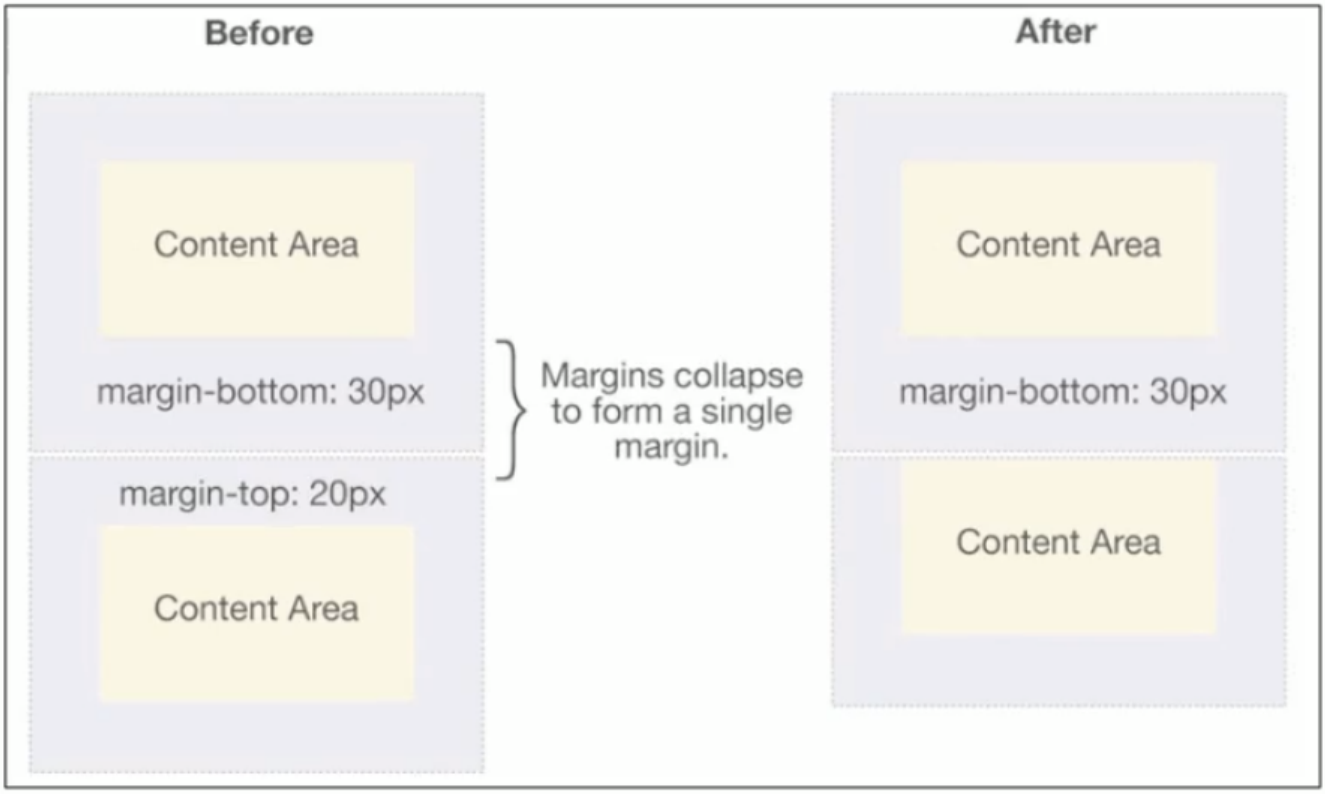
上下margin折叠

垂直方向上相邻的2个margin（margin-top、margin-bottom）有可能会合并为1个margin，这种象叫做折叠collapse。

水平方向上的margin永远不会折叠。

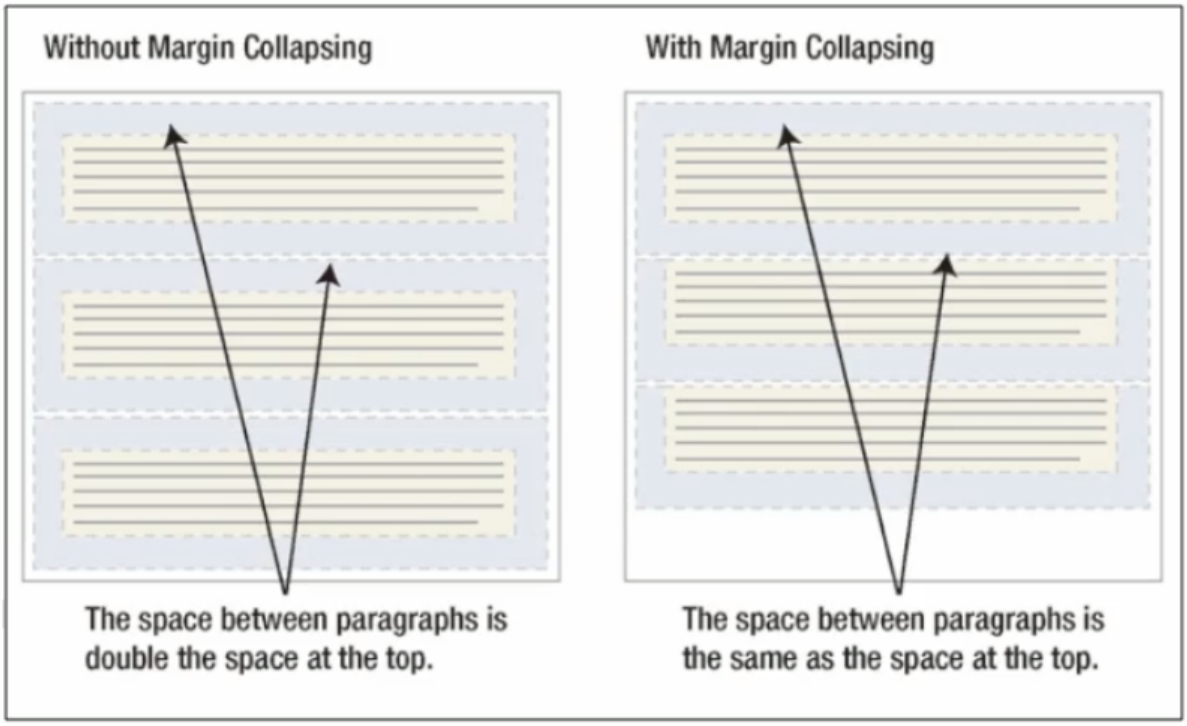
折叠后最终值的计算规则：两个值进行比较，取较大的值。

- 两个兄弟块级元素之间上下margin的折叠。



- 父子块级元素之间的margin折叠。

块级元素折叠问题看似有点莫名其妙，实际上还有有用处的。比如连续的段落之间的margin，恰好要这种折叠效果。



如何防止折叠?

- 只设置其中一个元素的 margin。

边框 border

边框宽度:

- border-top-width
- border-right-width
- border-bottom-width
- border-left-width
- border-width

边框颜色

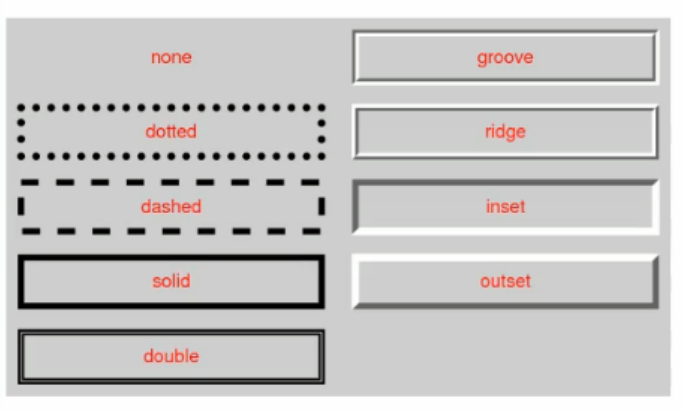
- border-top-color
- border-right-color
- border-bottom-color
- border-left-color
- border-color

边框样式

- border-top-style
- border-right-style
- border-bottom-style

- border-left-style
  - border-style

边框样式取值如图



border-top、border-right、border-bottom、border-left各个边的属性简写。

border 上面三个属性的简写，且不缺分顺序。按照个人习惯写就好。

边框 形状

边框的形状可能是：

- 矩形
- 梯形
- 三角形

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    div {
      margin-bottom: 30px;
    }

    /* 矩形 */
    .box1 {
      width: 100px;

      border-top: 20px solid red;
    }

    /* 梯形 */
    .box2 {
      width: 50px;
      height: 50px;
```

```
      background-color: blue;
      border-top: 50px solid red;
      border-right: 50px solid yellow;
      border-bottom: 50px solid green;
      border-left: 50px solid purple;
    }

    /* 三角形1 */
    .box3 {
      width: 0;
      height: 0;
      border-top: 100px solid red;
      border-left: 100px solid green;
    }

    /* 三角形2 */
    .box4 {
      width: 0;
      height: 0;
      border-top: 50px solid red;
      border-left: 50px solid green;
      border-bottom: 50px solid blue;
      border-right: 50px solid purple;
    }

    /* 三角形3 */
    .box5 {
      width: 0;
      height: 0;
      border-top: 50px solid red;
      border-left: 50px solid transparent;
      border-right: 50px solid transparent;
    }

    /* 三角形4 */
    .box6 {
      width: 0;
      height: 0;
      border-top: 100px solid red;
      border-left: 100px solid transparent;
      transform: rotate(-45deg);
    }
  </style>
</head>

<body>
  <div class="box1"></div>
  <div class="box2"></div>
  <div class="box3"></div>
  <div class="box4"></div>
  <div class="box5"></div>
  <div class="box6"></div>
</body>
</html>
```

边框 border-radius

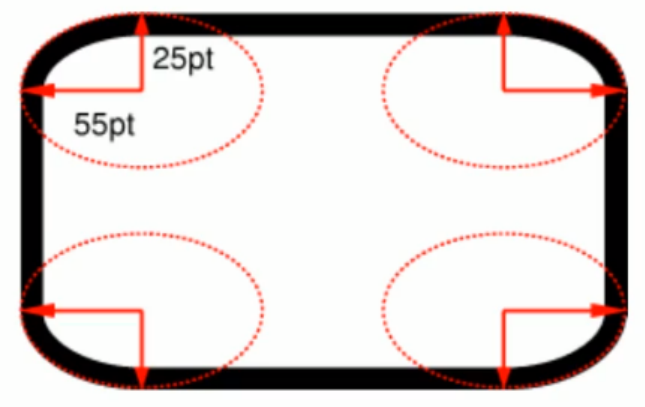
圆角可以设置具体数值，也可以设置百分比值。

圆角相关属性：

- border-top-left-radius
- border-top-right-radius
- border-bottom-left-radius
- border-bottom-right-radius
- border-radius

broder-*-*-radius定义的是四分之一椭圆的半径

- 第一个是水平半径
- 第二个是垂直半径（如果不设置，就跟随水平的半径值）
- border-top-left-radius: 55pt 25pt; 设置如图：



border-radius是一个缩写属性

- border-radius: 10px 20px 30px 40px/15px 25px 35px 45px;
- 斜线 /前面是水平半径，后面是垂直半径。
- 四个值的顺序是顺时针方向，上左 top-left、上右top-right、下右bottom-right、下左botto -left。也还是上右下左。不过很少这么用，一般只用来设置一个值。
    - 如果下左 bottom-left没设置，就跟随上右top-right
    - 如果下右 bottom-right没设置，就跟随上左top-left
    - 如果上右 top-right没设置，就跟随top-left

外轮廓 outline

outline表示元素的外轮廓，特点是

- 不占用空间
- 默认显示在 border的外面

outline相关属性有

- outline-width 宽度
- outline-style 样式，取值跟border样式一样，如solid、dotted等
- outline-color 颜色
- outline 是上面三个属性的简写。用法和border类似

应用实例：

- 去除 a元素、input元素的focus轮廓效果。

```
a, input {
  outline: none;
}
```

阴影 box-shadow

<shadow>= inset? && length{2, 4} && color?

- inset 外框阴影变内框阴影
- 第一个 length 水平方向的偏移，正数往右偏移
- 第二个 length 垂直方向的偏移，正数往下偏移
- 第三个 length 模糊半径（blur radius）
- 第四个 length 阴影向四周的延伸距离
- <color> 阴影的颜色，如果没有设置，就跟随color属性的颜色。

注：&&就表示顺序是任意的。所以这三个设置的顺序就是任意的。

阴影可以设置一个或多个，多个阴影设置使用逗号,隔开。

box-shadow: inset 10px 5px 5px orange, 5px 10px 5px green;

例子:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    div {
      margin: 20px auto;
    }
    .box1 {
      width: 100px;
      height: 100px;
      background-color: red;
      box-shadow: 10px 5px 5px 3px orange;
    }
```

```
    .box2 {
      width: 100px;
      height: 100px;
      background-color: green;
      box-shadow: inset 10px 5px 5px orange;
    }
  </style>
</head>
<body>
  <div class="box1"></div>
  <div class="box2"></div>
</body>
</html>
```

尺寸 box-sizing

用来设置盒子模型中的宽高的行为。

默认情况下，盒子尺寸是内容盒子，即设置的宽度和高度只是指定内容的高度。

border-sizing的属性值

- content-box 内容盒子。padding、border都布置在width、height外边
- border-box 盒子内减 padding、border都布置在width、height里边

水平居中

在一些需求中，需要元素在父元素中水平居中显示（父元素一般都是块级元素、inline-block）

行内元素、inline-block元素水平居中方法是：在父元素中设置text-align: center;

块级元素水平居中：在自身上设置margin: 0 auto;

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      height: 200px;
      background-color: red;
      /* 1.普通文本 */
      /* text-align: center; */
      /* 2.行内元素 */
      /* text-align: center; */
      /* 3.行内替换元素 */
      /* text-align: center; */
      /* 4.行内块级元素 */
      /* text-align: center; */
```

```
    /* 5.块级元素 这个设置只能文字居中 */
    /* text-align: center; */
  }

  .ib {
    width: 100px;
    height: 100px;
    background-color: green;
    display: inline-block;
  }

  .block {
    width: 200px;
    height: 100px;
    background-color: yellow;
    margin: 0 auto;
  }
 </style>
</head>

<body>
 <div class="box">
  <!-- 1.普通文本 -->
  <!-- 我是普通文本 -->

  <!-- 2.行内元素 -->
  <!-- <strong>我是行内元素</strong> -->

  <!-- 3.行内替换元素 -->
  <!-- <a href="#">我是a链接</a> -->

  <!-- 4.行内块级元素: inline-block -->
  <!-- <span class="ib">我是行内块级元素</span> -->

  <!-- 5.块级元素 -->
  <div class="block">我是块级元素</div>
 </div>
</body>

</html>
```

margin水平居中原理

maragin: 0 auto;中设置了margin-left和margin-right都为auto，这两个值如果不设置都有默认值0
如果只设置这两个值其中之一为auto，比如设置margin-left: auto;，此时margin-right是0，浏览器把当前行剩余的宽度自动分配给margin-left。

下面的设置，div.inner就会靠右侧显示。

```
<style>
 .box {
  height: 200px;
  background-color: red;
```

```
 }

 .inner {
   width: 200px;
   height: 100px;
   background-color: yellow;
   margin-left: auto;
   margin-right: 0;
 }
</style>

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

那么，如果把margin-left和margin-right都设置为auto，浏览器就会把剩余的空间让两者均分。

垂直方向如果也设置auto，是不会居中的，如果想要垂直方向的居中，意味着父元素垂直高度必须是auto。但如果设置父元素高度是auto，父元素的高度就必须由其内部的元素撑起，那么它的高度就等于中子元素的高度，也就不存在垂直居中了。如下：

```
<style>
 .box {
   height: auto;
   background-color: red;
 }


 .inner {
   width: 200px;
   height: 100px;
   background-color: yellow;
   margin-left: auto;
   margin-right: auto;
 }
</style>

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

CSS属性-背景

背景 background-image

用于设置元素的背景图片。

- 会**盖在（不是覆盖）**background-color的上面。
- 如果设置了多张图片 background-image: url("bg001.png"), url("bg002.png"), url("bg003.

ng");，设置的第一张图片将显示在最上面，其他图片按顺序层叠在下面。

- 如果设置了背景图片后，元素没有具体的宽高，背景图片是不会显示出来的。即背景图不会把子撑起来。

例子：

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      width: 800px;
      height: 600px;
      background-color: red;
      background-image: url('https://lh3.googleusercontent.com/proxy/JxeDPDa1PJJA55FcVdrK12W-7gHVoUS3YsSIfg1kl7P_ZNO6cW537L5FYeJoACiIXFMRPHLnPTEuiJ20XHfglwstM80py-Nd4ZELtl67KN_EdHr_rPWO3eX7DZwd1K'), url('https://pic1.zhimg.com/v2-3b4fc7e3a1195a0810259246c38debc_720w.jpg?source=172ae18b');
      background-repeat: repeat-y;
    }

  </style>
</head>

<body>
  <div class="box">
    <div class="inner"></div>
  </div>
</body>

</html>
```

背景 background-repeat

用于设置背景图片是否需要平铺。

常见设置有：

- repeat 平铺，默认值
- no-repeat 不平铺
- repeat-x 只在水平方向平铺
- repeat-y 只在垂直方向平铺

背景 background-size

用于设置背景图片的大小。

- auto 以背景图本身大小显示 默认值
  - cover 缩放背景图，以完全覆盖铺满元素
  - contain 缩放背景图，宽度或高度铺满元素，但是图片保持宽高比
  - <percentage> 百分比，相对于背景区（background positioning area）。当只设置一个值，是设置说水平方向的大小，垂直方向是auto，即使用默认值。设置两个值时，是水平、垂直方向。ackground-size: auto 180px;表示宽度保持原来宽高比自动计算，高度180px。background-size: 10px等价于background-size: 150px auto
  - length 具体的大小，比如100px 可以设置一个或两个值。规则同上。

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      width: 1200px;
      height: 500px;
      background-color: red;
      background-image: url('https://pic1.zhimg.com/v2-3b4fc7e3a1195a081d0259246c38debc
720w.jpg?source=172ae18b');
      background-repeat: no-repeat;
      /* 背景图片进行拉伸，让背景图片覆盖整个元素 */
      /* background-size: cover; */
      /* 对背景进行拉伸，拉升到一个方向的宽度（高度），不再进行拉伸，保持图片的宽高比 */
      background-size: contain;

      /* 设置百分比或具体值 */
      /* background-size: 33% 80%; */
      /* background-size: 300px 100px; */
    }

  </style>
</head>

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

</html>
```

背景 background-position

background-position用于设置背景图片在水平、垂直方向上的具体位置。可以设置一个或两个值

- <number> 设置具体值 水平或垂直方向
- <percentage> 设置百分比 水平或垂直方向
- 水平方向还可以设置值 left、center、bottom

- 垂直方向还可以设置 top、center、bottom
  - 如果只设置了一个方向，另一个方向默认是 center，如background-position: 80px;等价于background-position: 80px center;

背景 background-attachment

可以设置以下3个值：

- scroll 背景图片跟随元素一起滚动 默认值
- local 背景图片跟随元素以及元素内容一起滚动
- fixed 背景图片相对于浏览器窗口固定

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      width: 200px;
      height: 300px;
      background-image: url('http://res.xyq.netease.com/pc/zt/20151119153357/images/top_ac1ab4.jpg');
      overflow: auto;
      /* 随着box的滚动（浏览器），背景一起滚动 */
      /* background-attachment: scroll; */

      /* local 图片会随着box内容的滚动而滚动 */
      /* background-attachment: local; */

      /* 背景是固定的，不会随着box的滚动而滚动 游戏网站使用较多 */
      background-attachment: fixed;
    }
  </style>
</head>
<body>
  <div class="box">
    MongoDB是一个基于分布式文件存储 [1] 的数据库。由C++语言编写。旨在为WEB应用提供可扩的高性能数据存储解决方案。
    MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，像关系数据库的。它支持的数据结构非常松散，是类似json的bson格式，因此可以存储比较复杂的数类型。Mongo最大的特点是它支持的查询语言非常强大，其语法有点类似于面向对象的查询语言，乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。
  </div>
  <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><b
```

>

</body>
</html>

背景 background

background是一系列背景相关属性的简写属性

- 它是这些属性的缩写： image position/size repeat attachment color
- background-size可以省略，如果不省略，/background-size必须紧跟在background-position后面
- 其他属性也都可以省略，而且顺序任意

background: url("images/beer.png") center center/150px 150px no-repeat #fff;

background-image和img的选择

	img	background-image
性质	HTML元素	CSS样式
图片是否占用空间	√	X
浏览器右键直接查看地址	√	X
支持CSS Sprite	X	√
更有可能被搜索引擎收录	√（结合alt属性）	X
加载顺序	优先加载	等加载完HTML元素后再加载

总结：

- img，作为网页内容的重要组成部分，比如广告图片、LOGO图片、文章配图、产品图片
- background-image 可有可无。有，能让页面更加美观。无，也不影响用户获取完整的网页内信息

光标 cursor

可视设置鼠标指针在元素上面时的显示样式。

cursor常见的设置:

- auto 浏览器根据上下文决定指针的显示样式，比如根据文本和非文本切换指针样式
- default 由操作系统决定，一般就是一个小箭头
- pointer 一只小手，鼠标指针挪动到链接上面默认就是这个样式
- text 一条竖线，鼠标指针挪动到文本输入框上面默认就是这个样式
- none 没有任何指针显示在元素上面

CSS属性-定位

这个文章写的很不错。

利用position可以对元素进行定位，常用取值有：

- static 静态定位 默认值
- relative 相对定位
- absolute 绝对定位
- fixed 固定定位

	脱离标准流	定位元素	绝对定位元素	位参照对象
static	X	X	X	X
relative	X	==√==	X	素自己原来的位置
absolute	==√==	==√==	==√==	邻近的定位祖先元素 如果找不到这样的祖先元素，参照对象是视口
fixed	==√==	==√==	==√==	口

静态定位 static

默认值 元素按照标准流布局，left、right、top、bottom没有任何作用

相对定位 relative

元素按照标准流布局。

可以通过left、right、top、bottom进行定位。定位参照对象是原来的位置。

应用场景：

- 在不影响其他元素位置的前提下，对当前元素位置进行微调。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    sub, sup {
      font-size: 14px;
    }

    sub {
      position: relative;
      bottom: 5px;
    }

    sup {
```

```
      position: relative;
      top: 2px;
    }
  </style>
</head>
<body>
  <h1>请计算n<sub>1</sub>+n<sub>2</sub>+n<sup>2</sup>的值</h1>
</body>
</html>
```

梦幻西游相对定位练习

背景图根据浏览器窗口大小，决定显示内容大小，图片内容始终居中显示。这种方式比设置background-image的方式更好，后者必须有div的固定高度，没有高度的情况下是看不到背景的。使用img更合一点，可以直接撑起高度。

实现思路：图片向左移动的距离= 图片宽度 * 0.5 - div * 0.5。即向左移动的宽度等于图片宽度的一减去div容器的宽度的一半。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    body {
      margin: 0;
    }

    .box {
      overflow: hidden;
    }

    .box img {
      /* 1 向左移动img的一半 */
      position: relative;
      /* left: -960px; */
      transform: translate(-50%);
      /* 2 向右移动父元素的一半 */
      margin-left: 50%;
    }
  </style>
</head>
<body>
  <div class="box">
    <img src="http://res.xyq.netease.com/pc/zt/20151119153357/images/top_aec1ab4.jpg" al
="">
  </div>
</body>
</html>
```

固定定位 fixed

元素脱离标准流。

可以通过left、right、top、bottom进行定位。定位参照对象是视口（viewport）。

当画布滚动时，固定不动。

什么是视口?

能看到网页的部分区域叫做视口。

什么是画布?

能看到网页的部分区域是有限的，但是网页可能会很大，整个网页是画布。

绝对定位 absolute

元素脱离normal flow(标准流)

可以通过left right top bottom进行定位

- 定位参照对象时最邻近的定位祖先元素
- 如果找不到这样的祖先元素,参照对象是视口

定位元素指的是

- position值不为 static的元素
- 也就是position值为 ralative absolute fixed的元素

子绝父相

在觉大数情况下，子元素的绝对定位都是相对于父元素进行定位

如果希望子元素相对于父元素进行定位，又不希望父元素脱标,常用解决方案是:

- 父元素设置 position: relative，让父元素成为定位元素,而且父元素不脱离标准流
- 子元素设置 position: absolute

简称为子绝父相。

绝对定位技巧

绝对定位元素是position值为absolute或fiexed的元素

对于绝对定位元素来说

- 定位参照对象的宽度 = left + right + margin-left + margin-right + 绝对定位元素的实际占用度
- 定位参照对象的高度 = top + bottom + margin-top + margin-bottom + 绝对定位元素的实占用高度

如果希望绝对定位元素的宽度和定位参照对象一样，可以给绝对定位元素设置以下属性：left: 0; right:

0; bottom: 0; margin: 0。

如果希望绝对定位元素在定位参照对象中居中显示，可以给绝对定位元素设置以下属性：left: 0; right: 0; bottom: 0; margin: auto。

下面是具体实现的例子：

让子元素占据父元素，下面的div.inner没有设置宽度，当时利用上面的宽度公式，它的宽度就是div.box的宽度。高度也可以用此种方法实现

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      position: relative;
      width: 600px;
      height: 600px;
      background-color: red;
    }

    .inner {
      position: absolute;
      left: 0;
      right: 0;
      height: 100px;
      background: blue;
    }
  </style>
</head>
<body>
  <div class="box">
    <div class="inner"></div>
  </div>
</body>
</html>
```

让内容居中，根据上面的公式，首先设置left、right相等，一般设置为0，然后设置margin: 0 auto此时div.inner就会水平居中。同理，垂直方向上也可以这么设置。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      position: relative;
      width: 600px;
      height: 600px;
```

```
    background-color: red;
  }

  .inner {
    position: absolute;
    left: 0;
    right: 0;
    top: 0;
    bottom: 0;
    margin: auto auto;
    width: 200px;
    height: 200px;
    background: blue;
  }
 </style>
</head>
<body>
 <div class="box">
  <div class="inner"></div>
 </div>
</body>
</html>
```

定位元素的层叠关系

定位元素的层叠关系：

- 父子关系：子元素会层叠在父元素上
- 非父子关系：
  - 都是非定位元素：在标准流中一般存在的层叠想想
  - 一个是定位、一个是非定位元素：定位元素会层叠在非定位元素上面
  - 都是定位元素：默认后面出现的会盖住前面的。可以使用CSS属性 z-index来控制层叠顺序。

CSS属性-z-index

z-index属性用来设置**定位元素**的层叠关系，仅对定位元素有效。

取值可以是正整数、负整数、0、auto（默认）

比较原则：

- 如果是兄弟关系：
  - z-index越大，层叠在越上面
  - z-index相等，写在后面的那个元素层叠在上面
- 如果不是兄弟关系
  - 各自从元素自己以及祖先元素中，找出最相邻的2个定位元素进行比较。

脱标元素的特点

脱离标准流元素：相对定位、绝对定位、固定定位、浮动。

脱离标准流元素的特点：

- 没有设置宽高情况下，宽高默认由内容决定
- 可以随意设置宽高，即使是行内元素
- 不再受标准流约束
- 不再给父元素汇报高度。意思是脱离了文档流，父元素不知道这个元素的存在了。

脱标元素和display的关系：

脱标元素都会变成块级元素，块级元素会占据父级元素一整行，但是脱标后已经没有父元素了，这时高都会变为auto，即包裹内容，内容宽高是多少，元素的宽高就是多少。

CSS Sprite

CSS Sprite是一种CSS图像合成技术，将各种小图片合并到一张图片上，然后利用CSS的背景定位来示对应的图片部分。

有人翻译为CSS雪碧图、CSS精灵图。

使用CSS Sprite的好处：

- 减少网页的http请求数量，加快网页的响应速度，减轻服务器压力
- 减小图片总大小
- 解决了图片命名困扰，只需要针对一张集合的图片命名

Sprite图片制作（雪碧图、精灵图）：

- Photoshop
- 网页生成　https://www.toptal.com/developers/css/sprite-generator

前端一些框架 mpvue(vue) uniapp(vue) taro(react)

下面的例子，网页布局中一些居中显示，谁需要居中，class中就加上wrap

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .wrap {
      width: 1000px;
      margin: 0 auto;
    }

    .nav {
```

```
    margin-top: 20px;
    background-color: red;
    height: 50px;
  }

  .header {
    margin-top: 20px;
    background-color: green;
    height: 50px;
  }

  .content {
    margin-top: 20px;
    background-color: blue;
    height: 50px;
  }
 </style>
</head>
<body>
 <!-- 居中显示，谁需要居中，就加上wrap -->
 <div class="nav wrap"></div>
 <div class="header wrap"></div>
 <div class="content"></div>
</body>
</html>
```

CSS Sprite简单练习

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Document</title>
 <style>
  .wrap {
    width: 1100px;
    margin: 0 auto;
  }

  ul {
    list-style: none;
  }

  p, ul, li, h5 {
    margin: 0;
    padding: 0;
    float: left;
  }

  .service ul li {
    margin-left: 12px;
    margin-right: 20px;
```

```
    }

    .service ul li p {
      height: 42px;
      line-height: 42px;
      font-size: 18px;
      width: 180px;
      font-weight: 700;
      margin-left: 10px;
    }

    .service ul li h5 {
      text-indent: -9990px;
      width: 36px;
      height: 42px;
      background-image: url(http://misc.360buyimg.com/mtd/pc/index_2019/1.0.0/assets/img/
3f3ddf914b1b527d0429a3d713cfe3a.png);

    }

    .service ul li .duo {
      background-position: 0 -192px;
    }

    .service ul li .kuai {
      background-position: -41px -192px;
    }

    .service ul li .hao {
      background-position: -82px -192px;
    }

    .service ul li .sheng {
      background-position: -123px -192px;
    }
  </style>
</head>
<body>
  <div class="service wrap">
    <ul>
      <li>
        <h5 class="duo">多</h5>
        <p>品类齐全，轻松购物</p>
      </li>
      <li>
        <h5 class="kuai">快</h5>
        <p>多仓直发，急速配送</p>
      </li>
      <li>
        <h5 class="hao">好</h5>
        <p>正品行货，精致服务</p>
      </li>
      <li>
        <h5 class="sheng">省</h5>
```

```
      <p>天天低价，畅选无忧</p>
    </li>
   </ul>
  </div>
</body>
</html>
```

背景居中，梦幻西游适配练习：

能让产品适应各种运行环境，尽量保持一致的用户体验在前端开发中，一般都是要做浏览器适配、屏适配。

背景图根据浏览器窗口大小，决定显示内容大小，图片内容始终居中显示。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    body {
      margin: 0;
      padding: 0;
    }

    .box {
      height: 400px;
      background-image: url('http://res.xyq.netease.com/pc/zt/20151119153357/images/top_ac1ab4.jpg');
      background-position: center -81px;
    }
  </style>
</head>
<body>
  <div class="box"></div>
</body>
</html>
```

CSS属性-浮动

绝对定位、浮动都会让元素脱离标准流，以达到灵活布局的效果。

可以通过float属性让元素产生浮动效果，float的常用取值：

- none 不浮动，默认值
- left 向左浮动
- right 向右浮动

元素的层叠关系：

- 标准元素：标准流中的元素不存在层叠
- 定位元素：定位元素层叠到标准流元素上面

- 定位元素之间可以使用z-index改变层叠关系
  - 浮动元素：定位元素会层叠在浮动元素上面。
  - 层叠最终关系：定位元素 > 浮动元素 > 标准元素。

浮动的规则

- 规则一：元素一旦浮动后，会脱离标准流，朝着向左或向右方向移动，直到自己的边接紧贴着含块（一般是父元素）或者其他浮动元素的边界为止
- 规则二：浮动元素不能与行内级元素层叠，行内级内容将会被浮动元素推出。也就是都会被挤去。
  - 比如行内级元素、inline-block元素、块级元素的文字内容。可以看到下面的其他内容都会被trong元素推出去，但并意味着strong没有脱离标准流。只是规则使得它不能与这些元素层叠。因为开始出现浮动就是为了做图文环绕的。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    strong {
      float: left;
    }

    .inner {
      display: inline-block;
      width: 50px;
      height: 50px;
      background: blue;
    }

    a {
      background-color: #00f;
      color: #fff;
    }
  </style>
</head>
<body>
 <div class="box">
   div元素的文字
  <div class="inner"></div>
  <span>span元素</span>
  <strong>strong元素</strong>
  <a href="#">a元素</a>
 </div>
</body>
</html>
```

  - 利用此特性可以轻松实现图文环绕。

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      background-color: red;
      width: 500px;
    }

    img {
      float: left;
    }
  </style>
</head>
<body>
  <div class="box">
    据俄罗斯防疫指挥部4日消息
    <img
  src="https://ss0.bdstatic.com/94oJfD_bAAcT8t7mm9GUKT-xh_/timg?image&quality=100&size=b4000_4000&sec=1607166166&di=6f23129fe4375e54a9d8146e148b55cb&src=http://a0.at.hudong.com/30/29/01300000201438121627296084016.jpg"
      alt="">
    过去24小时俄新增新冠肺炎确诊病例27403例，累计确诊达2402949例，累计死亡42176例。莫斯市5日开始大规模新冠疫苗接种。随着新冠疫情在俄罗斯不断蔓延，俄罗斯首都莫斯科市5日开始大规疫苗接种。莫斯科市政府市民疫苗接种电子申请系统4日正式启动，接种站5日投入运营。
  </div>
</body>
</html>
```

- 规则三：行内级元素、inline-block元素浮动后，其顶部将与所在行的顶部对齐。只会在当前行右浮动，并不会向上浮动。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      background-color: red;
      width: 300px;
    }

    .inner {
      float: left; /* 这里会报错 */
      display: inline-block;
      width: 30px;
      height: 30px;
      background: blue;
    }
  </style>
```

```
</head>
<body>
  <div class="box">
    据俄罗斯防疫指挥部4日消息
    <div class="inner" ></div>
    过去24小时俄新增新冠肺炎确诊病例27403例，累计确诊达2402949例，累计死亡42176例。莫斯市5日开始大规模新冠疫苗接种。
    <div class="inner"></div>
    随着新冠疫情在俄罗斯不断蔓延，俄罗斯首都莫斯科市5日开始大规模疫苗接种。莫斯科市 <div class="inner"></div>政府市民疫苗接种电子申请系统4日正式启动，接种站5日投入运营。
  </div>
</body>
</html>
```

- 规则四：如果元素是向左（右）浮动，浮动元素的左（右）边接不能超出包含块的左（右）边。
- 规则五：浮动元素之间不能层叠。
  - 左浮找左浮，右浮找右浮。
  - 如果水平方向剩余的空间不够显示浮动元素，浮动元素将线下移动，直到找到充足的空间为;
- 规则六：浮动元素的顶端不能超过包含块的顶端，也不能超过之前所有浮动元素的顶端。

下面的div.inner3左边会挨着div.inner2，并不会挨着div.inner1。如果div.inner2的浮动改成float: right;也不会到最顶部。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      background-color: red;
      width: 300px;
      height: 320px;
    }

    .inner1 {
      float: left;
      width: 200px;
      height: 100px;
      background: green;
    }

    .inner2 {
      float: left;
      width: 150px;
      height: 150px;
      background: yellow;
    }

    .inner3 {
```

```
      float: left;
      width: 30px;
      height: 150px;
      background: purple;
    }
  </style>
</head>
<body>
  <div class="box">
    <div class="inner1"></div>
    <div class="inner2"></div>
    <div class="inner3"></div>
  </div>
</body>

</html>
```

理解浮动规则

- div.inner2左浮动，只会在它自己的行内浮动；
- div.inner1左浮动，div.inner2会跑到最上面，因为div.inner1脱离了文档流，此时会出现inner1和nner2层叠；此时inner2中的文本，会环绕inner1显示。
- div.inner1左浮动，设置div.inner2的display: inline-block;，div.inner2会跑到最上面且会被挤右侧，并不会层叠。
- div.inner1和div.inner2都左浮动，会挨着第一行显示，不会层叠；
- 如果此时去掉 div.box中的高度，此时，就没有了红色。脱离文档流的元素不会向父元素报告度，父元素没有了高度，即高度坍塌。可以清除浮动来解决。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      background-color: red;
      height: 200px;
    }

    .inner1 {
      /* float: left; */
      width: 50px;
      height: 50px;
      background: green;
    }

    .inner2 {
      /* float: left; */
      /* display: inline-block; */
      width: 80px;
```

```
      height: 80px;
      background: yellow;
    }
  </style>
</head>
<body>
  <div class="box">
    <div class="inner1"></div>
    <div class="inner2">我是内容啊</div>
  </div>
</body>
</html>
```

多出margin的处理方式

在进行布局时，同一行多个元素摆放后，设置margin-right让他们之间产生间距，那么最后一个元素置的margin-right总是多余的。

解决方法：

1. 每一行最后一个元素，总添加一个class，通过类选择器去除这个 margin-right。
2. 使用伪类选择器。但是使用伪类不好的有点是，如果布局改变，可能就要改变 nth-child。最的问题是兼容性问题。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .box {
      background-color: red;
      width: 430px;
      height: 320px;
    }

    .inner {
      float: left;
      width: 100px;
      height: 100px;
      background: green;
      margin-right: 10px;
      margin-bottom: 10px;
    }

    .inner:nth-child(4n) {
      margin-right: 0;
    }
  </style>
</head>
<body>
  <div class="box">
```

```
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
    <div class="inner"></div>
  </div>
</body>
</html>
```

3. 块级元素宽度计算。使用margin负值。

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    /* width of block = content-width + marfin-left + margin-right + padding-left + padding+r
ght + border-left + border-right */

    .box {
      background-color: red;
      width: 430px;
      height: 320px;
    }

    .wrap {
      margin-right: -10px;
    }

    .inner {
      float: left;
      width: 100px;
      height: 100px;
      background: green;
      margin-right: 10px;
      margin-bottom: 10px;
    }

  </style>
</head>

<body>
  <div class="box">
    <div class="wrap">
      <div class="inner"></div>
```

```
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
      <div class="inner"></div>
    </div>
  </div>
</body>

</html>
```

例子2:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    .container {
      width: 990px;
      height: 500px;
      background-color: red;
      margin: 0 auto;
    }

    .wrap {
      margin-right: -10px;
    }

    .item {
      float: left;
      margin-right: 10px;
      width: 240px;
      background-color: green;
    }

    .itema {
      height: 306px
    }

    .itemb {
      height: 148px;
    }

    .item-last {
```

```
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="wrap">
      <div class="item itema"></div>
      <div class="item itema"></div>
      <div class="item itemb"></div>
      <div class="item itemb"></div>
      <div class="item itemb item-last"></div>
      <div class="item itemb item-last"></div>
    </div>
  </div>
</body>
</html>
```

例子3：

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <style>
    ul {
      list-style: none;
      padding: 0;
      margin: 0;
    }

    .brand {
      width: 1100px;
      height: 180px;
      background: red;
    }

    .brand ul {
      margin-right: -10px;
    }

    .brand ul li {
      float: left;
      width: 219px;
      height: 167px;
      background: green;
      margin-right: -1px;
      border: 1px solid #333;
    }
  </style>
</head>
```

```
<body>
  <div class="brand">
    <ul>
      <li></li>
      <li></li>
      <li></li>
      <li></li>
      <li></li>
    </ul>
  </div>
</body>

</html>
```

CSS特性

继承

CSS中有些属性是可以继承的。一个元素如果没有设置某属性，就会跟随父元素的值。当然，一个元如果有设置某属性的值，就使用自己设置的值。

不能继承的属性，一般可以使用inherit值强制继承。

浏览器的开发者工具也会标识出哪些样式是继承过来的Inherited from ***。

要注意的是，CSS继承过来的是计算值，

```
<style>
  .box1 {
    font-size: 60px;
  }

  .box2 {
     /* 30px */
    font-size: 0.5em;
  }
</style>

<div class="box1">
  <div class="box2">
      <!-- p中文字30px -->
    <p>文字内容</p>
  </div>
</div>
```

层叠和权重

CSS允许相同名字的CSS属性层叠在同一个元素上。

层叠后的结果是只有一个CSS属性会生效。

哪个CSS属性会生效，取决于CSS属性所处环境的优先级高低。

浏览器的开发者工具非常清晰地显示了哪个CSS属性会生效。

基本层叠，使用了相同的选择器，后面一定会把前面的层叠掉。

下面文字内容是橘色。

```
<style>
 .box1 {
   color: red;
 }

 .box2 {
   background-color: green;
   color: purple;
 }

 .box3 {
   width: 300px;
   color: orange;
 }
</style>
<body>
 <div class="box1 box2 box3">文字内容1</div>
</body>
```

当选择器不同时，需要按照选择器的权重来层叠，谁的权重越大，谁就优先显示。

下面文字内容是蓝色。

```
<style>
#mian {
  color: blue;
}

.box {
  color: green;
}

div {
  color: red;
}

</style>
<body>
 <div class="box" id="main">文字内容</div>
</body>
```

CSS属性的优先级

按照经验，为了方便比较CSS属性的优先级，可以给CSS属性所处的环境定义一个权重。

- !important：
- 内联样式：1000

- id选择器：100
  - 类选择器、属性选择器、伪类：10
  - 元素选择器、伪元素：1
  - 通配符：0

比较优先级的严谨方法：

- 从权值最大的开始比较每一种权值的数量多少，数量多的则优先级高，即可结束比较
- 如果数量相同，比较下一个较小的权值，以此类推。
- 如果所有权值比较完毕后，发现数量相同，就采取就近原则。

```
/* 如果这2个color是作用在同一个标签上，哪个优先级高？ */

/* 2个id选择器、2个类选择器 */
.five#radio .one #three {
  color: blue;
}

/* 2个id选择器、1个类选择器、2个元素选择器 */
#box #btn .four div span {
  color: black;
}
```

要理解的是，权重只是为了方便记忆，并不是真的按照值去比较权重。

为什么"我是个a"是蓝色。

```
<style>
 div {
   color: red !important;
 }
</style>

<body>
  <div class="box1 box2 box3">
    我是div的内容 <br>
    <span>我是一个span</span> <br>
    <a href="">我是个a</a>
  </div>
</body>
```

因为a标签本身给自己设置了颜色，所以会使用自己的颜色，而不是继承div中的红色，即使设置了!important

为什么div中的文本没有变成红色

```
<style>
 p div {
   color: red !important;
 }
</style>
```

```
<body>
 <p>
  <div>我是一个div</div>
 </p>
</body>
```

浏览器本身不支持p中嵌套div，可以在浏览器中查看，html结构是乱掉的，和声明的不一样。

CSS属性的使用经验

为什么有时候编写的CSS属性不生效，有可能是因为：

- 选择器的优先级太低
- 选择器没选中对应的元素
- CSS属性的使用形式不对
    - 元素不支持此CSS属性，比如span默认是不支持width和heigt的
    - 浏览器不支持此CSS属性，如旧版本的浏览器不支持CSS3的某些属性
    - 被同类的CSS属性覆盖，比如font覆盖font-size

建议：

充分利用浏览器开发者工具进行调试（增加、修改样式）、查错。