- CSS样式
  - 内联式css样式，直接写在现有的HTML标签中
  - 嵌入式css样式，写在当前的文件中
  - 外部式css样式，写在单独的一个文件中
  - 三种方法的优先级
- CSS选择器
  - 标签选择器
  - 类选择器
  - ID选择器
  - 类和ID选择器的区别
  - 子选择器
  - 包含(后代)选择器
  - 通用选择器
  - 伪类选择符
  - 分组选择符
- CSS的继承、层叠和特殊性
  - 继承
  - 特殊性
  - 层叠
  - 重要性
- CSS格式化排版
  - 文字排版--字体 `font-family`
  - 文字排版--字号、颜色 `font-size: xpx, color`
  - 文字排版--粗体 `font-weight:bold`
  - 文字排版--斜体 `font-style:italic`
  - 文字排版--下划线 `text-decoration:underline`
  - 文字排版--删除线 `text-decoration:line-through`
  - 段落排版--缩进 `text-indent:xem`
  - 段落排版--行间距（行高） `line-height:xem`
  - 段落排版--中文字间距、字母间距 `letter-spacing:xpx` `word-spacing:xpx`
  - 段落排版--对齐 `text-align:center/left/right`
  - 颜色值
    - 英文命令颜色
    - RGB颜色
    - 十六进制颜色
  - 颜色缩写
  - 字体缩写
  - 长度值
    - 像素
    - em
    - 百分比
- CSS盒模型
  - 元素分类--块级元素
  - 元素分类--内联元素
  - 元素分类--内联块级元素
  - 盒模型--边框(border)
  - 盒模型--宽度和高度(width and height)
  - 盒模型--填充(padding)
  - 盒模型--边界(margin)
  - 盒模型代码简写

CSS样式

CSS全称为“层叠样式表 (Cascading Style Sheets)”，它主要是用于定义HTML内容在浏览器内的显示样式，如文字大小、颜色、字体加粗等。

css 样式由选择符和声明组成，而声明又由属性和值组成。

选择符：又称选择器，指明网页中要应用样式规则的元素，如本例中是网页中所有的段（p）的文字将变成蓝色，而其他的元素（如ol）不会受到影响。

声明：在英文大括号 `{}` 中的的就是声明，属性和值之间用英文冒号 `:` 分隔。当有多条声明时，中间可以英文分号 `;` 分隔，如下所示：

```
p{ /*选择符 选择器”指明了{}中的“样式”的作用对象，也就是“样式”作用于网页中的哪些元素。*/
  font-size: 12px; { /*属性: 值;*/
  color:red;

}
```

注意:

1. 最后一条声明可以没有分号，**但是为了以后修改方便，一般也加上分号。**
2. 为了使用样式更加容易阅读，可以将每条代码写在一个新行内

CSS样式可以写在哪些地方呢？从CSS 样式代码插入的形式来看基本可以分为以下3种：**内联式、嵌入式和外部式**三种。

内联式css样式，直接写在现有的HTML标签中

内联式css样式表就是把css代码直接写在现有的HTML标签中，如下面代码：

```
<p style="color:red">这里文字是红色。</p>
```

注意要写在元素的**开始**标签里，下面这种写法是错误的：

`<p>这里文字是红色。</p style="color:red">`

并且css样式代码要写在 `style=""` 双引号中，如果有多条css样式代码设置可以写在一起，中间用分号隔开。如下代码：

```
<p style="color:red;font-size:12px">这里文字是红色。</p>
```

嵌入式css样式，写在当前的文件中

嵌入式css样式，就是可以把css样式代码写在 `<style type="text/css"></style>` 标签之间。嵌入式css样式必须写在 `<style></style>` 之间，并且一般情况下嵌入式css样式写在 `<head></head>` 之间。

```
<style type="text/css">
span{
color:red;
}
</style>
```

外部式css样式，写在单独的一个文件中

外部式css样式(也可称为外联式)就是把css代码写一个单独的外部文件中，这个css样式文件以 `.css` 为扩展名，在 `<head>` 内（不是在 `<style>` 标签内）使用 `<link>` 标签将css样式文件链接到HTML文件内，如下面代码：

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>嵌入式css样式</title>
<link href="style.css" rel="stylesheet" type="text/css" />
</head>
```

注意：

1. css样式文件名称以有意义的英文字母命名，如 `main.css` 。
2. `rel="stylesheet" type="text/css"` 是固定写法不可修改。
3. `<link>` 标签位置一般写在 `<head>` 标签之内。

三种方法的优先级

这三种样式是有优先级的，记住他们的优先级：**内联式 > 嵌入式 > 外部式**

但是嵌入式>外部式有一个前提：嵌入式css样式的位置一定在外部式的后面。如右代码编辑器就是这样，`<link href="style.css" ...>` 代码在 `<style type="text/css">...</style>` 代码的前面（实际开发中也是这么写的）。

```
<link href="style.css" rel="stylesheet" type="text/css">
<style type="text/css">
span{
   color:red;
}
</style>
```

其实总结来说，就是--就近原则（离被设置元素越近优先级别越高）。

但注意上面所总结的优先级是有一个前提：内联式、嵌入式、外部式样式表中css样式是在的**相同权值**的情况下

CSS选择器

标签选择器

标签选择器其实就是html代码中的标签。如右侧代码编辑器中的 `<html>、<body>、<h1>、<p>、<img>`

```
<style type="text/css">
h1{
    font-weight:normal;
    color:red;
}
</style>
```

类选择器

类选择器在css样式编码中是最常用到的： `.类选器名称{css样式代码;}`

注意：

1. 英文圆点开头
2. 其中类选器名称可以任意起名（但不要起中文噢）

使用方法：

1. 使用合适的标签把要修饰的内容标记起来： `<span>胆小如鼠</span>`
2. 使用class="类选择器名称"为标签设置一个类： `<span class="stress">胆小如鼠</span>`
3. 设置类选器css样式： `.stress{color:red;}/*类前面要加入一个英文圆点*/`

ID选择器

在很多方面，ID选择器都类似于类选择符，但也有一些重要的区别：

1. 为标签设置 `id="ID名称"` ，而不是 `class="类名称"` 。
2. ID选择符的前面是井号 `#` 号，而不是英文圆点 `.` 。

类和ID选择器的区别

相同点：可以应用于任何元素 不同点：

1. ID选择器只能在文档中使用一次。**与类选择器不同，在一个HTML文档中，ID选择器只能使用一次，而且仅一次。而类选择器可以使用多次。** 下面代码是正确的：

   ```
   <p>三年级时，我还是一个<span class="stress">胆小如鼠</span>的小女孩，上课从来不敢回答老师提出的问题，生怕回答错了老师会批评我。就一直没有这个<span class="stress">勇气</span>来回答老师提出的问题。</p>
   ```

   而下面代码是错误的：

```
<p>三年级时，我还是一个<span id="stress">胆小如鼠</span>的小女孩，上课从来不敢回答老师提出的问题，生怕回答错了老师会批评我。就一直没有这个<span id="stress">勇气</span>来回答老师提出的问题。</p>
```

2. 可以使用类选择器词列表方法为一个元素同时设置多个样式。**我们可以为一个元素同时设多个样式，但只可以用类选择器的方法实现，ID选择器是不可以的（不能使用 ID 词列表）。** 下面的代码是正确的

```
.stress{
    color:red;
}
.bigsize{
    font-size:25px;
}
<p>到了<span class="stress bigsize">三年级</span>下学期时，我们班上了一节公开课...</p>
```

上面代码的作用是为“三年级”三个文字设置文本颜色为红色并且字号为25px。

下面的代码是错误的

```
#stressid{
    color:red;
}
#bigsizeid{
    font-size:25px;
}
<p>到了<span id="stressid bigsizeid">三年级</span>下学期时，我们班上了一节公开课...</p>
```

上面代码不可以实现为“三年级”三个文字设置文本颜色为红色并且字号为25px的作用。

我的理解：类选择器更加全面通用，可以用超过一次，也可以为同一个元素同时设置多个样式

子选择器

还有一个比较有用的选择器子选择器，即大于符号(>),用于选择指定标签元素的第一代子元素。

```
.food>li{border:1px solid red;}
```

这行代码会使class名为food下的子元素li（水果、蔬菜）加入红色实线边框。

包含(后代)选择器

即加入空格,用于选择指定标签元素下的后辈元素。

```
.first  span{color:red;}
```

这行代码会使第一段文字内容中的“胆小如鼠”字体颜色变为红色。

请注意这个选择器与子选择器的区别，**子选择器（child selector）仅是指它的直接后代，或者你可以理解为作用于子元素的第一代后代。而后代选择器是作用于所有子后代元素。后代选择器通过空格来进行选择，而子选择器是通过“>”进行选择。**

总结：>作用于元素的第一代后代，空格作用于元素的所有后代。

我的理解： 注意是第一代不是第一个

```
<!DOCTYPE HTML>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>后代选择器</title>
<style type="text/css">
.first span{color:red;}

.food>li{
    border:1px solid red;/*添加边框样式（粗细为1px， 颜色为红色的实线）*/
}
</style>
</head>

<body>
    <!--这不算是一个特别好的例子，因为第一代后代之后没有后代了-->
 <p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩，上课从来不敢回答老师提出的问题，生怕回答错了老师会批评我。就一直没有这个勇气来回答老师提出的问题。学校举办的活动我也没勇气参加。</p>
<!--下面是本小节任务代码-->
<ul class="food">
    <li>水果 <!--边框会应用在这些后代上，因为他们是第一代后代-->
        <ul>
            <li>香蕉</li><!--边框不会应用在这些后代上，因为他们不是第一代后代-->
            <li>苹果</li>
            <li>梨</li>
        </ul>
    </li>
    <li>蔬菜
        <ul>
            <li>白菜</li>
            <li>油菜</li>
            <li>卷心菜</li>
        </ul>
    </li>
</ul>
```

通用选择器

通用选择器是功能最强大的选择器，它使用一个（*）号指定，它的作用是匹配html中所有标签元素，如下使用下面代码使用html中**任意标签元素**字体颜色全部设置为红色：

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

伪类选择符

更有趣的是伪类选择符，为什么叫做伪类选择符，它允许给html不存在的标签**（标签的某种状态）**设置样式，比如说我们给html中一个标签元素的鼠标滑过的状态来设置字体颜色：

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

上面一行代码就是为 a 标签鼠标滑过的状态设置字体颜色变红。这样就会使第一段文字内容中的“胆小如鼠”文字加入鼠标滑过字体颜色变为红色特效。

关于伪类选择符，到目前为止，可以兼容所有浏览器的“伪类选择符”就是 `a` 标签上使用 `:hover` 了**（其实伪类选择符还有很多，尤其是 css3 中，但是因为不能兼容所有浏览器，本教程只是讲了这一种最常用的）**。其实 `:hover` 可以放在任意的标签上，比如说 `p:hover` ，但是它们的兼容性也是很不好的，所以现在比较常用的还是 `a:hover` 的组合。

分组选择符

当你想为html中多个标签元素设置同一个样式时，可以使用分组选择符（ `,` ），如下代码为右侧代码编辑器中的 `h1` 、 `span` 标签同时设置字体颜色为红色：

```
h1,span{color:red;}
```

它相当于下面两行代码：

```
h1{color:red;}
span{color:red;}
```

CSS的继承、层叠和特殊性

继承

CSS的某些样式是具有继承性的，那么什么是继承呢？**继承是一种规则，它允许样式不仅应用于某个特定html标签元素，而且应用于其后代。**比如下面代码：如某种颜色应用于p标签，这个颜色设置不仅应用p标签，还应用于p标签中的所有子元素文本，这里子元素为span标签。

```
p{color:red;}

<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p> <!--这里span的权重算继承-->
```

可见右侧结果窗口中p中的文本与span中的文本都设置为了红色。但注意有一些css样式是不具有继承性的。如 `border:1px solid red;`

```
p{border:1px solid red;}

<p>三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

在上面例子中它代码的作用只是给p标签设置了边框为1像素、红色、实心边框线，而对于子元素span是没用起到作用的。

特殊性

有的时候我们为同一个元素设置了不同的CSS样式代码，那么元素会启用哪一个CSS样式呢?我们来看一下面的代码：

```
p{color:red;}
.first{color:green;}
<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

p和.first都匹配到了p这个标签上，那么会显示哪种颜色呢？green是正确的颜色，那么为什么呢？是因为浏览器是根据权值来判断使用哪种css样式的，权值高的就使用哪种css样式。

下面是权值的规则：

标签的权值为1，类选择符的权值为10，ID选择符的权值最高为100。例如下面的代码：

```
p{color:red;} /*权值为1*/
p span{color:green;} /*权值为1+1=2*/
.warning{color:white;} /*权值为10*/
p span.warning{color:purple;} /*权值为1+1+10=12, what is span.warning*/
/*tested: span.warning与.warning span不同*/
#footer .note p{color:yellow;} /*权值为100+10+1=111*/
```

注意：还有一个权值比较特殊--继承也有权值但很低，有的文献提出它只有0.1，所以可以理解为继承的权值最低。 *我的理解：就是要往下加上子标签*

层叠

我们来思考一个问题：如果在html文件中对于同一个元素可以有多个css样式存在并且这多个css样式具有相同权重值怎么办？好，这一小节中的层叠帮你解决这个问题。

层叠就是在html文件中对于同一个元素可以有多个css样式存在，当有相同权重的样式存在时，会根据这些css样式的前后顺序来决定，**处于最后面的css样式会被应用。**

如下面代码:

```
p{color:red;}
p{color:green;}
<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

最后p中的文本会设置为green，这个层叠很好理解，理解为后面的样式会覆盖前面的样式。

所以前面的css样式优先级就不难理解了：

内联样式表（标签内部）> 嵌入样式表（当前文件中）> 外部样式表（外部文件中）。（就近原则）

重要性

我们在做网页代码的时，有些特殊的情况需要为某些样式设置具有最高权值，怎么办？这时候我们可以使用 `!important` 来解决。

如下代码：

```
p{color:red!important;}
p{color:green;}
<p class="first">三年级时，我还是一个<span>胆小如鼠</span>的小女孩。</p>
```

这时 p 段落中的文本会显示的red红色。

注意：!important要写在分号的前面

这里注意当网页制作者不设置css样式时，浏览器会按照自己的一套样式来显示网页。并且用户也可以在浏览器中设置自己习惯的样式，比如有的用户习惯把字号设置为大一些，使其查看网页的文本更加清楚。**这时注意样式优先级为：浏览器默认的样式 < 网页制作者样式 < 用户自己设置的样式，但记住 `!important` 优先级样式是个例外，权值高于用户自己设置的样式。**

CSS格式化排版

文字排版--字体 `font-family`

```
body{font-family:"宋体";}
```

这里注意不要设置不常用的字体，因为如果用户本地电脑上如果没有安装你设置的字体，就会显示浏览器默认的字体。（因为用户是否可以看到你设置的字体样式取决于用户本地电脑上是否安装你设置的字体。） 现在一般网页喜欢设置“微软雅黑”，如下代码：

```
body{font-family:"Microsoft Yahei";}
```

或

```
body{font-family:"微软雅黑";}
```

注意：第一种方法比第二种方法兼容性更好一些。

因为这种字体即美观又可以在客户端安全的显示出来（用户本地一般都是默认安装的）。

文字排版--字号、颜色 font-size: xpx, color

```
body{font-size:12px;color:#666} /*灰色*/
```

文字排版--粗体 font-weight:bold

```
p span{font-weight:bold;}
```

文字排版--斜体 font-style:italic

```
p a{font-style:italic;}

<p>三年级时，我还是一个<a>胆小如鼠</a>的小女孩。</p>
```

文字排版--下划线 text-decoration:underline

```
p a{text-decoration:underline;}

<p>三年级时，我还是一个<a>胆小如鼠</a>的小女孩。</p>
```

文字排版--删除线 text-decoration:line-through

```
 .oldPrice{text-decoration:line-through;}
```

段落排版--缩进 text-indent:xem

中文文字中的段前习惯空两个文字的空白，这个特殊的样式可以用下面代码来实现：

```
p{text-indent:2em;}
<p>1922年的春天，一个想要成名名叫尼克卡拉威（托比?马奎尔Tobey Maguire 饰）的作家，离开了美国中西部，来到了纽约。那是一个道德感渐失，爵士乐流行，走私为王，股票飞涨的时代。为了追寻他的美国梦，他搬入纽约附近一海湾居住。</p>
```

注意：2em的意思就是文字的2倍大小。

段落排版--行间距（行高） `line-height:xem`

如下代码实现设置段落行间距为1.5倍

```
p{line-height:1.5em;}
<p>菲茨杰拉德，二十世纪美国文学巨擘之一，兼具作家和编剧双重身份。他以诗人的敏感和戏剧家的想象为"爵士乐时代"吟唱华丽挽歌，其诗人和梦想家的气质亦为那个奢靡年代的不二注解。</p>
```

段落排版--中文字间距、字母间距 `letter-spacing:xpx` `word-spacing:xpx`

如果想在网页排版中设置文字间隔或者字母间隔就可以使用 `letter-spacing` 来实现，如下面代码：

```
h1{
    letter-spacing:50px;
}
...
<h1>了不起的盖茨比</h1>
```

注意：这个样式使用在英文单词时，是设置字母与字母之间的间距。

单词间距设置：

如果我想设置英文单词之间的间距呢？可以使用 `word-spacing` 来实现。如下代码：

```
h1{
    word-spacing:50px;
}
...
<h1>welcome to imooc!</h1>
```

段落排版--对齐 `text-align:center/left/right`

想为**块状元素**中的文本、图片设置居中样式吗？可以使用text-align样式代码，如下代码可实现文本居中显示。

```
h1{
    text-align:center;/*left or right*/
}
<h1>了不起的盖茨比</h1>
```

颜色值

在网页中的颜色设置是非常重要，有字体颜色（color）、背景颜色（background-color）、边框颜色（border）等，设置颜色的方法也有很多种：

英文命令颜色

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

RGB颜色

这个与 photoshop 中的 RGB 颜色是一致的，由 R(red)、G(green)、B(blue) 三种颜色的比例来配色。

```
p{color:rgb(133,45,200);}
```

每一项的值可以是 0~255 之间的整数，也可以是 0%~100% 的百分数。如：

```
p{color:rgb(20%,33%,25%);}
```

十六进制颜色

这种颜色设置方法是现在比较普遍使用的方法，其原理其实也是 RGB 设置，但是其每一项的值由 0-255 变成了十六进制 00-ff。

```
p{color:#00ffff;}
```

颜色缩写

关于颜色的css样式也是可以缩写的，当你设置的颜色是16进制的色彩值时，如果每两位的值相同，可以缩写一半。

`p{color:#000000;}` 可以缩写为：`p{color: #000;}`

`p{color: #336699;}` 可以缩写为：`p{color: #369;}`

字体缩写

网页中的字体css样式代码也有他自己的缩写方式，下面是给网页设置字体的代码：

```
body{
    font-style:italic;
    font-variant:small-caps;
    font-weight:bold;
    font-size:12px;
    line-height:1.5em;
    font-family:"宋体",sans-serif;
}
```

这么多行的代码其实可以缩写为一句：

```
body{
    font:italic  small-caps  bold  12px/1.5em  "宋体",sans-serif;
}
```

注意：

1、使用这一简写方式你至少要指定**font-size**和**font-family**属性，其他的属性(如 font-weight、font-style、font-variant、line-height)如未指定将自动使用默认值。

2、在缩写时**font-size**与**line-height**中间要加入“/”斜扛。

一般情况下因为对于中文网站，英文还是比较少的，所以下面缩写代码比较常用：

```
body{
    font:12px/1.5em  "宋体",sans-serif;
}
```

只是有字号、行间距、中文字体、英文字体设置。

长度值

长度单位总结一下，目前比较常用到px（像素）、em、% 百分比，要注意其实这三种单位都是相对单位。

像素

像素为什么是相对单位呢？因为像素指的是显示器上的小点（CSS规范中假设“90像素=1英寸”）。实际情况是浏览器会使用显示器的实际像素值有关，在目前大多数的设计者都倾向于使用像素（px）作为单位。

em

就是本元素给定字体的 font-size 值，如果元素的 font-size 为 14px ，那么 1em = 14px；如果 font-size 为 18px，那么 1em = 18px。如下代码：

```
p{font-size:12px;text-indent:2em;}
```

上面代码就是可以实现段落首行缩进 24px（也就是两个字体大小的距离）。

下面注意一个特殊情况：

但当给 font-size 设置单位为 em 时，此时计算的标准以**p的父元素的font-size**为基础。如下代码：

```
<p>以这个<span>例子</span>为例。</p>
```

```
p{font-size:14px}
span{font-size:0.8em;}
```

结果 span 中的字体“例子”字体大小就为 11.2px（14 * 0.8 = 11.2px）。

百分比

```
p{font-size:12px;line-height:130%}
```

设置行高（行间距）为字体的130%（12 * 1.3 = 15.6px）。

CSS盒模型

在讲解CSS布局之前，我们需要提前知道一些知识，在CSS中，html中的标签元素大体被分为三种不同的类型：**块状元素、内联元素(又叫行内元素)和内联块状元素。**

常用的块状元素有：`<div>、<p>、<h1>...<h6>、<ol>、<ul>、<dl>、<table>、<address>、<blockquote> 、<form>`

常用的内联元素有：`<a>、<span>、<br>、<i>、<em>、<strong>、<label>、<q>、<var>、<cite>、<code>`

常用的内联块状元素有：`<img>、<input>`

元素分类--块级元素

html中`<div>、 <p>、<h1>、<form>、<ul> 和 <li>`就是块级元素。设置`display:block`就是将元素显示为块级元素。如下代码就是将内联元素a转换为块状元素，从而使a元素具有块状元素特点。

```
a{display:block;}
```

块级元素特点：

1. **每个块级元素都从新的一行开始，并且其后的元素也另起一行。**（真霸道，一个块级元素独占一行）
2. **元素的高度、宽度、行高以及顶和底边距都可设置。**
3. 元素宽度在不设置的情况下，是它本身父容器的100%（和父元素的宽度一致），除非设定一个宽度。

元素分类--内联元素

在html中，`<span>、<a>、<label>、 <strong> 和<em>`就是典型的内联元素（行内元素）（inline）元素。当然块状元素也可以通过代码`display:inline`将元素设置为内联元素。如下代码就是将块状元素`div`转换为内联元素，从而使`div`元素具有内联元素特点。

```
 div{
     display:inline;
 }

.....

<div>我要变成内联元素</div>
```

内联元素特点：

1. 和其他元素都在一行上；
2. 元素的高度、宽度及顶部和底部边距**不可设置**；
3. 元素的宽度就是它包含的文字或图片的宽度，**不可改变**。

元素分类--内联块级元素

内联块状元素（inline-block）就是同时具备内联元素、块状元素的特点，代码`display:inline-block`就是将元素设置为内联块状元素。(css2.1新增)，`<img>、<input>`标签就是这种内联块状标签。

inline-block元素特点：

1. 和其他元素都在一行上；
2. 元素的高度、宽度、行高以及顶和底边距都可设置。

盒模型--边框(border)

盒子模型的边框就是围绕着内容及补白的线，这条线你可以设置它的粗细、样式和颜色(边框三个属性)。

如下面代码为 div 来设置边框粗细为 2px、样式为实心的、颜色为红色的边框：

```
div{
    border:2px  solid  red;
}
```

上面是 border 代码的缩写形式，可以分开写：

```
div{
    border-width:2px; /*thin, medium, thick, xpx(pixel)*/
    border-style:solid; /*dashed, dotted, solid*/
    border-color:red; /*pr
}
```

如果有想为p标签单独设置下边框，而其它三边都不设置边框样式怎么办呢？css 样式中允许只为一个方向的边框设置样式：

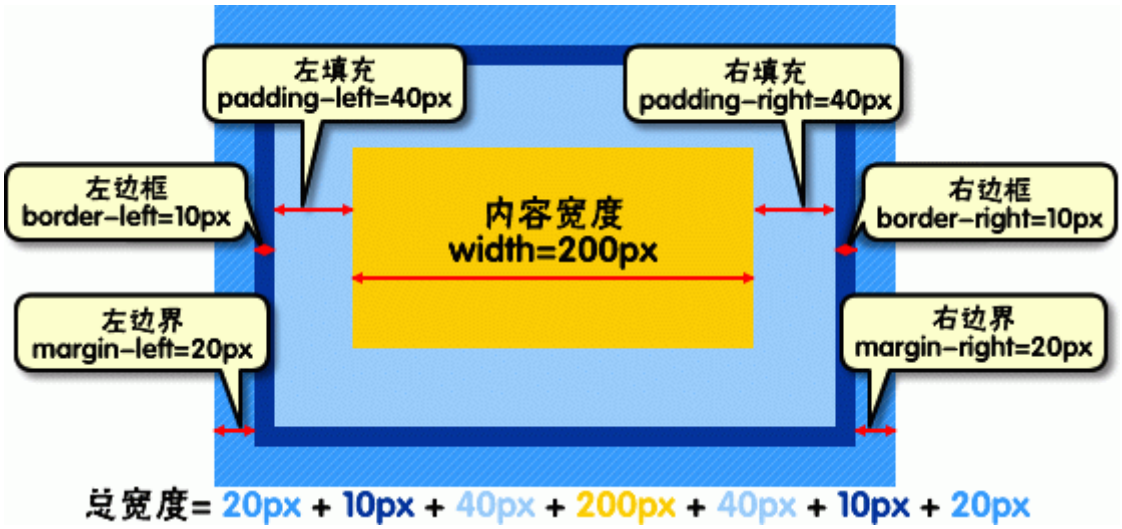
```
div{
    border-top:1px solid red;
    /*border-right:1px solid red; */
    /*border-left:1px solid red;*/
    /*border-bottom:1px solid red;*/
}
```

盒模型--宽度和高度(width and height)

盒模型宽度和高度和我们平常所说的物体的宽度和高度理解是不一样的，css内定义的宽（width）和高（height），指的是填充以里的内容范围。

因此一个元素实际宽度（盒子的宽度）=左边界+左边框+左填充+内容宽度+右填充+右边框+右边界。

元素的高度也是同理。



```
div{
    width:200px;
    padding:20px;
    border:1px solid red;
    margin:10px;
}

<body>
   <div>文本内容</div>
</body>
```

元素的实际长度为：10px+1px+20px+200px+20px+1px+10px=262px

盒模型--填充(padding)

元素内容与边框之间是可以设置距离的，称之为“填充”。填充也可分为上、右、下、左(顺时针)。如下代码：

```
div{padding:20px 10px 15px 30px;}
```

顺序一定不要搞混。可以分开写上面代码：

```
div{
   padding-top:20px;
   padding-right:10px;
   padding-bottom:15px;
   padding-left:30px;
}
```

如果上、右、下、左的填充都为10px;可以这么写

```
div{padding:10px;}
```

如果上下填充一样为10px，左右一样为20px，可以这么写：

```
div{padding:10px 20px;}
```

盒模型--边界(margin)

元素与其它元素之间的距离可以使用边界（margin）来设置。边界也是可分为上、右、下、左。如下代码：

```
div{margin:20px 10px 15px 30px;}
```

也可以分开写：

```
div{
   margin-top:20px;
   margin-right:10px;
   margin-bottom:15px;
   margin-left:30px;
}
```

如果上右下左的边界都为10px;可以这么写：

```
div{ margin:10px;}
```

如果上下边界一样为10px，左右一样为20px，可以这么写：

```
div{ margin:10px 20px;}
```

总结一下：padding和margin的区别，padding在边框里，margin在边框外。

盒模型代码简写

如果left和right的值相同 `margin:10px 20px 30px 20px;` 可缩写为： `margin:10px 20px 30px;`

注意：padding、border的缩写方法和margin是一致的。

css布局模型

清楚了CSS 盒模型的基本概念、 盒模型类型， 我们就可以深入探讨网页布局的基本模型了。布局模型与盒模型一样都是 CSS 最基本、 最核心的概念。 但布局模型是建立在盒模型基础之上，又不同于我们常说的 CSS 布局样式或 CSS 布局模板。如果说布局模型是本，那么 CSS 布局模板就是末了，是外在的表现形式。 CSS包含3种基本的布局模型，用英文概括为：**Flow、Layer 和 Float。** 在网页中，元素有三种布局模型：

1. 流动模型（Flow）
2. 浮动模型 (Float)
3. 层模型（Layer）

流动模型（Flow）

流动（Flow）是默认的网页布局模式。也就是说网页在默认状态下的 HTML 网页元素都是根据流动模型来分布网页内容的。

流动布局模型具有2个比较典型的特征：

第一点，**块状元素都会在所处的包含元素内自上而下按顺序垂直延伸分布，因为在默认状态下，块状元素的宽度都为100%。**实际上，块状元素都会以行的形式占据位置。

第二点，**在流动模型下，内联元素都会在所处的包含元素内从左到右水平分布显示。（内联元素可不像块状元素这么霸道独占一行）**

```
div{
    width:200px;
    height:200px;
    border:2px red solid;
    float:left; /*or both right, left and right*/
}
```

浮动模型 (Float)

块状元素这么霸道都是独占一行，如果现在我们想让两个块状元素并排显示，怎么办呢？不要着急，设置元素浮动就可以实现这一愿望。

任何元素在默认情况下是不能浮动的，但可以用 CSS 定义为浮动，如 div、p、table、img 等元素都可以被定义为浮动。

层模型（Layer）

层布局模型就像是图像软件PhotoShop中非常流行的图层编辑功能一样，每个图层能够精确定位操作，但在网页设计领域，由于网页大小的活动性，层布局没能受到热捧。但是在网页上局部使用层布局还是有其方便之处的。

CSS定义了一组定位（positioning）属性来支持层布局模型。

1. 绝对定位(position: absolute)
2. 相对定位(position: relative)
3. 固定定位(position: fixed)

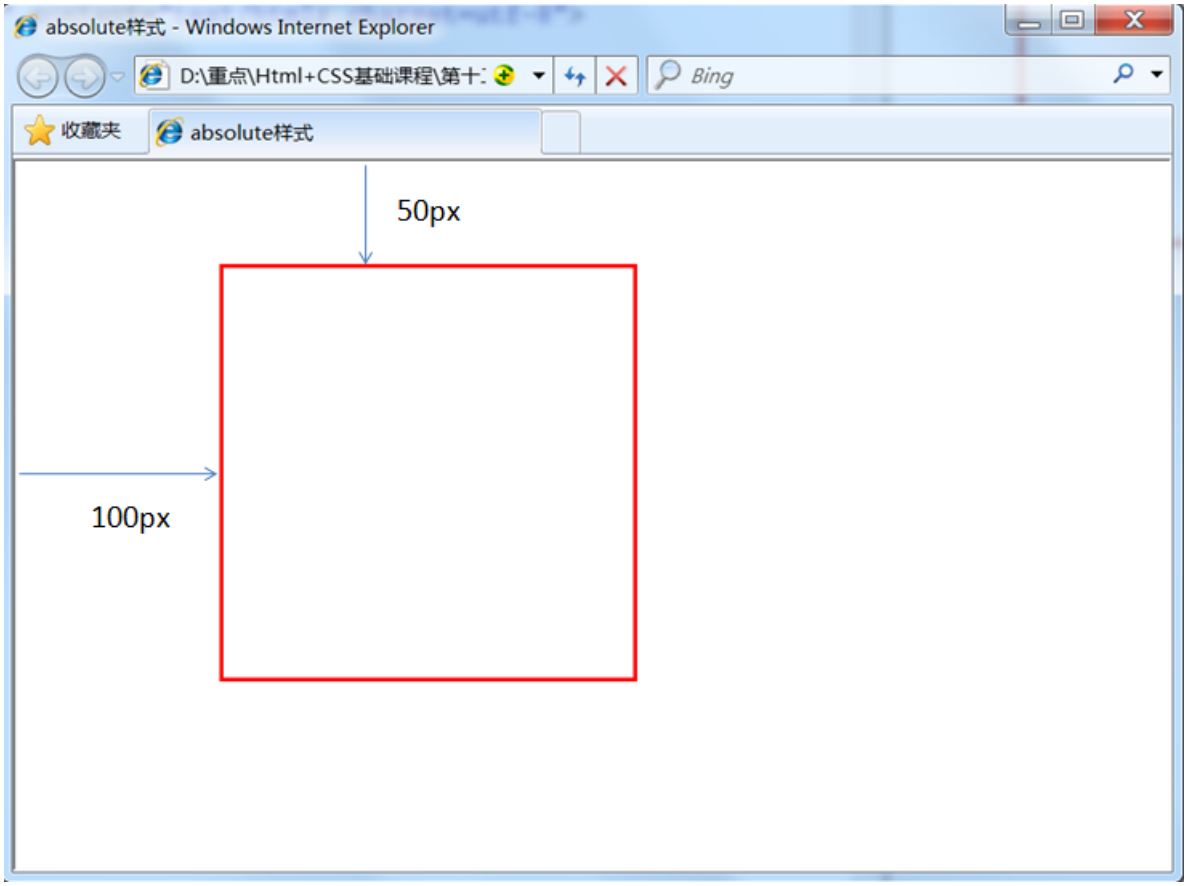
层模型--绝对定位

如果想为元素设置层模型中的绝对定位，需要设置 `position:absolute` (表示绝对定位)，这条语句的作用将元素从文档流中拖出来，然后使用left、right、top、bottom属性相对于其 ***最接近的一个具有定位属性的父包含块进行绝对定位*** 。**如果不存在这样的包含块，则相对于body元素，即相对于浏览器窗口。**

如下面代码可以实现div元素相对于浏览器窗口向右移动100px，向下移动50px。

```
div{
    width:200px;
    height:200px;
    border:2px red solid;
    position:absolute;
    left:100px; /*可以这么理解：距离（相对）浏览器窗口左边差100px，上边差50px*/
    top:50px;
}
<div id="div1"></div>
```

效果如下：



层模型--相对定位

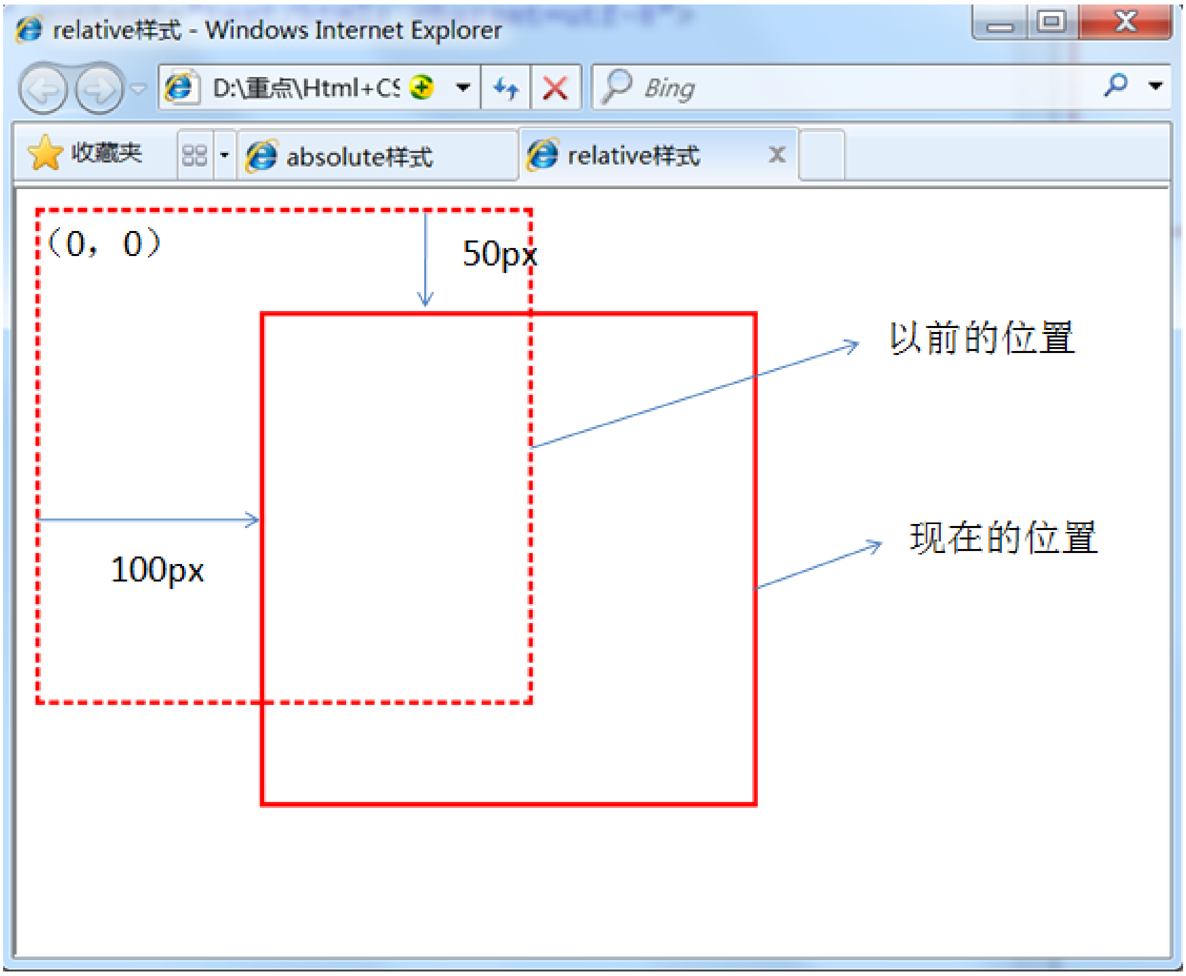
如果想为元素设置层模型中的相对定位，需要设置 `position:relative` （表示相对定位），它通过left、right、top、bottom属性确定**元素在正常文档流中的偏移位置。**相对定位完成的过程是首先按static(float)方式生成一个元素(并且元素像层一样浮动了起来)，然后**相对于以前的位置移动**，移动的方向和幅度由left、right、top、bottom属性确定， ***偏移前的位置保留不动*** 。

如下代码实现相对于以前位置向下移动50px，向右移动100px;

```
#div1{
    width:200px;
    height:200px;
    border:2px red solid;
    position:relative;
    left:100px;/*可以这么理解：距离（相对）浏览器窗口左边差100px，上边差50px*/
    top:50px;
}

<div id="div1"></div><!--<span>偏移前的位置还保留不动，覆盖不了前面的div没有偏移前的位置
</span>-->
```

效果图:



什么叫做“**偏移前的位置保留不动**”呢?

从效果图中可以明显的看出，虽然div元素相对于以前的位置产生了偏移，但是div元素以前的位置还是保留着，所以后面的span元素是显示在了div元素以前位置的后面。

层模型--固定定位

fixed：表示固定定位，与absolute定位类型类似，但它的相对移动的坐标是视图（屏幕内的网页窗口）本身。由于视图本身是固定的，它不会随浏览器窗口的滚动条滚动而变化，除非你在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小，因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流动影响，这与 `background-attachment:fixed;` 属性功能相同。以下代码可以实现相对于浏览器视图向右移动100px，向下移动50px。并且**拖动滚动条时位置固定不变。** *总在相对于视图的那个位置，想像你的屏幕上贴了个东西*

已测试：`position:absolute` 时，拖动滚动条时位置不是固定的

```
#div1{
    width:200px;
    height:200px;
    border:2px red solid;
    position:fixed;
    left:100px;
    top:50px;
}
<p>文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本文本。</p>
....
```

Relative与Absolute组合使用

小伙伴们学习了12-6小节的绝对定位的方法：使用 `position:absolute` 可以实现被设置元素**相对于浏览器（body）**设置定位以后，大家有没有想过可不可以相对于其它元素进行定位呢？答案是肯定的，当然可以。使用position:relative来帮忙，但是必须遵守下面规范：

1、参照定位的元素必须是相对定位元素的前辈元素：

```
<div id="box1"><!--参照定位的元素-->
    <div id="box2">相对参照元素进行定位</div><!--相对定位元素-->
</div>
```

从上面代码可以看出box1是box2的父元素（父元素当然也是前辈元素了）。

2、参照定位的元素必须加入position:relative;

```
#box1{
    width:200px;
    height:200px;
    position:relative;
}
```

3、定位元素加入position:absolute，便可以使用top、bottom、left、right来进行偏移定位了。

```
#box2{
    position:absolute;
    top:20px;
    left:30px;
}
```

这样box2就可以相对于父元素box1定位了（这里注意参照物就可以不是浏览器了，而可以自由设置了）。

CSS样式设置小技巧

水平居中设置-行内元素

如果被设置元素为**文本、图片**等行内元素时，水平居中是通过给父元素设置 `text-align:center` 来实现的。(父元素和子元素：如下面的html代码中，div是 *我想要在父容器中水平居中显示* 这个文本的父元素。反之这个文本是div的子元素)如下代码：

```
<body>
  <div class="txtCenter">我想要在父容器中水平居中显示。</div>
</body>
```

```
<style>
  .txtCenter{
    text-align:center;
  }
</style>
```

水平居中设置-定宽块状元素

当被设置元素为**块状元素**时用 `text-align: center` 就不起作用了，这时也分两种情况：定宽块状元素和不定宽块状元素。

这一小节我们先来讲一讲定宽块状元素。(定宽块状元素：块状元素的宽度width为固定值。)

满足定宽和块状两个条件的元素是可以通过设置“左右margin”值为“auto”来实现居中的。我们来看个例子就是设置 div 这个块状元素水平居中：

```
<body>
  <div>我是定宽块状元素，哈哈，我要水平居中显示。</div>
</body>
```

```
 <style>
 div{
     border:1px solid red;/*为了显示居中效果明显为 div 设置了边框*/

     width:200px;/*定宽*/
     margin:20px auto;/* margin-left 与 margin-right 设置为 auto */
 }
 /*也可以写成：*/
     /*margin-left:auto;*/
     /*margin-right:auto;*/

 </style>
```

水平居中总结-不定宽块状元素方法（一）

在实际工作中我们会遇到需要为“不定宽度的块状元素”设置居中，比如网页上的分页导航，因为分页的数量是不确定的，所以我们不能通过设置宽度来限制它的弹性。(不定宽块状元素：块状元素的宽度width不固定。)

不定宽度的块状元素有三种方法居中（这三种方法目前使用的都很多）：

1. 加入 `table` 标签
2. 设置 `display: inline` 方法：与第一种类似，显示类型设为 行内元素，进行不定宽元素的属性设置
3. 设置 `position:relative` 和 `left:50%` ：利用**相对定位**的方式，将元素向左偏移**50%**，即达到居中的目的 这一小节我们来讲一下第一种方法：

为什么选择方法一加入 `table` 标签? 是利用 `table` 标签的**长度自适应性**---即不定义其长度也不默认父元素body的长度（table其长度根据其内文本长度决定），**因此可以看做一个定宽度块元素**，然后再利用定宽度块状居中的margin的方法，使其水平居中。

第一步：为需要设置的居中的元素外面加入一个 `table` 标签（包括 `<tbody>、<tr>、<td>` ）。

第二步：为这个 `table` 设置**左右margin居中**（这个和定宽块状元素的方法一样）。

举例如下：

```
<div>
 <table>
  <tbody>
    <tr><td>
    <ul>
        <li>我是第一行文本</li>
        <li>我是第二行文本</li>
        <li>我是第三行文本</li>
    </ul>
    </td></tr>
  </tbody>
 </table>
</div>
```

```
<style>
table{
    border:1px solid;
    margin:0 auto;
}
</style>
```

水平居中总结-不定宽块状元素方法（二）

本节介绍第2种实现这种效果的方法，改变元素的display类型为行内元素，利用其属性直接设置。

第二种方法：改变块级元素的 `display` 为 `inline` 类型（设置为**行内元素**显示），然后使用 `text-align:cente` 来实现居中效果。如下例子：

```
<body>
<div class="container">
    <ul>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
    </ul>
</div>
</body>
```

```
<style>
.container{
    text-align:center;
}
/* margin:0;padding:0（消除文本与div边框之间的间隙）*/
.container ul{
    list-style:none;
    margin:0;
    padding:0;
    display:inline;
}
/* margin-right:8px（设置li文本之间的间隔）*/
.container li{
    margin-right:8px;
    display:inline;
}
</style>
```

这种方法相比第一种方法的优势是不用增加无语义标签，但也存在着一些问题：它将块状元素的 display 类型改为 inline，变成了行内元素，所以少了一些功能，比如设定长度值。

水平居中总结-不定宽块状元素方法（三）（？）

除了前两节讲到的插入table标签，以及改变元素的display类型，可以使不定宽块状元素水平居中之外，本节介绍第3种实现这种效果的方法，设置浮动和相对定位来实现。

方法三：通过给父元素设置 `float` ，然后给父元素设置 `position:relative` 和 `left:50%` ，子元素设置 `position:relative` 和 `left: -50%` 来实现水平居中。

我们可以这样理解：假想ul层的父层（即下面例子中的div层）中间有条平分线将ul层的父层（div层）平均分为两份，ul层的css代码是将ul层的最左端与ul层的父层（div层）的平分线对齐；而li层的css代码则是将li层的平分线与ul层的最左端（也是div层的平分线）对齐，从而实现li层的居中。

代码如下：

```
<body>
<div class="container">
    <ul>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
    </ul>
</div>
</body>
```

```
<style>
.container{
    float:left;
    position:relative;
    left:50%
}

.container ul{
    list-style:none;
    margin:0;
    padding:0;

    position:relative;
    left:-50%;
}
.container li{float:left;display:inline;margin-right:8px;}
</style>
```

垂直居中-父元素高度确定的单行文本

父元素高度确定的单行文本的竖直居中的方法是通过设置父元素的 `height` 和 `line-height` 高度一致来实现的。(`height` : 该元素的高度， `line-height` : 顾名思义，行高（行间距），指在文本中，行与行之间的 基线间的距离)。

`line-height` 与 `font-size` 的计算值之差，在 CSS 中成为“行间距”。分为两半，分别加到一个文本行内容的顶部和底部。

这种文字行高与块高一致带来了一个弊端：当文字内容的长度大于块的宽时，就有内容脱离了块。

```
<div class="container">
    hi,imooc!
</div>
```

```
<style>
.container{
    height:100px;
    line-height:100px;
    background:#999;
}
</style>
```

父元素给一行分配了100px，一整行将会占据100px，剩下的空位置（line-height和font-size计算值之差）将会分为两半，分别降到文本内容的顶部和底部

垂直居中-父元素高度确定的多行文本（方法一）

父元素高度确定的多行文本、图片等的竖直居中的方法有两种：

方法一：使用插入 `table` (包括 `tbody` 、 `tr` 、 `td`)标签，同时设置 `vertical-align: middle` 。

css 中有一个用于竖直居中的属性 `vertical-align` ，在父元素设置此样式时，会对 `inline-block` 类型的子元素都有用。下面看一下例子：

```
<body>
<table><tbody><tr><td class="wrap">
<div>
    <p>看我是否可以居中。</p>
</div>
</td></tr></tbody></table>
</body>
```

```
table td{height:500px;background:#ccc}
```

因为 td 标签默认情况下就默认设置了 `vertical-align` 为 `middle` ，所以我们不需要显式地设置了。

垂直居中-父元素高度确定的多行文本（方法二）

除了上一节讲到的插入 `table` 标签，可以使父元素高度确定的多行文本垂直居中之外，本节介绍另外一种实现这种效果的方法。但这种方法兼容性比较差，只是提供大家学习参考。

在 chrome、firefox 及 IE8 以上的浏览器下可以设置块级元素的 `display` 为 `table-cell` （设置为表格单元显示），激活 `vertical-align` 属性，但注意 IE6、7 并不支持这个样式，兼容性比较差。

```
<div class="container">
    <div>
        <p>看我是否可以居中。</p>
        <p>看我是否可以居中。</p>
        <p>看我是否可以居中。</p>
    </div>
</div>
```

```
<style>
.container{
    height:300px;
    background:#ccc;
    display:table-cell;/*IE8以上及Chrome、Firefox*/
    vertical-align:middle;/*IE8以上及Chrome、Firefox*/
}
</style>
```

这种方法的好处是不用添加多余的无意义的标签，但缺点也很明显，它的兼容性不是很好，不兼容IE6、7而且这样修改改display的block变成了table-cell，破坏了原有的块状元素的性质。

一中的方法因为是block所以图和文字不在一行上，但是二中的方法图和文字排在了一行上

隐性改变display类型

有一个有趣的现象就是当为元素（不论之前是什么类型元素，display:none 除外）设置以下 2 个句之一：

1. `position : absolute`
2. `float : left 或 float:right`

简单来说，只要html代码中出现以上两句之一，元素的display显示类型就会自动变为以 `display:inline-block` 的方式显示，当然就可以设置元素的 `width` 和 `height` 了，**且默认宽度不占满父元素。**

如下面的代码，小伙伴们都知道 a 标签是 行内元素 ，所以设置它的 width 是 没有效果的，但是设置为 position:absolute 以后，就可以了。

```
<div class="container">
    <a href="#" title="">进入课程请单击这里</a>
</div>
```

```
<style>
.container a{
    position:absolute;
    width:200px;
    background:#ccc;
}
</style>
```