HTML+CSS基础

本节目标

1. 前端三板斧(HTML、CSS、JavaScript)简介
2. HTML标签(上)
3. HTML标签(下)
4. CSS简介
5. CSS样式、选择器
6. CSS继承、层叠、特殊性
7. CSS格式化排版
8. CSS盒模型
9. CSS布局模型

一、前端三板斧简介

学习web前端开发基础技术需要掌握：HTML、CSS、JavaScript语言。下面我们就来了解下这三门技术都是用来实现什么的：

1. HTML是网页内容的载体。内容就是网页制作者放在页面上想要让用户浏览的信息，可以包含文字、图片、视频等。
2. CSS样式是表现。就像网页的外衣。比如，标题字体、颜色变化，或为标题加入背景图片、边框等。所有这些用来改变内容外观的东西称之为表现。
3. JavaScript是用来实现网页上的特效效果。如：鼠标滑过弹出下拉菜单。或鼠标滑过表格的背景颜色改变。还有焦点新闻（新闻图片）的轮换。可以这么理解，有动画的，有交互的一般都是用JavaScript来实现的。

第一个HTML代码:

```
<html>
<body>
<h1>My First Heading</h1>
<p>My first paragraph.</p>
</body>
</html>
```

大家可以看到，HTML是网页内容的具体载体。

第一个CSS代码:

```
<html>
<head>
<style type="text/css">
body {color:red}
```

```
h1 {color:#00ff00}
p.ex {color:rgb(0,0,255)}
</style>
</head>

<body>
<h1>My First Heading</h1>
<p>My first paragraph.在 body 选择器中定义了本页面中的默认文本颜色。</p>
<p class="ex">该段落定义了 class="ex"。该段落中的文本是蓝色的。</p>
</body>
</html>
```

CSS定义如何显示HTML元素

第一个JavaScript代码:

```
<html>
<body>

<script type="text/javascript">
document.write("Hello My First JavaScript Code!")
</script>

</body>
</html>
```

JavaScript动态展示页面效果

二、HTML标签(上)

2.1 HTML简介

HTML简介

HTML是用于创建网页的语言。我们通过使用HTML标记标签创建html文档来创建网页。HTML代表超文本标记语言。HTML是一种标记语言，它具有标记标签的集合。

HTML标签是由尖括号括起来的词，如`<html>`，`<body>`。标签通常成对出现，例如`<html>`和`</html>`。

一对中的第一个标签是开始标签;第二个标签是结束标签。在上面的示例中，`<html>`是开始标签，而</html>是结束标签。

我们还可以将开始标签称为起始标签，结束标签称为闭合标签。

HTML元素

以下代码显示了一个应用于某些文本的HTML元素的简单示例。

```
I like <code>web</code> and CSS.
```

起始标记为<code>，结束标记为</code>。标签之间是元素的内容，即web。标签和内容一起形成代码元素。

注意 1 我们使用元素来执行以下操作：

元素告诉浏览器您的内容。 元素的效果应用于元素内容。 每个HTML元素都有不同的和特定的含义。例如，代码元素表示计算机代码的片段。

注意 2 元素名称不区分大小写。HTML标签不区分大小写，<h1>和<H1>是一样的，但建议小写，因为大部分程序员都以小写为准。

浏览器会识别<CODE>和<code>，甚至<CoDe>作为代码元素的起始标签。

一般来说，惯例是采用单一案例格式并坚持。更常见的是使用小写字符。

注意 3 HTML定义了在HTML文档中实现各种角色的不同类型的元素。

代码元素是语义元素的示例。语义元素允许我们定义我们的内容的意义以及内容的不同部分之间的关系。

HTML文件结构

一个HTML文件是有自己固定的结构的。

```
<html>
    <head>...</head>
    <body>...</body>
</html>
```

代码讲解：

1. <html></html>称为根标签，所有的网页标签都在<html></html>中。
2. <head> 标签用于定义文档的头部，它是所有头部元素的容器。头部元素有<title>、<script>、<style>、<link>、 <meta>等标签，头部标签在下一小节中会有详细介绍。
3. 在<body>和</body>标签之间的内容是网页的主要内容，如<h1>、<p>、<a>、<img>等网页内容标签，在这里的标签中的内容会在浏览器中显示出来。

HTML标签与注释

标签:

下面这些标签可用在 head 部分：

```
<head>
    <title>...</title>
    <meta>
    <link>
    <style>...</style>
    <script>...</script>
</head>
```

HTML注释: 代码注释不仅方便程序员自己回忆起以前代码的用途，还可以帮助其他程序员很快的读懂你的程序的功能，方便多人合作开发网页代码。 语法：

```
<!--注释文字 -->
```

CSS注释代码： 就像在Html的注释一样，在CSS中也有注释语句：用/*注释语句*/来标明（Html中使用)

HTML语义化

语义化：说的通俗点就是：明白每个标签的用途（在什么情况下我可以使用这个标签才合理）比如，网页上的文章的标题就得用标题标签，网页上的各个栏目的栏目名称也可以使用标题标签。

语义化的作用:

1. 更容易被搜索引擎收录。
2. 更容易让屏幕阅读器读出网页内容。

HTML文档类型

<!DOCTYPE> 声明帮助浏览器正确地显示网页。

<!DOCTYPE> 声明

Web 世界中存在许多不同的文档。只有了解文档的类型，浏览器才能正确地显示文档。

HTML 也有多个不同的版本，只有完全明白页面中使用的确切 HTML 版本，浏览器才能完全正确地显示出 HTML 页面。这就是 <!DOCTYPE> 的用处。

不是 HTML 标签。它为浏览器提供一项信息（声明），即 HTML 是用什么版本编写的。

要注意的是:

- 必须放在HTML文档的第一行
- 声明不是HTML标签

常用的声明如下:

HTML5:

```
<!DOCTYPE html>
```

HTML 4.01:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

HTML乱码解决

当网页出现乱码时，在<head></head>标签之间添加: <meta charset="utf-8">

2.2 HTML文本标签

2.2.1 段落标签<p>

如果想在网页上显示文章，这时就需要<p>标签了，把文章的段落放到<p>标签中。 语法：

```
<p>段落文本</p>
```

范例：使用文本标签

```
<html>
<body>
<p>这是段落1。</p>
<p>这是段落2。</p>
<p>这是段落3。</p>
<p>段落元素由 p 标签定义。</p>
</body>
</html>
```

浏览器会自动地在段落的前后添加空行。（<p> 是块级元素）

`不要忘记结束标签`

即使忘了使用结束标签，大多数浏览器也会正确地将 HTML 显示出来：

范例:文本标签未使用结束标签

```
<html>
<body>
<p>This is a paragraph.
<p>This is a paragraph.
<p>不要忘记关闭你的 HTML 标签!</p>
</body>
</html>
```

上面的例子在大多数浏览器中都没问题，但不要依赖这种做法。忘记使用结束标签会产生意想不到的结果和错误。

`在未来的 HTML 版本中，不允许省略结束标签。`

我们无法确定 HTML 被显示的确切效果。屏幕的大小，以及对窗口的调整都可能导致不同的结果。

对于 HTML，您无法通过在 HTML 代码中添加额外的空格或换行来改变输出的效果。

当显示页面时，浏览器会移除*源代码中*多余的空格和空行。所有连续的空格或空行都会被算作一个空格。需要注意的是，HTML 代码中的所有连续的空行（换行）也被显示为一个空格。

范例:手工进行HTML格式化

```
<html>
<body>
<h1>春晓</h1>
<!--注意，浏览器忽略了源代码中的排版（省略了多余的空格和换行）-->
<p>
    春眠不觉晓，
      处处闻啼鸟。
         夜来风雨声，
            花落知多少。
</p>
</body>
</html>
```

2.2.2 标题标签<hx>

在 HTML 文档中，标题很重要。

标题（Heading）是通过<h1> - <h6> 等标签进行定义的。

<h1> 定义最大的标题。<h6> 定义最小的标题。

范例:观察各级标签

```
<html>
<body>
<!--请仅仅把标题标签用于标题文本。不要仅仅为了产生粗体文本而使用它们。请使用其它标签或 css 代替.-->
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
</body>
</html>
```

浏览器会自动地在标题的前后添加空行。

默认情况下，HTML 会自动地在块级元素前后添加一个额外的空行，比如段落、标题元素前后。

请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号的文本而使用标题。 应该将 h1 用作主标题（最重要的），其后是 h2（次重要的），再其次是 h3，以此类推。

2.2.3 强调标签、

有了段落又有了标题，现在如果想在一段话中特别强调某几个文字，这时候就可以用到或标签。

但两者在强调的语气上有区别: 表示强调， 表示更强烈的强调。并且在浏览器中默认用斜体表示， 用粗体表示。两个标签相比，目前国内前端程序员更喜欢使用表示强调

范例:使用强调标签

```
<html>
<body>
<em>em标签</em>
<br/>
<strong>strong标签</strong>
</body>
</html>
```

、、的区别：

1. 和标签是为了强调一段话中的关键字时使用，它们的语义是强调。
2. 标签是没有语义的，它的作用就是为了设置单独的样式用的，把一段话圈起来，然后用css设置样式。

2.2.4 引用标签

<q> 用于短的引用

浏览器通常会为 <q> 元素包围""。

语法： <q>引用文本</q>

范例:使用短引用

```
<html>
<body>
<!--浏览器通常会在 q 元素周围包围引号.-->
<p>以下文本为引用 <q>引用文本</q></p>
</body>
</html>
```

1. 要引用的文本不用加双引号，浏览器会对q标签自动添加双引号。
2. 这里用<q>标签的真正关键点不是它的默认样式双引号（如果这样我们不如自己在键盘上输入双引号就行了），而是它的**语义**：引用别人的话

<blockquote>用于长引用

<blockquote>的作用也是引用别人的文本。但它是对长文本的引用

<q>标签是对简短文本的引用，比如说引用一句话就用到<q>标签。

语法:

```
<blockquote>引用文本</blockquote>
```

范例：使用长引用标签

```
<html>
<body>
<p>以下为百度百科对于JavaScript的定义 <blockquote>JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。</blockquote></p>
</body>
</html>
```

浏览器对<blockquote>标签的解析是缩进样式

2.2.5 换行标签与分割线标签

2.2.5.1 换行标签

如果希望在不产生一个新段落的情况下进行换行（新行），请使用
标签：

标签作用相当于word文档中的回车。

范例:使用
标签

```
<html>
<body>
<p>
To break<br />lines<br>in a<br />paragraph,<br>use the br tag.
</p>
</body>
</html>
```


还是
?

大家也许发现
 与
 很相似。

在 XHTML、XML 以及未来的 HTML 版本中，不允许使用没有结束标签（闭合标签）的 HTML 元素。

即使
 在所有浏览器中的显示都没有问题，使用
 也是*更长远的保障*。

2.2.5.2 分隔线标签

<hr/>标签和
标签一样也是一个空标签，所以只有一个开始标签，没有结束标签。

范例：使用分割线标签

```
<html>
<body>
<p>
Java从入门到精通<br />
<hr />
HTML从入门到精通
</p>
</body>
</html>
```

<hr/>标签的在浏览器中的默认样式线条比较粗，颜色为灰色，可能有些人觉得这种样式不美观，没有关系，这些外在样式在我们以后学习了css样式表之后，都可以对其修改。

2.2.6 特殊字符

html特殊字符:

空格：` ` (;分号必不可少)

范例:使用` `

```
<html>
<body>
<h1>春晓</h1>
<!--注意，浏览器忽略了源代码中的排版（省略了多余的空格和换行）-->
<p>
春眠不觉晓，<br />
  处处闻啼鸟。<br />
    夜来风雨声，<br />
      花落知多少。<br />
</p>
</body>
</html>
```

其他常用特殊字符:

属性	显示结果	描述
`&reg;`	®	已注册
`&copy;`	©	版权
`&trade;`	™	商标

范例：观察其他特殊字符

```
<html>
<body>
<h1>观察特殊字符</h1>
<p>
注册:&reg;<br />
版权:&copy;<br />
商标:&trade;<br />
</p>
</body>
</html>
```

2.3 address标签

address标签，为网页加入地址信息

语法：

```
<address>地址信息</address>
```

范例:使用address标签

```
<!DOCTYPE html>
<html>
<body>
<address>中国北京市海淀区北太平庄西土城路10号</address>
</body>
</html>
```

在浏览器上显示的样式为斜体，如果不喜欢斜体，当然可以，可以在后面的课程中使用css样式来修改<address>标签的默认样式

2.4 code标签

在介绍语言技术的网站中，必免不了在网页中显示一些计算机专业的编程代码，当代码为**一行代码**时，你就可以使用<code>标签了

语法：

```
<code>代码语言</code>
```

范例:使用code标签

```
<html>
<body>
<code>var i = i+3000 </code>
</body>
</html>
```

在文章中一般如果要插入多行代码时不能使用<code>标签，如果是多行代码，可以使用<pre>标签。

<pre> 标签的主要作用:预格式化的文本。被包围在 pre 元素中的文本通常会保留空格和换行符。如果用以前的方法，回车需要输入
签，空格需要输入` `。

范例:使用pre标签

```
<html>
<body>
<pre>
// 以下为Java代码
int x = 0;
int y = 1;
int z = x+y;
</pre>
</body>
</html>
```

<pre> 标签不只是为显示计算机的源代码时用的，在你需要在网页中预显示格式时都可以使用它，只是<pre>标签的一个常见应用就是用来展示计算机的源代码。

三、HTML标签（下）

3.1 HTML列表标签

HTML 支持有序、无序和定义列表

3.1.1 无序列表

无序列表是一个项目的列表，此列项目使用粗体圆点（典型的小黑圆圈）进行标记。

无序列表始于 标签。每个列表项始于 。

范例:使用无序列表

```
<!DOCTYPE html>
<html>
<body>
<ul>
    <li>Java从入门到精通</li>
    <li>HTML、CSS、JavaScript从入门到精通</li>
    <li>JavaWeb从入门到精通</li>
    <li>颈椎病康复指南</li>
</ul>
</body>
</html>
```

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

3.1.2 有序列表

同样，有序列表也是一列项目，列表项目使用数字进行标记。

有序列表始于 <ol> 标签。每个列表项始于 <li> 标签。

范例：使用有序列表

```
<!DOCTYPE html>
<html>
<body>
<ol>
    <li>Java从入门到精通</li>
    <li>HTML、CSS、JavaScript从入门到精通</li>
    <li>JavaWeb从入门到精通</li>
    <li>颈椎病康复指南</li>
</ol>
</body>
</html>
```

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

3.1.3 定义列表

自定义列表不仅仅是一列项目，而是项目及其注释的组合。

自定义列表以 <dl> 标签开始。每个自定义列表项以 <dt> 开始。每个自定义列表项的定义以 <dd> 开始。

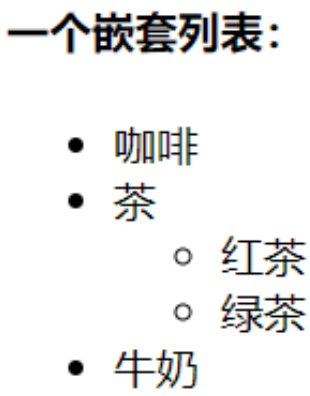
范例：使用自定义列表

```
<!DOCTYPE html>
<html>
<body>
```

```
<dl>
    <dt>JavaSE从入门到精通</dt>
    <dd>多线程</dd>
    <dd>集合类</dd>
    <dd>JVM</dd>
    <dt>前端从入门到精通</dt>
    <dd>HTML</dd>
    <dd>CSS</dd>
    <dd>JavaScript</dd>
</dl>
</body>
</html>
```

定义列表的列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

小练习:请用列表实现下列输出



3.2 HTML块

可以通过 和将HTML元素组合起来。

3.2.1 HTML块元素

大多数 HTML 元素被定义为块级元素或内联元素。

“块级元素”译为 block level element，“内联元素”译为 inline element。

块级元素在浏览器显示时，通常会以新行来开始（和结束）。

例子：<h1>, <p>, <ul>, <table>

3.2.2 HTML内联元素

内联元素在显示时通常不会以新行开始。 例子：<b>, <td>, <a>, <img>

3.2.3 HTML 元素

定义和用法

 可定义文档中的分区或节（division/section）。

 标签可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具，并且不使用任何格式与其关联。

如果用 id 或 class 来标记 ，那么该标签的作用会变得更加有效。

用法

 是一个块级元素。这意味着它的内容自动地开始一个新行。实际上，换行是 固有的唯一格式表现。可以通过 的 class 或 id 应用额外的样式。

不必为每一个 都加上类或 id，虽然这样做也有一定的好处。

可以对同一个 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。这两者的主要差异是，class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一的元素。

范例:使用div标签

```
<html>
<body>
<body>

 <h1>Java课程</h1>
  <p>Java课程分为SE、前端、EE三部分</p>
  ...
 <div class="JavaSE">
  <h2>JavaSE课程</h2>
  <p>SE需要大家学会基本语法、多线程、集合类的使用等</p>
  ...
 </div>

 <div class="JavaEE">
  <h2>JavaEE课程</h2>
  <p>EE需要大家学会Spring、SpringBoot框架等</p>
  ...
 </div>
 ...
</body>
</body>
</html>
```

上面这段 HTML 模拟了网站的结构。就是说，div 为文档添加了额外的结构。同时，由于这些 div 属于同一类元素，所以可以使用 class="JavaSE" 对这些 div 进行标识，这么做不仅为 div 添加了合适的语义，而且便于进一步使用样式对 div 进行格式化，可谓一举两得。

3.2.4 HTML元素

HTML 元素是内联元素，可用作文本的容器。 元素也没有特定的含义。 当与 CSS 一同使用时, 元素可用于为部分文本设置样式属性。

定义和用法

 标签被用来组合文档中的行内元素。

提示：请使用 来组合行内元素，以便通过样式来格式化它们。

注释：span 没有固定的格式表现。当对它应用样式时，它才会产生视觉上的变化。

范例:使用span标签

```
<p><span>some text.</span>some other text.</p>
```

如果不对 span 应用样式，那么 span 元素中的文本与其他文本不会任何视觉上的差异。尽管如此，上例中的 span 元素仍然为 p 元素增加了额外的结构。

可以为 span 应用 id 或 class 属性，这样既可以增加适当的语义，又便于对 span 应用样式。

可以对同一个 元素应用 class 或 id 属性，但是更常见的情况是只应用其中一种。这两者的主要差异是，class 用于元素组（类似的元素，或者可以理解为某一类元素），而 id 用于标识单独的唯一的元素。

3.3 HTML表格

3.3.1 表格

表格由 <table> 标签来定义。每个表格均有若干行（由 <tr> 标签定义），每行被分割为若干单元格（由 <td> 标签定义）。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

范例:HTML表格

```
<html>
<body>
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
</body>
</html>
```

3.3.2 表格和边框属性

如果不定义边框属性(border属性)，表格将不显示边框。有时这很有用，但是大多数时候，我们希望显示边框。

范例:不带边框表格

```
<html>
<body>
<table>
<tr>
<td>Row 1, cell 1</td>
<td>Row 1, cell 2</td>
</tr>
</table>
</table>
</body>
</html>
```

3.3.3 表格的表头

表格的表头使用 <th> 标签进行定义。

大多数浏览器会把表头显示为粗体居中的文本：

范例:表格表头

```
<html>
<body>
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
</body>
</html>
```

3.3.4 表格中的空单元格

在一些浏览器中，没有内容的表格单元显示得不太好。如果某个单元格是空的（没有内容），浏览器可能无法显示出这个单元格的边框。

范例:单元格无内容

```
<html>
<body>
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td></td>
<td>row 2, cell 2</td>
</tr>
</table>
</body>
</html>
```

注意：这个空的单元格的边框没有被显示出来。为了避免这种情况，在空单元格中添加一个空格占位符，就可以将边框显示出来。

范例:在空单元格中添加空格占位符

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td> </td>
<td>row 2, cell 2</td>
</tr>
</table>
```

3.3.5 带有标题的表格

caption 元素定义表格标题。

caption 标签必须紧随 table 标签之后。只能对每个表格定义一个标题。通常这个标题会被居中于表格之上。

范例:使用caption元素为表格定义标题

```
<html>
<body>
<table border="1">
<caption>我的标题</caption>
<tr>
  <td>100</td>
```

```
  <td>200</td>
  <td>300</td>
</tr>
<tr>
  <td>400</td>
  <td>500</td>
  <td>600</td>
</tr>
</table>
</body>
```

3.4 HTML链接

HTML 使用超级链接与网络上的另一个文档相连。

几乎可以在所有的网页中找到链接。点击链接可以从一张页面跳转到另一张页面。

3.4.1 HTML超链接

超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，你可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用 <a> 标签在 HTML 中创建链接。

有两种使用 <a> 标签的方式：

1. 通过使用 href 属性 - 创建指向另一个文档的链接
2. 通过使用 name 属性 - 创建文档内的书签

3.4.2 链接语法

链接的 HTML 代码很简单。 href 属性规定链接的目标。

开始标签和结束标签之间的文字被作为超级链接来显示。

范例 : html链接

```
<html>
<body>
<a href="http://www.bitedu.vip/#/">比特官网</a>
</body>
```

点击这个超链接会把用户带到比特的官网首页。

注意："链接文本" 不必一定是文本。图片或其他 HTML 元素都可以成为链接。

3.4.3 HTML链接-target属性

使用 Target 属性，你可以定义被链接的文档在何处显示。

下面的这行会在新窗口打开文档：

```
<html>
<body>
<a href="http://www.bitedu.vip/#/" target="_blank">比特官网</a>
</body>
</html>
```

3.4.4 HTML链接-name属性

name 属性规定锚（anchor）的名称。

可以使用 name 属性创建 HTML 页面中的书签。

书签不会以任何特殊方式显示，它对读者是不可见的。

当使用命名锚（named anchors）时，我们可以创建直接跳至该命名锚（比如页面中某个小节）的链接，这样使用者就无需不停地滚动页面来寻找他们需要的信息了。

命名锚的语法：

```
<a name="label">锚（显示在页面上的文本）</a>
```

提示：锚的名称可以是任何你喜欢的名字。

提示：您可以使用 id 属性来替代 name 属性，命名锚同样有效。

范例：使用name属性

首先，我们在HTML中对锚进行命名(创建一个书签)

```
<a name="tips">锚点(书签点)</a>
```

然后，我们在同一个文档中创建指向该锚的链接：

```
<a href="#tips">跳转到锚点</a>
```

完整代码如下:

```
<html>

<a name="tips">锚点(书签点)</a>

<p>
陪考家长等待参加首场考试的孩子。中新社记者 沙见龙 摄

中新社记者走访拥有重庆多所重点高中的沙坪坝区发现，临近考场附近的宾馆房间早在10天前就已订满。除了快捷酒店和普通宾馆，五星级酒店也加入了高考房的销售。
</p>

<p>“高考期间的入住率能赶上旺季的节假日。”沙坪坝一家星级酒店前台工作人员告诉记者，除了每年都会推出的高考房，今年他们还推出了“高考套餐”：包含两晚八餐的食宿服务，同时配备专车接送考生。
```

```
“这个套餐是今年才开始推的，高考期间卖出了20多套。”该工作人员说，一个月前已经有不少家长打电话或上门咨询高考套餐的情况，“今年明显觉得家长对高考房的配套上要求高了，所以我们也推出多样化的产品满足不同的需求”。</p>

<p>高考期间，进口营养保健品也受到家长追捧。药店里，抗疲劳、健脑等进口保健品被摆放在显眼的货架上。据售货员介绍，这些保健品单盒价格约为300至1000元人民币不等。虽然不便宜，但在高考前半月，好几种保健品卖到脱销。</p>

<p>“毕竟在高考这样的‘人生大事’面前，花点钱不算什么。”一家药店的店长介绍称，五月以来，选购各种针对青少年的营养保健品的家长明显增多。从功能上看，多数家长主要选购的是补脑安神和增强记忆力的营养品。“大家都想在高考期间，让孩子们发挥到最佳状态。”</p>

<p>“家长望子成龙的心在任何时候都是不变的。参加过高考的家长对这场考试会有更直观的认识，相对‘一考定终身’的想法没那么强烈。”西南大学博士生导师董小玉接受采访时表示，随着个性更鲜明的“00后”成为高考主力，高考本身也在发生着变化。“近几年明显感觉无论是考生还是家长，对待高考都更加理性。”董小玉说，虽然未来高考的热度并不会降低，“但让考试回归考试，这本身就是一件好事”。（完）</p>

<a href="#tips">跳转到锚点</a>
</body>
</html>
```

还可以在其他页面中创建指向该锚的链接:

```
<a href="http://www.w3school.com.cn/html/html_links.asp#tips">有用的提示</a>
```

在上面的代码中，我们将 # 符号和锚名称添加到 URL 的末端，就可以直接链接到 tips 这个命名锚了。

3.5 HTML图像

3.5.1 图像标签和源属性

在 HTML 中，图像由 <img> 标签定义。

<img> 是空标签，意思是说，它只包含属性，并且没有闭合标签。

要在页面上显示图像，你需要使用源属性（src）。src 指 "source"。源属性的值是图像的 URL 地址。

定义图像的语法是:

```
<img src="url" />
```

URL 指存储图像的位置。如果名为 "Fitting room.jpg " 的图像位于https://justtalk.oss-cn-shanghai.aliyuncs.com 的 picture目录中，那么其 URL 为 https://justtalk.oss-cn-shanghai.aliyuncs.com/picture/Fitting%20room.jpg

浏览器将图像显示在文档中图像标签出现的地方。如果你将图像标签置于两个段落之间，那么浏览器会首先显示第一个段落，然后显示图片，最后显示第二段。

范例：使用img标签

```
<html>
<body>
    <img src="Java课件/十元1.jpg">
</body>
</html
```

3.5.2 替换文本属性(alt属性)

alt 属性用来为图像定义一串预备的可替换的文本。替换文本属性的值是用户定义的。

在浏览器无法载入图像时，替换文本属性告诉读者她们失去的信息。此时，浏览器将显示这个替代性的文本而不是图像。为页面上的图像都加上替换文本属性是个好习惯，这样有助于更好的显示信息，并且对于那些使用纯文本浏览器的人来说是非常有用的。

范例:使用替换文本

```
<img src="boat.gif" alt="Big Boat">
```

3.5.3 背景图片

gif 和 jpg 文件均可用作 HTML 背景。body标签的background属性设置背景图片。

如果图像小于页面，图像会进行重复。

范例:将图片用于背景

```
<html>

<body background="Java课件/十元1.jpg">

<h3>图像背景</h3>

</body>
</html>
```

3.6 HTML表单

3.6.1 HTML表单

HTML 表单用于搜集不同类型的用户输入。

3.6.1.1 <form>元素

<form> 元素

HTML 表单用于收集用户输入。

<form> 元素定义 HTML 表单语法如下:

```
<form>
 .
form elements
 .
</form>
```

HTML 表单包含*表单元素*。

表单元素指的是不同类型的 input 元素、复选框、单选按钮、提交按钮等等。

3.6.1.2 <input>元素

<input> 元素是最重要的*表单元素*。

<input> 元素有很多形态，根据不同的 *type* 属性。

常用属性如下

类型	描述
text	定义常规文本输入。
radio	定义单选按钮输入（选择多个选择之一）
submit	定义提交按钮（提交表单）

文本输入

<input type="text"> 定义用于*文本输入*的单行输入字段：

范例:使用文本输入

```
<!DOCTYPE html>
<html>
<body>
<form>
userName:<br>
<input type="text" name="userName">
<br>
password:<br>
<input type="password" name="password">
</form>
</body>
</html>
```

表单本身并不可见。还要注意文本字段的默认宽度是 20 个字符。

单选按钮输入

<input type="radio"> 定义*单选按钮*。

单选按钮允许用户在有限数量的选项中选择其中之一

范例:使用单选按钮

```
<!DOCTYPE html>
<html>
<body>
<form>
<p>请选择性别</p>
<input type="radio" name="sex" value="男" checked>男
<br>
<input type="radio" name="sex" value="女">女
</form>
</body>
</html>
```

提交按钮

<input type="submit"> 定义用于向*表单处理程序*（form-handler）*提交*表单的按钮。

表单处理程序通常是包含用来处理输入数据的脚本的服务器页面。

表单处理程序在表单的 *action* 属性中指定：

```
<!DOCTYPE html>
<html>
<body>
<form action="http://www.w3school.com.cn//demo/demo_form.asp">
First name:<br>
<input type="text" name="firstname" value="liu">
<br>
Last name:<br>
<input type="text" name="lastname" value="yiming">
<br><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Action 属性

*action 属性*定义在提交表单时执行的动作。

向服务器提交表单的通常做法是使用提交按钮。

通常，表单会被提交到 web 服务器上的网页。

在上面的例子中，指定了某个服务器脚本来处理被提交表单：

```
<form action="http://www.w3school.com.cn//demo/demo_form.asp">
```

如果省略 action 属性，则 action 会被设置为当前页面。

Method 属性

*method 属性*规定在提交表单时所用的 HTTP 方法（*GET* 或 *POST*）：

```
<form action="http://www.w3school.com.cn//demo/demo_form.asp" method="GET">
```

或：

```
<form action="http://www.w3school.com.cn//demo/demo_form.asp" method="POST">
```

Name 属性

如果要正确地被提交，每个输入字段必须设置一个 name 属性。

范例:下方代码只会提交一个字段

```
<!DOCTYPE html>
<html>
<body>
<form action="/demo/demo_form.asp">
First name:<br>
<input type="text" value="liu">
<br>
Last name:<br>
<input type="text" name="lastName" value="yiming">
<br><br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

用 <fieldset> 组合表单数据

<*fieldset*> 元素组合表单中的相关数据

<*legend*> 元素为<fieldset> 元素定义标题。

```
<!DOCTYPE html>
<html>
<body>
<form action="http://www.w3school.com.cn//demo/demo_form.asp">
<fieldset>
<legend>Personal information:</legend>
First name:<br>
<input type="text" name="firstname" value="liu">
```

```
<br>
Last name:<br>
<input type="text" name="lastname" value="yiming">
<br><br>
<input type="submit" value="Submit">
</fieldset>
</form>
</body>
</html>
```

HTML Form 属性

HTML<form> 元素，已设置所有可能的属性如下:

```
<form action="action_page.php" method="GET" target="_blank" accept-charset="UTF-8"
ectype="application/x-www-form-urlencoded" autocomplete="off" novalidate>
.
form elements
 .
</form>
```

属性	描述
accept-charset	规定在被提交表单中使用的字符集（默认：页面字符集）。
action	规定向何处提交表单的地址（URL）（提交页面）。
autocomplete	规定浏览器应该自动完成表单（默认：开启）。
enctype	规定被提交数据的编码（默认：url-encoded）。
method	规定在提交表单时所用的 HTTP 方法（默认：GET）。
name	规定识别表单的名称（对于 DOM 使用：document.forms.name）。
novalidate	规定浏览器不验证表单。
target	规定 action 属性中地址的目标（默认：_self）。

3.6.2 表单元素

3.6.2.1 <select> 元素（下拉列表）

<select> 元素定义 下拉列表:

范例:使用下拉列表

```
<!DOCTYPE html>
<html>
```

```
<body>
<form action="/demo/demo_form.asp">
<select name="cars">
<option value="Benz>奔驰</option>
<option value="BMW">宝马</option>
<option value="Audi">奥迪</option>
<option value="Ford">福特</option>
</select>
<br><br>
<input type="submit">
</form>
</body>
</html>
```

<option> 元素定义待选择的选项。

列表通常会把首个选项显示为被选选项。

可以通过添加 selected 属性来定义预定义选项。

范例:下拉列表添加预定义选项

```
<option value="BMW" selected>宝马</option>
```

3.6.2.2 <textarea> 元素

<textarea> 元素定义多行输入字段(*文本域*)

范例:使用多行输入

```
<!DOCTYPE html>
<html>
<body>
<form>
<textarea name="message" rows="10" cols="30">
The cat was playing in the garden.
</textarea>
</form>
</body>
</html>
```

3.6.2.3 <button>元素

<button> 元素定义可点击的*按钮*:

```
<!DOCTYPE html>
<html>
<body>
<form>
<button type="button" onclick="alert('Hello World!')">点击我! </button>
</form>
</body>
</html>
```

3.6.2.4 HTML5新增 <datalist>元素

<*datalist*> 元素为 <input> 元素规定预定义选项列表。

用户会在他们输入数据时看到预定义选项的下拉列表。

<input> 元素的 *list* 属性必须引用 <datalist> 元素的 *id* 属性。

```
<!DOCTYPE html>
<html>
<body>
<form action="action_page.php">
<input list="cars">
<datalist id="cars">
   <option value="Benz">
   <option value="Audi">
   <option value="BMW">
   <option value="Ford">
</datalist>
<input type="submit">
</form>
</body>
</html>
```

3.6.3 HTML输入类型

3.6.3.1 checkbox定义复选框

<input type="checkbox"> 定义复选框。

复选框允许用户在有限数量的选项中选择零个或多个选项。

范例:使用复选框

```
<!DOCTYPE html>
<html>
<body>
<form action="http://www.w3school.com.cn/demo/demo_form.asp">
<input type="checkbox" name="vehicle" value="Bike">I have a bike
<br>
<input type="checkbox" name="vehicle" value="Car">I have a car
<br><br>
<input type="submit">
</form>
</body>
</html>
```

3.6.3.2 HTML5 新增的number输入

<input type="number"> 用于应该包含数字值的输入字段。

您能够对数字做出限制。

根据浏览器支持，限制可应用到输入字段。

```
<!DOCTYPE html>
<html>
<body>
<p>
根据浏览器支持：<br>
数值约束会应用到输入字段中。
</p>
<form action="/demo/demo_form.asp">
  数量（1 到 5 之间）：
  <input type="number" name="quantity" min="1" max="5">
  <input type="submit">
</form>
</body>
</html>
```

3.6.3.3 HTML5新增的date输入

<input type="date"> 用于应该包含日期的输入字段。

根据浏览器支持，日期选择器会出现输入字段中。还可以向输入添加限制

```
<!DOCTYPE html>
<html>
<body>
<form action="/demo/demo_form.asp">
请输入 2020-01-01 之前的日期：<br>
<input type="date" name="bday" max="2020-01-01"><br><br>
请输入 2000-01-01 之后的日期：<br>
<input type="date" name="bday" min="2000-01-02"><br><br>
<input type="submit">
</form>
</body>
</html>
```

3.6.3.4 HTML新增的color输入

<input type="color"> 用于应该包含颜色的输入字段。

根据浏览器支持，颜色选择器会出现输入字段中。

```
<!DOCTYPE html>
<html>
<body>
<form action="action_page.php">
  Select your favorite color:
  <input type="color" name="favcolor" value="#ff0000">
  <input type="submit">
</form>
</body>
</html>
```

3.6.3.5 HTML新增的email输入

<input type="email"> 用于应该包含电子邮件地址的输入字段。

根据浏览器支持，能够在被提交时自动对电子邮件地址进行验证。

某些智能手机会识别 email 类型，并在键盘增加 ".com" 以匹配电子邮件输入。

范例:使用email输入

```
<!DOCTYPE html>
<html>
<body>
<form action="/demo/demo_form.asp">
  E-mail:
  <input type="email" name="email">
  <input type="submit">
</form>
</body>
</html>
```

3.6.4 HTML输入属性

3.6.4.1 value属性

value 属性规定输入字段的初始值：

```
<!DOCTYPE html>
<html>
<body>
<form action="">
First name:<br>
<input type="text" name="firstname" value="yuisama">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
</body>
</html>
```

3.6.4.2 readonly属性

readonly 属性规定输入字段为只读（不能修改）：

```
<!DOCTYPE html>
<html>
<body>
<form action="">
First name:<br>
<input type="text" name="firstname" value ="yuisama" readonly>
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
</body>
</html>
```

3.6.4.3 disabled属性

disabled 属性规定输入字段是禁用的。

被禁用的元素是不可用和不可点击的。

被禁用的元素不会被提交。

```
<!DOCTYPE html>
<html>
<body>
<form action="">
First name:<br>
<input type="text" name="firstname" value ="yuisama" disabled>
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
</body>
</html>
```

3.6.4.4 maxlength属性

maxlength 属性规定输入字段允许的最大长度：

```
<!DOCTYPE html>
<html>
<body>
<form action="">
First name:<br>
<input type="text" name="firstname" maxlength="5">
<br>
Last name:<br>
<input type="text" name="lastname">
</form>
</body>
</html>
```

四、CSS简介

4.1 CSS概述

第一个CSS程序

```
<html>
<head>
<style type="text/css">
body {color:red}
h1 {color:#00ff00}
p.ex {color:rgb(0,0,255)}
</style>
</head>
<body>
<h1>第一个CSS程序</h1>
<p>hello my first css code</p>
```

```
<p class="ex">该标签class字段为ex，显示为蓝色</p>
</body>
</html>
```

- CSS 指层叠样式表 (*C*ascading *S*tyle *S*heets)
- 样式定义*如何显示* HTML 元素
- 样式通常存储在*样式表*中
- 把样式添加到 HTML 4.0 中，是为了*解决内容与表现分离的问题*
- *外部样式表*可以极大提高工作效率
- 外部样式表通常存储在 *CSS 文件*中
- 多个样式定义可*层叠*为一

4.2 样式层叠次序

当同一个 HTML 元素被不止一个样式定义时，会使用哪个样式呢?

一般而言，所有的样式会根据下面的规则层叠于一个新的虚拟样式表中，其中数字 4 拥有最高的优先权。

1. 浏览器缺省设置
2. 外部样式表
3. 内部样式表（位于 <head> 标签内部）
4. 内联样式（在 HTML 元素内部）

因此，内联样式（在 HTML 元素内部）拥有最高的优先权，这意味着它将优先于以下的样式声明：<head> 标签中的样式声明，外部样式表中的样式声明，或者浏览器中的样式声明（缺省值）。

4.3 CSS基础语法

CSS 语法

CSS 规则由两个主要的部分构成：选择器，以及一条或多条声明。规则如下:

```
selector {declaration1; declaration2; ... declarationN }
```

选择器通常是需要改变样式的 HTML 元素。

每条声明由一个属性和一个值组成。

属性（property）是希望设置的样式属性（style attribute）。每个属性有一个值。属性和值被冒号分开。

```
selector {property: value}
```

范例:使用CSS基础语法

```
h1 {color:red; font-size:14px;}
```

在上面代码中，我们将 h1 元素内的文字颜色定义为红色，同时将字体大小设置为 14 像素。

此处的h1就是我们的选择器，color 和 font-size 是属性，red 和 14px 是值。

值的不同写法和单位

除了英文单词 red，我们还可以使用十六进制的颜色值 #ff0000：

```
p { color: #ff0000; }
```

为了节约字节，我们可以使用 CSS 的缩写形式：

```
p { color: #f00; }
```

我们还可以通过两种方法使用 RGB 值：

```
p { color: rgb(255,0,0); }
p { color: rgb(100%,0%,0%); }
```

请注意，当使用 RGB 百分比时，即使当值为 0 时也要写百分比符号。但是在其他的情况下就不需要这么做了。比如说，当尺寸为 0 像素时，0 之后不需要使用 px 单位，因为 0 就是 0，无论单位是什么。

记得写引号

如果值为若干单词，则要给值加引号：

```
p {font-family: "sans serif";}
```

多重声明：

提示：如果要定义不止一个声明，则需要用分号将每个声明分开。下面的例子展示出如何定义一个红色文字的居中段落。最后一条规则是不需要加分号的，因为分号在英语中是一个分隔符号，不是结束符号。然而，大多数有经验的设计师会在每条声明的末尾都加上分号，这么做的好处是，当你从现有的规则中增减声明时，会尽可能地减少出错的可能性。就像这样：

```
p {text-align:center; color:red;}
```

你应该在每行只描述一个属性，这样可以增强样式定义的可读性，就像这样：

```
p {
  text-align: center;
  color: black;
  font-family: arial;
}
```

空格和大小写

大多数样式表包含不止一条规则，而大多数规则包含不止一个声明。多重声明和空格的使用使得样式表更容易被编辑：

```
body {
  color: #000;
  background: #fff;
  margin: 0;
  padding: 0;
  font-family: Georgia, Palatino, serif;
  }
```

是否包含空格不会影响 CSS 在浏览器的工作效果，同样，与 XHTML 不同，CSS 对大小写不敏感。不过存在一个例外：如果涉及到与 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的。

五、CSS样式、选择器

5.1 CSS样式单的基本使用

CSS样式单可以控制HTML文档的显示。为了在HTML中使用CSS，有如下四种方式使用样式单:

1. 链接外部样式文件:这种方式将样式文件彻底与HTML文档分离，样式文件需要额外引用。在这种方式下，一批样式可以控制多个HTML文档。
2. 导入外部式文件:这种方式与上一方式类似，只是使用@import来引入外部样式文件。
3. 使用内部样式定义:此方式是通过在HTML文档头定义样式单部分来实现的。在这种方式下，每批css样式只控制一份HTML文档。
4. 使用内联样式:这种方式将样式内联定义到具体的HTML元素，通常用于精确控制一个HTML元素的表现。在这种方式下，每份css样式只控制单个HTML元素。

5.1.1 链接外部样式文件

HTML中使用<link.../>元素来引入外部样式文件，引入外部样式文件应在<head../>元素增加如下<link../>子元素:

```
<link type="text/css" rel="stylesheet" href="CSS样式文件的URL"
```

在上面的语法格式中，type和rel表明该页面使用了CSS样式单，对于引入CSS样式单情形，这两个属性的值无法改变。href属性的值则指向CSS样式单文件的地址，此处的地址既可以是相对地址，也可以是互联网的绝对地址。

范例：html链接外部css文件。

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>链接外部CSS样式单</title>
        <link rel="stylesheet" type="text/css" href="Outter.css">
</head>
<body>
        <table>
                <tr>
                        <td>HTML从入门到精通</td>
```

```
                </tr>
                <tr>
                        <td>CSS从入门到精通</td>
                </tr>
                <tr>
                        <td>JavaScript从入门到精通</td>
                </tr>
        </table>
</body>
</html>
```

5.1.2 导入外部样式单

导入外部样式单的功能与链接外部样式单的功能差不多，只是语法上存在差别。导入外部样式单需要在<style.../>元素中使用@import来执行导入。

使用@import的完整语法如下:

```
@import url (样式单地址)
```

范例:使用import导入外部样式单

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>链接外部CSS样式单</title>
    <style type="text/css">
        @import "Outter.css";
    </style>
</head>
<body>
    <table>
        <tr>
            <td>HTML从入门到精通</td>
        </tr>
        <tr>
            <td>CSS从入门到精通</td>
        </tr>
        <tr>
            <td>JavaScript从入门到精通</td>
        </tr>
    </table>
</body>
</html>
```

链接外部样式单与导入外部样式单的功能差不多，但还是推荐大家使用链接外部样式单。因为某些浏览器会在导入外部样式单时导致"屏闪"。

5.1.3 使用内部CSS样式

一般来说不建议使用内部css样式，主要有三大缺点:

1. 如果此CSS样式需要被其他HTML文档使用，那么这些CSS样式必须在其他HTML文档中重复定义。
2. 大量CSS嵌套在HTML文档中，会导致HTML文档过大，大量的重复下载，导致网络负载加重。
3. 如果需要修改网页风格时，必须依次打开每个页面进行重复修改，不利于项目管理。

但内部样式也不是一无是处，如果想让某些css样式仅对某个页面有效而不会影响整个网站，则可以选择使用内部css样式定义。

内部css样式需要放在<style../>元素中定义，每个CSS样式定义与外部CSS样式文件定义的内容完全相同。<style../>元素应该放在<head../>元素内，作为它的子元素。使用内部css样式定义的语法如下:

```
<style type="text/css"
    样式单文件定义
</style>
```

范例:使用内部css样式

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>内部样式单</title>
    <style type="text/css">
        table {
    background-color:#003366;
    width: 400px;
    }

    td {
    background-color:#fff;
    font-family:"楷体_GB2312";
    font-size:20px;
    text-shadow:-8px 6px 2px #333;
    }
    .title {
        font-size: 18px;
        color: #60C;
        height: 30px;
        width: 200px;
        border-top: 3px solid #CCCCCC;
        border-left: 3px solid #CCCCCC;
        border-bottom: 3px solid #000000;
        border-right: 3px solid #000000;
    }
    </style>
</head>
<body>
```

```
    <div class="title">
        前端课程体系：
    </div><hr />
    <table>
        <tr>
            <td>HTML从入门到精通</td>
        </tr>
        <tr>
            <td>CSS从入门到精通</td>
        </tr>
        <tr>
            <td>JavaScript从入门到精通</td>
        </tr>
    </table>
</body>
</html>
```

5.1.4 使用内联样式

内联CSS样式只对单个标签有效，它甚至不会影响整个文件。内联样式定义可以精确控制某个HTML元素的外观表现，并且允许通过JavaScript动态修改HTML元素的CSS样式，从而改变该元素的外观。

为了使用内联样式，CSS扩展了HTML元素，几乎所有的HTML元素都增加了一个style通用属性，该属性值是一个或者多个CSS样式定义，多个CSS样式定义之间以英文分号隔开。

定义内联CSS样式的语法格式如下:

```
style="property1:value1;property2:value2..."
```

范例:使用内联样式

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>内联样式</title>
</head>
<body>
        <div style="font-size: 18px;
                color: #60C;
                height: 30px;
                width: 200px;
                border-top: 3px solid #CCCCCC;
                border-left: 3px solid #CCCCCC;
                border-bottom: 3px solid #000000;
                border-right: 3px solid #000000;">
                前端课程体系：
        </div><hr />
        <table style="background-color:#003366;
```

```
        width: 400px;">
                <tr>
                        <td style="     background-color:#fff;
        font-family:'楷体_GB2312';
        font-size:20px;
        text-shadow:-8px 6px 2px #333;">HTML从入门到精通</td>
                </tr>
                <tr>
                        <td style="     background-color:#fff;
        font-family:'楷体_GB2312';
        font-size:20px;
        text-shadow:-8px 6px 2px #333;">CSS从入门到精通</td>
                </tr>
                <tr>
                        <td style="     background-color:#fff;
        font-family:'楷体_GB2312';
        font-size:20px;
        text-shadow:-8px 6px 2px #333;">JavaScript从入门到精通</td>
                </tr>
        </table>
</body>
</html>
```

5.2 CSS选择器

正如上一节看到的，除了内联样式之外，定义CSS样式的语法总遵守如下格式:

```
Selector {
    property1: value1;
    property2: value2;
}
```

上面语法格式可分为两个部分。

- Selector(选择器) : 选择器决定该样式定义对哪些元素起作用。
- {property1:value1;property2:value2}(属性定义) : 属性定义部分决定这些样式起怎样的作用(字体、颜色、布局等)

下面我们来看CSS常用的一些选择器

5.2.1 元素选择器

元素选择器是最简单的选择器，其语法格式如下 :

```
E { ... } /* 其中E代表有效的HTML元素名 */
```

范例:使用元素选择器

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>元素选择器</title>
    <style type="text/css">
        /* 定义对div元素起作用的CSS样式 */
        div {
            background-color: grey;
            font: italic normal bold 14pt normal 楷体_GB2312;
        }
          /* 定义对p元素起作用的CSS样式 */
        p {
            background-color: #444;
            color: #fff;
            font: normal small-caps bold 20pt normal 宋体;
        }
    </style>
</head>
<body>
    <div>div 内的文字</div>
    <p>p内的文字</p>
</body>
</html>
```

5.2.2 属性选择器

其实元素选择器是属性选择器的特例。属性选择器一共有如下几种语法格式:

```
E {...}: 指定该CSS样式对所有E元素起作用
E[attr] {...}: 指定该CSS样式对具有attr属性的E元素起作用
E[attr=value]{...}: 指定该CSS样式对所有包含attr属性，且attr属性为value的E元素起作用
E[attr~=value]{...}: 指定该CSS样式对所有包含attr属性，且attr属性的值为以空格隔开的系列值，其中某个值为value的E元素起作用
E[attr|=value]{...}: 指定该CSS样式对所有包含attr属性，且attr属性的值为以连字符分隔的系列值，其中第一个值为value的Tag元素起作用
E[attr^="value"]{...}: 指定该CSS样式对所有包含attr属性，且attr属性的值为以value开头的字符串的E元素起作用(css3)
E[attr$="value"]{...}: 指定该CSS样式对所有包含attr属性，且attr属性的值为以value结尾的字符串的E元素起作用(css3)
E[attr*="value"]{...}: 指定该CSS样式对所有包含attr属性，且attr属性的值为所有包含value的字符串的E元素起作用(css3)
```

上述这几个选择器的匹配规则越严格，优先级越高。例如，对于包含abc属性的<div../>元素，如果其属性值为xyz,则div[abc=xyz]选择器定义的CSS样式会覆盖div[abc]定义的CSS样式。

范例:使用属性选择器

```
<!DOCTYPE html>
```

```
<html>
<head>
    <meta charset="utf-8">
    <title>属性选择器</title>
    <style type="text/css">
        div {
            width: 300px;
            height: 30px;
            background-color: #eee;
            border: 1px solid black;
            padding: 10px;
        }
        /* 对所有id属性的div元素起作用 */
        div[id] {
            background-color: #aaa;
        }
        /* 对所有id属性值包含xx的div元素起作用 */
        div[id*=xx] {
            background-color: #999;
        }
        /* 对所有id属性值以xx开头的div元素起作用 */
        div[id^=xx] {
            background-color: #555;
        }
        /* 对所有id属性值等于xx的div元素起作用  */
        div[id=xx] {
            background-color: #111;
            color: #fff;
        }
    </style>
</head>
<body>
    <div>没有任何属性的div元素</div>
    <div id="a">带id属性的div元素</div>
    <div id="zzxx">id属性值包含xx子字符串的div元素</div>
    <div id="xxyy">id属性值以xx开头的div元素</div>
    <div id="xx">id属性值等于xx的div元素</div>
</body>
</html>
```

上述定义的5个选择器的匹配规则依次升高，因此它们的优先级也是依次升高的。优先级越高的选择器对应的CSS样式的背景色越深。

从效果上我们能看到，虽然div[id=xx]选择器定义的CSS样式并没有定义长、宽。但该<div id="xx"../>元素依然具有指定高度、宽度，这表明该<div../>元素的显示外观是多个CSS样式"迭加"作用的效果。

当做个CSS样式定义都可以对某个HTML元素起作用时，该HTML元素的显示外观将是多个CSS样式定义"迭加"作用的效果。如果多个CSS样式定义之间有冲突时，则冲突属性以优先级更高的CSS样式取胜。

5.2.3 ID选择器

ID选择器指定CSS样式将会对具有指定id属性值的HTML元素起作用。ID选择器的语法格式如下:

```
#idValue {...}
```

上面语法指定该CSS样式对id为idValue的HTML元素起作用。

各种浏览器对ID选择器都有很好的支持。

范例:使用ID选择器

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>属性选择器</title>
        <style type="text/css">
                div {
                        width: 200px;
                        height: 30px;
                        background-color: #ddd;
                        padding: 3px;
                }
                #xx {
                        border:2px dotted black;
                        background-color: #888;
                }
        </style>
</head>
<body>
        <div>没有任何属性的div元素</div>
        <div id="xx">id属性值为xx的div元素</div>
</body>
</html>
```

仅对指定元素起作用的ID选择器

定义仅对指定元素起作用的ID选择器的语法格式如下:

```
E#idValue {...}
```

该语法指定该CSS样式对id为idValue的E元素起作用。

范例:仅对指定元素起作用的ID选择器

```
<!DOCTYPE html>
<html>
<head>
```

```
        <meta charset="utf-8">
        <title>属性选择器</title>
        <style type="text/css">
                div {
                        width: 200px;
                        height: 30px;
                        background-color: #ddd;
                        padding: 3px;
                }
                p#xx {
                        border: 2px dotted black;
                        background-color: #888;
                }
        </style>
</head>
<body>
        <div>没有任何属性的div元素</div>
        <div id="xx">id属性值为xx的div元素</div>
        <p id="xx">带有id的p标签</p>
</body>
</html>
```

5.2.4 class选择器

class选择器指定CSS样式对具有指定class属性的元素起作用。class选择器的语法格式如下:

```
[E].classValue {...} /* 其中E是有效的HTML元素*/
```

指定该CSS定义对class属性值为classValue的E元素起作用。此处的E可以忽略，如果省略E，则指定CSS对所有的class属性为classValue的元素都起作用。

为了让HTML页面支持class选择器，W3C规定几乎所有的HTML元素都可指定class属性，该属性唯一的作用正是让class选择器起作用。

范例:使用class选择器

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>属性选择器</title>
        <style type="text/css">
                .myclass {
                        width: 240px;
                        height: 40px;
                        background-color: #dddddd;
                }
                div.myclass {
                        border:2px dotted black;
```

```
                        background-color: #888888;
                }
        </style>
</head>
<body>
        <p class="myclass">class属性为myclass的P元素</p>
        <div class="myclass">class属性为myclass的div元素</div>
</body>
</html>
```

上面页面中定义的两个CSS都可作用于<div.../>元素，因此该元素显示的效果是两个CSS样式"迭加"的效果。既指定标签又指定class值的选择器的优先级更高。

5.2.5 包含选择器

包含选择器用于指定目标选择器必须处于某个选择器对应的元素内部。其语法格式如下:

```
Selector1 Selector2 {...} /* Selector1 Selector2都是有效的选择器
```

范例:使用包含选择器

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>属性选择器</title>
        <style type="text/css">
                div {
                        width: 350px;
                        height: 60px;
                        background-color: #ddd;
                        margin: 5px;
                }
                div .a {
                        width: 200px;
                        height: 35px;
                        border: 2px dotted black;
                        background-color: #888;
                }
        </style>
</head>
<body>
        <div>没有任何属性的div元素</div>
        <div>
                <p class="a">处于div内部且class属性为a的元素</p>
        </div>
        <p class="a">没有处于div之内，但class属性为a的元素</p>
</body>
```

```
</html>
```

5.2.6 子选择器

子选择器用于指定目标选择器必须是某个选择器对应的元素的子元素。子选择器的语法如下:

```
Selector1>Selector2 {...} /* 其中Selector1、Selector2都是有效的选择器 */
```

包含选择器与子选择器有些类似，他们之间存在以下区别:对于包含选择器，只要目标选择器位于外部选择器对应的元素内部，即使是"孙子元素"也可；而对于子选择器，要求目标选择器必须作为外部选择器对应元素的直接子元素才可。

范例:使用子选择器

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>属性选择器</title>
        <style type="text/css">
                div {
                        width: 350px;
                        height: 60px;
                        background-color: #ddd;
                        margin: 5px;
                }
                div>.a {
                        width: 200px;
                        height: 35px;
                        border: 2px dotted black;
                        background-color: #888;
                }
        </style>
</head>
<body>
        <div>没有任何属性的div元素</div>
        <div>
                <p class="a">处于div内部且class属性为a的元素，并且为直接子元素</p>
        </div>
        <div>
                <section><p class="a">处于div内部且class属性为a的元素，并且为孙子元素
</p></section>
        </div>
</body>
</html>
```

5.3 CSS样式

5.3.1 CSS背景

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果。

CSS 在这方面的能力远远在 HTML 之上。

背景色

可以使用 background-color 属性为元素设置背景色。这个属性接受任何合法的颜色值。

这条规则把元素的背景设置为灰色：

```
p {background-color: gray;}
```

如果希望背景色从元素中的文本向外少有延伸，只需增加一些内边距：

```
p {background-color: gray; padding: 20px;}
```

可以为所有元素设置背景色，这包括 body 一直到 em 和 a 等行内元素。

background-color 不能继承，其默认值是 transparent。transparent 有“透明”之意。也就是说，如果一个元素没有指定背景色，那么背景就是透明的，这样其祖先元素的背景才能可见。

范例:CSS设置背景色

```
<html>
<head>
<style type="text/css">
body {background-color: yellow}
h1 {background-color: #00ff00}
h2 {background-color: transparent}
p {background-color: rgb(250,0,255)}
p.no2 {
                background-color: gray;
                padding: 20px;
        }
</style>
</head>
<body>
<h1>标题 1</h1>
<h2>标题 2</h2>
<p>段落</p>
<p class="no2">这个段落设置了内边距。</p>
</body>
</html>
```

背景图像

要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 URL 值：

```
body {background-image: url(/i/eg_bg_04.gif);}
```

大多数背景都应用到 body 元素，不过并不仅限于此。

下面例子为一个段落应用了一个背景，而不会对文档的其他部分应用背景：

```
p.flower {background-image: url(/i/eg_bg_03.gif);}
```

甚至可以为行内元素设置背景图像，下面的例子为一个链接设置了背景图像：

```
a.radio {background-image: url(/i/eg_bg_07.gif);}
```

理论上讲，甚至可以向 textareas 和 select 等替换元素的背景应用图像，不过并不是所有用户代理都能很好地处理这种情况。

另外还要补充一点，background-image 也不能继承。事实上，所有背景属性都不能继承。

范例:使用背景图像

```
<html>
<head>

<style type="text/css">
body {background-image:url(04.gif);}
p.flower {background-image: url(03.gif); padding: 20px;}
a.radio {background-image: url(07.gif);  padding: 20px;}
</style>

</head>

<body>
<p class="flower">我是一个有花纹背景的段落。<a href="#" class="radio">我是一个有放射性背景的链接。</a></p>
<p><b>注释：</b>为了清晰地显示出段落和链接的背景图像，我们为它们设置了少许内边距。</p>
</body>

</html>
```

背景重复

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性

属性值 repeat 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺。

默认地，背景图像将从一个元素的左上角开始。请看下面的例子：

```
<html>
<head>
<style type="text/css">
body
{
background-image:
url(03.gif);
background-repeat: repeat-y
}
</style>
</head>
<body>
</body>
</html>
```

背景定位

可以利用 background-position 属性改变图像在背景中的位置。

下面的例子在 body 元素中将一个背景图像居中放置：

```
body
  {
    background-image:url('03.gif');
    background-repeat:no-repeat;
    background-position:center;
  }
```

为 background-position 属性提供值有很多方法。首先，可以使用一些关键字：top、bottom、left、right 和 center。通常，这些关键字会成对出现，不过也不总是这样。还可以使用长度值，如 100px 或 5cm，最后也可以使用百分数值。不同类型的值对于背景图像的放置稍有差异。

关键字

图像放置关键字最容易理解，其作用如其名称所表明的。例如，top right 使图像放置在元素内边距区的右上角。

根据规范，位置关键字可以按任何顺序出现，只要保证不超过两个关键字 - 一个对应水平方向，另一个对应垂直方向。

如果只出现一个关键字，则认为另一个关键字是 center。

所以，如果希望每个段落的中部上方出现一个图像，只需声明如下：

```
<html>
<head>
<style type="text/css">
p
  {
    background-image:url('03.gif');
```

```
    background-repeat:no-repeat;
    background-position:top;
  }
</style>
</head>
<body>
        <p>段落1</p>
        <p>段落2</p>
</body>
</html>
```

下面是等价的位置关键字：

单一关键字	等价的关键字
center	center center
top	top center 或 center top
bottom	bottom center 或 center bottom
right	right center 或 center right
left	left center 或 center left

百分数值

百分数值的表现方式更为复杂。假设你希望用百分数值将图像在其元素中居中，这很容易：

```
body
  {
    background-image:url('03.gif');
    background-repeat:no-repeat;
    background-position:50% 50%;
  }
```

这会导致图像适当放置，其中心与其元素的中心对齐。**换句话说，百分数值同时应用于元素和图像。**也就是说，图像中描述为 50% 50% 的点（中心点）与元素中描述为 50% 50% 的点（中心点）对齐。

如果图像位于 0% 0%，其左上角将放在元素内边距区的左上角。如果图像位置是 100% 100%，会使图像的右下角放在右边距的右下角。

因此，如果你想把一个图像放在水平方向 2/3、垂直方向 1/3 处，可以这样声明：

```
body
  {
    background-image:url('03.gif');
    background-repeat:no-repeat;
    background-position:66% 33%;
  }
```

如果只提供一个百分数值，所提供的这个值将用作水平值，垂直值将假设为 50%。这一点与关键字类似。

background-position 的默认值是 0% 0%，在功能上相当于 top left。这就解释了背景图像为什么总是从元素内边距区的左上角开始平铺，除非设置了不同的位置值。

长度值

长度值解释的是元素内边距区左上角的偏移。偏移点是图像的左上角。

比如，如果设置值为 50px 100px，图像的左上角将在元素内边距区左上角向右 50 像素、向下 100 像素的位置上：

```
<html>
<head>
<style type="text/css">
body
  {
    background-image:url('03.gif');
    background-repeat:no-repeat;
    background-position:50px 100px;
  }
</style>
</head>
<body>
</body>
</html>
```

注意，这一点与百分数值不同，因为偏移只是从一个左上角到另一个左上角。也就是说，图像的左上角与 background-position 声明中的指定的点对齐。

背景关联

如果文档比较长，那么当文档向下滚动时，背景图像也会随之滚动。当文档滚动到超过图像的位置时，图像就会消失。

可以通过 background-attachment 属性防止这种滚动。通过这个属性，可以声明图像相对于可视区是固定的（fixed），因此不会受到滚动的影响：

```
<html>
<head>
<style type="text/css">
```

```
body
{
background-image:url(03.gif);
background-repeat:no-repeat;
background-attachment:fixed
}
</style>
</head>
<body>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
<p>图像不会随页面的其余部分滚动。</p>
</body>
</html>
```

background-attachment 属性的默认值是 scroll，也就是说，在默认的情况下，背景会随文档滚动。

CSS 背景属性

属性	描述
background	简写属性，作用是将背景属性设置在一个声明中。
background-attachment	背景图像是否固定或者随着页面的其余部分滚动。
background-color	设置元素的背景颜色。
background-image	把图像设置为背景。
background-position	设置背景图像的起始位置。
background-repeat	设置背景图像是否及如何重复。

5.3.2 CSS文本

CSS 文本属性可定义文本的外观。

通过文本属性，可以改变文本的颜色、字符间距，对齐文本，装饰文本，对文本进行缩进，等等。

缩进文本

把 Web 页面上的段落的第一行缩进，这是一种最常用的文本格式化效果。

CSS 提供了 text-indent 属性，该属性可以方便地实现文本缩进。

通过使用 text-indent 属性，所有元素的第一行都可以缩进一个给定的长度，甚至该长度可以是负值。

这个属性最常见的用途是将段落的首行缩进，下面的规则会使所有段落的首行缩进 5 em：

```
p {text-indent: 5em;}
```

注意：一般来说，可以为所有块级元素应用 text-indent，但无法将该属性应用于行内元素，图像之类的替换元素上也无法应用 text-indent 属性。不过，如果一个块级元素（比如段落）的首行中有一个图像，它会随该行的其余文本移动。

提示：如果想把一个行内元素的第一行“缩进”，可以用左内边距或外边距创造这种效果。

使用负值

text-indent 还可以设置为负值。利用这种技术，可以实现很多有趣的效果，比如“悬挂缩进”，即第一行悬挂在元素中余下部分的左边：

```
p {text-indent: -5em;}
```

不过在为 text-indent 设置负值时要当心，如果对一个段落设置了负值，那么首行的某些文本可能会超出浏览器窗口的左边界。为了避免出现这种显示问题，建议针对负缩进再设置一个外边距或一些内边距：

```
p {text-indent: -5em; padding-left: 5em;}
```

使用百分比值

text-indent 可以使用所有长度单位，包括百分比值。

百分数要相对于缩进元素父元素的宽度。换句话说，如果将缩进值设置为 20%，所影响元素的第一行会缩进其父元素宽度的 20%。

在下例中，缩进值是父元素的 20%，即 100 个像素：

```
div {width: 500px;}
p {text-indent: 20%;}

<div>
<p>this is a paragragh</p>
</div>
```

继承

text-indent 属性可以继承，比如如下标记：

```
div#outer {width: 500px;}
div#inner {text-indent: 10%;}
p {width: 200px;}

<div id="outer">
<div id="inner">some text. some text. some text.
<p>this is a paragragh.</p>
</div>
</div>
```

以上标记中的段落也会缩进 50 像素，这是因为这个段落继承了 id 为 inner 的 div 元素的缩进值。

水平对齐

text-align是一个基本的属性，它会影响一个元素中的*文本行*互相之间的对齐方式。它的前 3 个值相当直接，不过第 4 个和第 5 个则略有些复杂。

值 left、right 和 center 会导致元素中的文本分别左对齐、右对齐和居中。

西方语言都是从左向右读，所有 text-align 的默认值是 left。文本在左边界对齐，右边界呈锯齿状（称为“从左到右”文本）。对于希伯来语和阿拉伯语之类的的语言，text-align 则默认为 right，因为这些语言从右向左读。不出所料，center 会使每个文本行在元素中居中。

提示：将块级元素或表元素居中，要通过在这些元素上适当地设置左、右外边距来实现。

范例:使用text-align进行水平对齐

```
<html>
<head>
        <style>
                p#id {
                        text-align: center
                }
        </style>
</head>
<body>
        <p>段落1</p>
        <p id="id">段落2</p>
</body>
</html>
```

text-align:center 与 <CENTER>

大家可能会认为 text-align:center 与 <CENTER> 元素的作用一样，但实际上二者大不相同。

<CENTER> 不仅影响文本，还会把整个元素居中。text-align 不会控制元素的对齐，而只影响内部内容。元素本身不会从一段移到另一端，只是其中的文本受影响。

字间隔

word-spacing 属性可以改变字（单词）之间的标准间隔。其默认值 normal 与设置值为 0 是一样的。

word-spacing 属性接受一个正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把它拉近：

范例：设置字间隔

```
<html>
<head>
        <style type="text/css">
                p.spread {
                        word-spacing: 30px;
                }

                p.tight {
                        word-spacing: -0.5em;
                }
        </style>
</head>
<body>
        <p class="spread">This is some text. This is some text.</p>
        <p class="tight">This is some text. This is some text.</p>
</body>
</html>
```

字母间隔

letter-spacing 属性与 word-spacing 的区别在于，字母间隔修改的是字符或字母之间的间隔。

与 word-spacing 属性一样，letter-spacing 属性的可取值包括所有长度。默认关键字是 normal（这与 letter-spacing:0 相同）。输入的长度值会使字母之间的间隔增加或减少指定的量：

范例:设置字母间隔

```
<html>
<head>
        <style type="text/css">
                h1 {
                        letter-spacing: -0.5em
                }
                h4 {
                        letter-spacing: 20px
                }
        </style>
</head>
<body>
        <h1>This is header 1</h1>
        <h4>This is header 4</h4>
</body>
</html>
```

字符转换

text-transform 属性处理文本的大小写。这个属性有 4 个值：

- none
- uppercase
- lowercase
- capitalize

默认值 none 对文本不做任何改动，将使用源文档中的原有大小写。顾名思义，uppercase 和 lowercase 将文本转换为全大写和全小写字符。最后，capitalize 只对每个单词的首字母大写。

作为一个属性，text-transform 可能无关紧要，不过如果您突然决定把所有 h1 元素变为大写，这个属性就很有用。不必单独地修改所有 h1 元素的内容，只需使用 text-transform 为你完成这个修改：

```
<html>
<head>
        <style type="text/css">
                h1 {
                        text-transform: uppercase
                }
        </style>
</head>

<body>
        <h1>This is header 1</h1>
        <h4>This is header 4</h4>
</body>
```

```
</html>
```

使用 text-transform 有两方面的好处。首先，只需写一个简单的规则来完成这个修改，而无需修改 h1 元素本身。其次，如果以后决定将所有大小写再切换为原来的大小写，可以更容易地完成修改。

文本装饰

接下来，我们 text-decoration 属性，这是一个很有意思的属性，它提供了很多非常有趣的行为。

text-decoration 有 5 个值：

- none
- underline
- overline
- line-through
- blink

underline 会对元素加下划线，就像 HTML 中的 U 元素一样。overline 的作用恰好相反，会在文本的顶端画一个上划线。值 line-through 则在文本中间画一个贯穿线，等价于 HTML 中的 S 和 strike 元素。blink 会让文本闪烁，类似于 Netscape 支持的颇招非议的 blink 标记。

none 值会关闭原本应用到一个元素上的所有装饰。通常，无装饰的文本是默认外观，但也不总是这样。例如，链接默认地会有下划线。如果希望去掉超链接的下划线，可以使用以下 CSS 来做到这一点：

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

注意：如果显式地用这样一个规则去掉链接的下划线，那么锚与正常文本之间在视觉上的唯一差别就是颜色（至少默认是这样的，不过也不能完全保证其颜色肯定有区别）。

还可以在一个规则中结合多种装饰。如果希望所有超链接既有下划线，又有上划线，则规则如下：

```
a:link a:visited {text-decoration: underline overline;}
```

不过要注意的是，如果两个不同的装饰都与同一元素匹配，胜出规则的值会完全取代另一个值。请考虑以下的规则：

```
h2.stricken {text-decoration: line-through;}
h2 {text-decoration: underline overline;}
```

对于给定的规则，所有 class 为 stricken 的 h2 元素都只有一个贯穿线装饰，而没有下划线和上划线，因为 *text-decoration 值会替换而不是累积起来*。

处理空白符

white-space 属性会影响到用户代理对源文档中的空格、换行和 tab 字符的处理。

通过使用该属性，可以影响浏览器处理字之间和文本行之间的空白符的方式。从某种程度上讲，默认的 HTML 处理已经完成了空白符处理：它会把所有空白符合并为一个空格。所以给定以下标记，它在 Web 浏览器中显示时，各个字之间只会显示一个空格，同时忽略元素中的换行：

```
<p>This     paragraph has    many
    spaces           in it.</p>
```

可以用以下声明显式地设置这种默认行为：

```
p {white-space: normal;}
```

上面的规则告诉浏览器按照平常的做法去处理：丢掉多余的空白符。如果给定这个值，换行字符（回车）会转换为空格，一行中多个空格的序列也会转换为一个空格。

下面的表格总结了 white-space 属性的行为：

值	空白符	换行符	自动换行
pre-line	合并	保留	允许
normal	合并	忽略	允许
nowrap	合并	忽略	不允许
pre	保留	保留	不允许
pre-wrap	保留	保留	允许

5.3.3 CSS字体

CSS 字体属性定义文本的字体系列、大小、加粗、风格（如斜体）和变形（如小型大写字母）。

CSS 字体系列

在 CSS 中，有两种不同类型的字体系列名称：

- 通用字体系列 - 拥有相似外观的字体系统组合（比如 "Serif" 或 "Monospace"）
- 特定字体系列 - 具体的字体系列（比如 "Times" 或 "Courier"）

除了各种特定的字体系列外，CSS 定义了 5 种通用字体系列：

- Serif 字体
- Sans-serif 字体
- Monospace 字体
- Cursive 字体
- Fantasy 字体

指定字体系列

使用 font-family 属性 定义文本的字体系列。

使用通用字体系列

如果你希望文档使用一种 sans-serif 字体，但是你并不关心是哪一种字体，以下就是一个合适的声明：

```
body {font-family: sans-serif;}
```

样用户代理就会从 sans-serif 字体系列中选择一个字体（如 Helvetica），并将其应用到 body 元素。因为有继承，这种字体选择还将应用到 body 元素中包含的所有元素，除非有一种更特定的选择器将其覆盖。

指定字体系列

除了使用通用的字体系列，还可以通过 font-family 属性设置更具体的字体。

下面的例子为所有 h1 元素设置了 Georgia 字体：

```
<html>
<head>
<style type="text/css">
h1 {font-family:Georgia;}
</style>
</head>
<body>
<h1>This is heading 1</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

这样的规则同时会产生另外一个问题，如果用户代理上没有安装 Georgia 字体，就只能使用用户代理的默认字体来显示 h1 元素。

我们可以通过结合特定字体名和通用字体系列来解决这个问题：

```
h1 {font-family: Georgia, serif;}
```

如果读者没有安装 Georgia，但安装了 Times 字体（serif 字体系列中的一种字体），用户代理就可能对 h1 元素使用 Times。尽管 Times 与 Georgia 并不完全匹配，但至少足够接近。

因此，我们建议在所有 font-family 规则中都提供一个通用字体系列。这样就提供了一条后路，在用户代理无法提供与规则匹配的特定字体时，就可以选择一个候选字体。

如果对字体非常熟悉，也可以为给定的元素指定一系列类似的字体。要做到这一点，需要把这些字体按照优先顺序排列，然后用逗号进行连接：

```
p {font-family: Times, TimesNR, 'New Century Schoolbook',
     Georgia, 'New York', serif;}
```

根据这个列表，用户代理会按所列的顺序查找这些字体。如果列出的所有字体都不可用，就会简单地选择一种可用的 serif 字体。

使用引号

大家也许已经注意到了，上面的例子中使用了单引号。只有当字体名中有一个或多个空格（比如 New York），或者如果字体名包括 # 或 $ 之类的符号，才需要在 font-family 声明中加引号。

单引号或双引号都可以接受。但是，如果把一个 font-family 属性放在 HTML 的 style 属性中，则需要使用该属性本身未使用的那种引号：

```
<p style="font-family: Times, TimesNR, 'New Century Schoolbook', Georgia,
 'New York', serif;">...</p>
```

字体风格

font-style 属性最常用于规定斜体文本。

该属性有三个值：

- normal - 文本正常显示
- italic - 文本斜体显示
- oblique - 文本倾斜显示

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

italic 和 oblique 的区别

font-style 非常简单：用于在 normal 文本、italic 文本和 oblique 文本之间选择。唯一有点复杂的是明确 italic 文本和 oblique 文本之间的差别。

斜体（italic）是一种简单的字体风格，对每个字母的结构有一些小改动，来反映变化的外观。与此不同，倾斜（oblique）文本则是正常竖直文本的一个倾斜版本。

通常情况下，italic 和 oblique 文本在 web 浏览器中看上去完全一样。

字体变形

font-variant 属性可以设定小型大写字母。

小型大写字母不是一般的大写字母，也不是小写字母，这种字母采用不同大小的大写字母

```
p {font-variant:small-caps;}
```

字体加粗

font-weight 属性设置文本的粗细。

使用 bold 关键字可以将文本设置为粗体。

关键字 100 ~ 900 为字体指定了 9 级加粗度。如果一个字体内置了这些加粗级别，那么这些数字就直接映射到预定义的级别，100 对应最细的字体变形，900 对应最粗的字体变形。数字 400 等价于 normal，而 700 等价于 bold。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度下移而不是上移。

```
p.normal {font-weight:normal;}
p.thick {font-weight:bold;}
p.thicker {font-weight:900;}
```

字体大小

font-size 属性设置文本的大小。

有能力管理文本的大小在 web 设计领域很重要。但是，不应当通过调整文本大小使段落看上去像标题，或者使标题看上去像段落。

请始终使用正确的 HTML 标题，比如使用<h1> - <h6> 来标记标题，使用 <p> 来标记段落。

font-size 值可以是绝对或相对值。

绝对值：

将文本设置为指定的大小 不允许用户在所有浏览器中改变文本大小（不利于可用性） 绝对大小在确定了输出的物理尺寸时很有用 相对大小：

相对于周围的元素来设置大小 允许用户在浏览器改变文本大小 注意：如果没有规定字体大小，普通文本（比如段落）的默认大小是 16 像素 (16px=1em)。

使用像素来设置字体大小 通过像素设置文本大小，可以对文本大小进行完全控制：

```
h1 {font-size:60px;}
h2 {font-size:40px;}
p {font-size:14px;}
```

使用 em 来设置字体大小 如果要避免在 Internet Explorer 中无法调整文本的问题，许多开发者使用 em 单位代替 pixels。

W3C 推荐使用 em 尺寸单位。

1em 等于当前的字体尺寸。如果一个元素的 font-size 为 16 像素，那么对于该元素，1em 就等于 16 像素。在设置字体大小时，em 的值会相对于父元素的字体大小改变。

浏览器中默认的文本大小是 16 像素。因此 1em 的默认尺寸是 16 像素。

可以使用下面这个公式将像素转换为 em：pixels/16=em

（注：16 等于父元素的默认字体大小，假设父元素的 font-size 为 20px，那么公式需改为：pixels/20=em）

```
h1 {font-size:3.75em;} /* 60px/16=3.75em */
h2 {font-size:2.5em;}  /* 40px/16=2.5em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

结合使用百分比和 EM

在所有浏览器中均有效的方案是为 body 元素（父元素）以百分比设置默认的 font-size 值：

```
body {font-size:100%;}
h1 {font-size:3.75em;}
h2 {font-size:2.5em;}
p {font-size:0.875em;}
```

在所有浏览器中，可以显示相同的文本大小，并允许所有浏览器缩放文本的大小。

CSS 字体属性

属性	描述
font	简写属性。作用是把所有针对字体的属性设置在一个声明中。
font-family	设置字体系列。
font-size	设置字体的尺寸。
font-size-adjust	当首选字体不可用时，对替换字体进行智能缩放。（CSS2.1 已删除该属性。）
font-stretch	对字体进行水平拉伸。（CSS2.1 已删除该属性。）
font-style	设置字体风格。
font-variant	以小型大写字体或者正常字体显示文本。
font-weight	设置字体的粗细。

5.3.4 CSS链接

设置链接的样式

能够设置链接样式的 CSS 属性有很多种（例如 color, font-family, background 等等）。

链接的特殊性在于能够根据它们所处的状态来设置它们的样式。

链接的四种状态：

- a:link - 普通的、未被访问的链接
- a:visited - 用户已访问的链接
- a:hover - 鼠标指针位于链接的上方
- a:active - 链接被点击的时刻

范例：设置链接样式

```
<!DOCTYPE html>
<html>
<head>
<style>
        a:link {color:#FF0000;}    /* 未被访问的链接 */
        a:visited {color: yellow} /* 已被访问的链接 */
        a:hover {color:#FF00FF;}   /* 鼠标指针移动到链接上 */
        a:active {color:#0000FF;}  /* 正在被点击的链接 */
</style>
</head>
<body>
        <a href="#">链接标签</a>
</body>
```

当为链接的不同状态设置样式时，请按照以下次序规则：

- a:hover 必须位于 a:link 和 a:visited 之后
- a:active 必须位于 a:hover 之后

常见的链接样式

在上面的例子中，链接根据其状态改变颜色。

让我们看看其他几种常见的设置链接样式的方法：

文本修饰

text-decoration 属性大多用于去掉链接中的下划线：

范例:去掉链接下划线

```
<!DOCTYPE html>
<html>
<head>
<style>
        a:link {text-decoration:none;}    /* 未被访问的链接 */
        a:visited {text-decoration:none;} /* 已被访问的链接 */
        a:hover {text-decoration:underline;}   /* 鼠标指针移动到链接上 */
        a:active {text-decoration:underline;}  /* 正在被点击的链接 */
</style>
</head>
<body>
<p><b><a href="#" target="_blank">这是一个链接</a></b></p>
</body>
</html>
```

背景色

background-color 属性规定链接的背景色：

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
a:link {background-color:#B2FF99;}    /* 未被访问的链接 */
a:visited {background-color:#FFFF85;} /* 已被访问的链接 */
a:hover {background-color:#FF704D;}   /* 鼠标指针移动到链接上 */
a:active {background-color:#FF704D;}  /* 正在被点击的链接 */
</style>
</head>

<body>
<p><a href="#">链接1</a></p>
</body>
</html>
```

5.3.5 CSS列表

CSS 列表属性允许你放置、改变列表项标志，或者将图像作为列表项标志。

列表类型

要影响列表的样式，最简单（同时支持最充分）的办法就是改变其标志类型。

例如，在一个无序列表中，列表项的标志 (marker) 是出现在各列表项旁边的圆点。在有序列表中，标志可能是字母、数字或另外某种计数体系中的一个符号。

要修改用于列表项的标志类型，可以使用属性 list-style-type：

```
ul {list-style-type : square}
```

上面的声明把无序列表中的列表项标志设置为方块。

范例:设置列表标志类型

设置不同的列表样式：

```
ul.circle {list-style-type:circle;}
ul.square {list-style-type:square;}
ol.upper-roman {list-style-type:upper-roman;}
ol.lower-alpha {list-style-type:lower-alpha;}
```

列表项图像

有时，常规的标志是不够的。你可能想对各标志使用一个图像，这可以利用 list-style-image属性做到：

```
ul li {list-style-image : url(xxx.gif)}
```

只需要简单地使用一个 url() 值，就可以使用图像作为标志。

范例:将列表项变为图像

CSS 列表属性(list)

属性	描述
list-style	简写属性。用于把所有用于列表的属性设置于一个声明中。
list-style-image	将图象设置为列表项标志。
list-style-position	设置列表中列表项标志的位置。
list-style-type	设置列表项标志的类型。

5.3.6 CSS表格

表格边框

如需在 CSS 中设置表格边框，请使用 border 属性。

下面的例子为 table、th 以及 td 设置了蓝色边框：

```
table, th, td
  {
  border: 1px solid blue;
  }
```

范例:为表格设置蓝色边框

```
<html>
<head>
<style type="text/css">
table,th,td
{
border:1px solid blue;
}
</style>
</head>

<body>
<table>
<tr>
<th>Firstname</th>
<th>Lastname</th>
</tr>
<tr>
<td>刘</td>
<td>益铭</td>
</tr>
<tr>
```

```
<td>蛋</td>
<td>哥</td>
</tr>
</table>
</body>
</html>
```

折叠边框

border-collapse 属性设置是否将表格边框折叠为单一边框：

```
table
  {
  border-collapse:collapse;
  }

table,th, td
  {
  border: 1px solid blue;
  }
```

表格宽度和高度

通过 width 和 height 属性定义表格的宽度和高度。

下面的例子将表格宽度设置为 100%，同时将 th 元素的高度设置为 50px：

```
table
  {
  width:100%;
  }

th
  {
  height:50px;
  }
```

表格文本对齐

text-align 和 vertical-align 属性设置表格中文本的对齐方式。

text-align 属性设置水平对齐方式，比如左对齐、右对齐或者居中：

```
td
  {
  text-align:right;
  }
```

vertical-align 属性设置垂直对齐方式，比如顶部对齐、底部对齐或居中对齐：

```
td
  {
  height:50px;
  vertical-align:bottom;
  }
```

表格内边距

如需控制表格中内容与边框的距离，请为 td 和 th 元素设置 padding 属性：

```
td
  {
  padding:15px;
  }
```

表格颜色

下面的例子设置边框的颜色，以及 th 元素的文本和背景颜色：

```
table, td, th
  {
  border:1px solid blue;
  }

th
  {
  background-color:green;
  color:white;
  }
```

CSS Table 属性

属性	描述
border-collapse	设置是否把表格边框合并为单一的边框。
border-spacing	设置分隔单元格边框的距离。
caption-side	设置表格标题的位置。
empty-cells	设置是否显示表格中的空单元格。
table-layout	设置显示单元、行和列的算法。

综合案例:制作一个相对漂亮的表格

```
<html>
<head>
<style type="text/css">
```

```
#customers
  {
  font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
  width:100%;
  border-collapse:collapse;
  }

#customers td, #customers th
  {
  font-size:1em;
  border:1px solid #98bf21;
  padding:3px 7px 2px 7px;
  }

#customers th
  {
  font-size:1.1em;
  text-align:left;
  padding-top:5px;
  padding-bottom:4px;
  background-color:#A7C942;
  color:#ffffff;
  }

#customers tr.alt td
  {
  color:#000000;
  background-color:#EAF2D3;
  }
</style>
</head>

<body>
<table id="customers">
<tr>
<th>Company</th>
<th>Contact</th>
<th>Country</th>
</tr>
<tr>
<td>Apple</td>
<td>Steven Jobs</td>
<td>USA</td>
</tr>
<tr class="alt">
<td>Baidu</td>
<td>Li YanHong</td>
<td>China</td>
</tr>
```

```
<tr>
<td>Google</td>
<td>Larry Page</td>
<td>USA</td>
</tr>
<tr class="alt">
<td>Lenovo</td>
<td>Liu Chuanzhi</td>
<td>China</td>
</tr>
<tr>
<td>Microsoft</td>
<td>Bill Gates</td>
<td>USA</td>
</tr>
<tr class="alt">
<td>Nokia</td>
<td>Stephen Elop</td>
<td>Finland</td>
</tr>
</table>
</body>
</html>
```

5.3.7 CSS轮廓

轮廓（outline）是绘制于元素周围的一条线，位于边框边缘的外围，可起到突出元素的作用。

CSS outline 属性规定元素轮廓的样式、颜色和宽度。

CSS 边框属性

"CSS" 列中的数字指示哪个 CSS 版本定义了该属性。

属性	描述	CSS
outline	在一个声明中设置所有的轮廓属性。	2
outline-color	设置轮廓的颜色。	2
outline-style	设置轮廓的样式。	2
outline-width	设置轮廓的宽度。	2

范例:使用css轮廓为元素画线

```
<!DOCTYPE html>
<html>
```

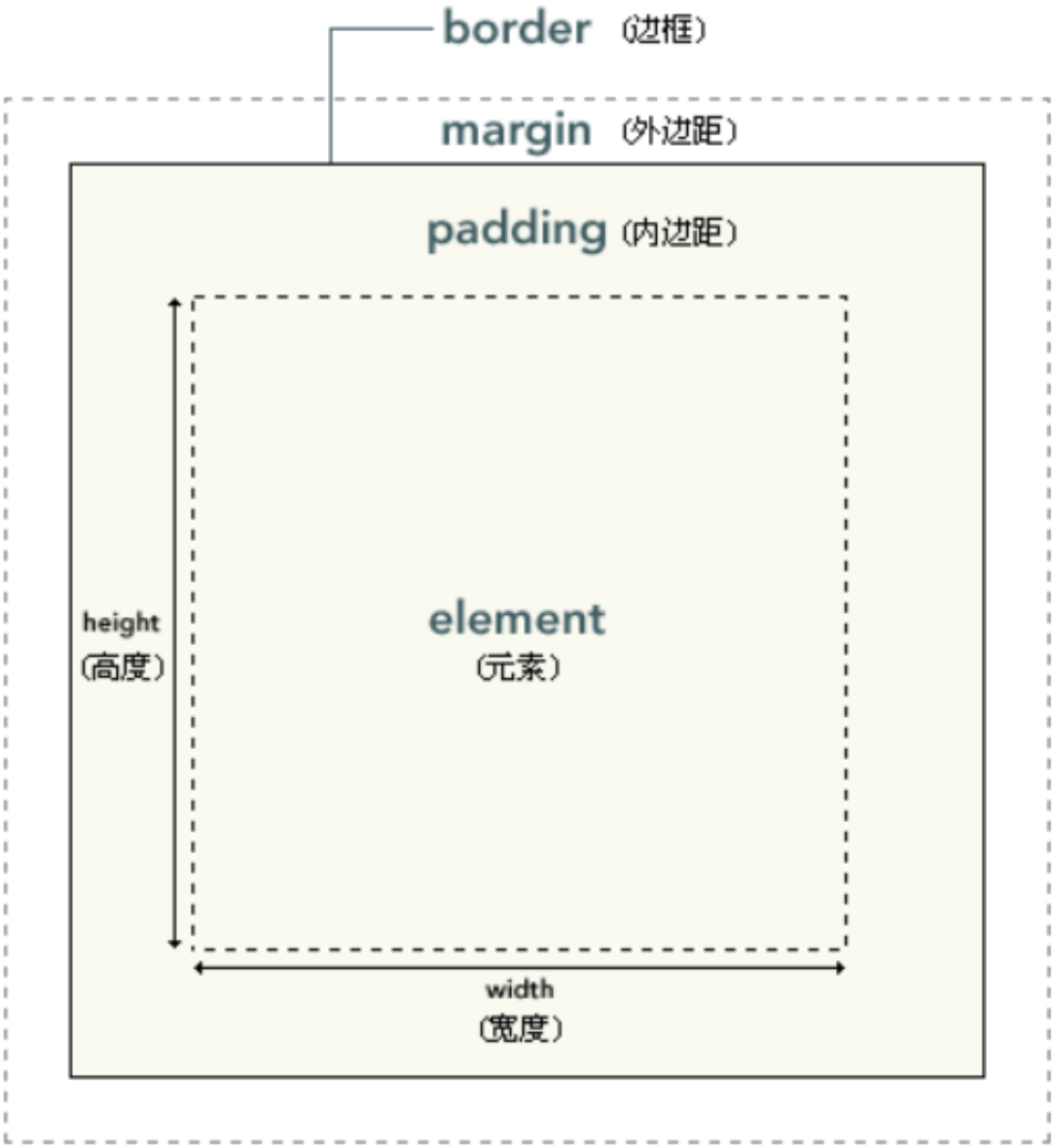
```
<head>
<style type="text/css">
p
{
border:black solid thin;
outline:blue dotted thick;
}
</style>
</head>

<body>
<p>段落</p>
</body>
</html>
```

六、CSS盒模型

CSS盒模型 (Box Model)规定了元素框处理元素内容、内边距、边框和外边距 的方式。

6.1盒模型概述



元素框的最内部分是实际的内容，直接包围内容的是内边距。内边距呈现了元素的背景。内边距的边缘是边框。边框以外是外边距，外边距默认是透明的，因此不会遮挡其后的任何元素。

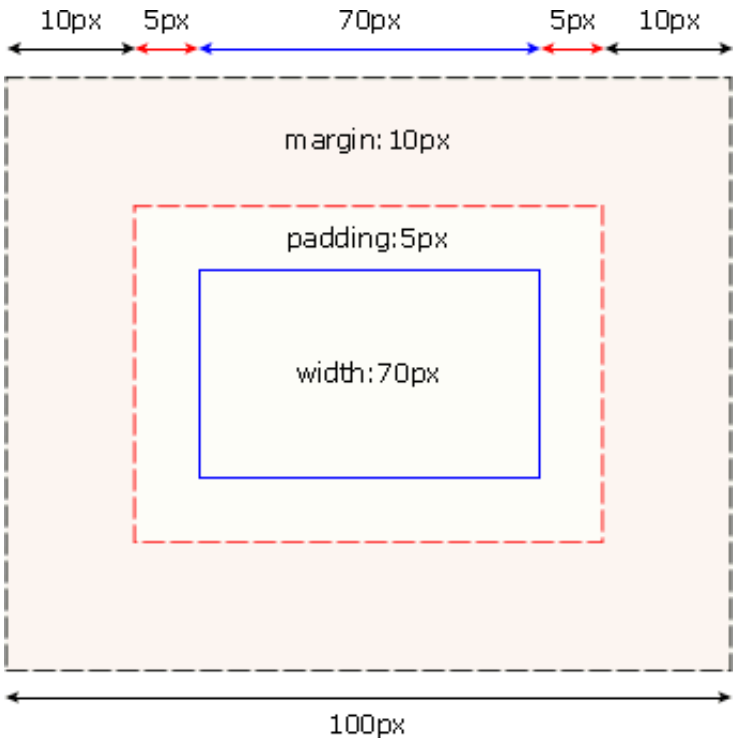
提示：背景应用于由内容和内边距、边框组成的区域。

内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器对所有元素进行设置：

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

在 CSS 中，width 和 height 指的是内容区域的宽度和高度。增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素框的总尺寸。

假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，请看下图：




```
#box {
  width: 70px;
  margin: 10px;
  padding: 5px;
}
```

我们把 padding 和 margin 统一地称为内边距和外边距。边框内的空白是内边距，边框外的空白是外边距。

6.2 CSS内边距

元素的内边距在边框和内容区之间。控制该区域最简单的属性是 padding 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性

CSS padding 属性定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。

例如，如果希望所有 h1 元素的各边都有 10 像素的内边距，只需要这样：

```
h1 {padding: 10px;}
```

还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
h1 {padding: 10px 0.25em 2ex 20%;}
```

单边内边距属性

也通过使用下面四个单独的属性，分别设置上、右、下、左内边距：

- padding-top
- padding-right
- padding-bottom
- padding-left

下面的规则实现的效果与上面的简写规则是完全相同的：

```
h1 {
  padding-top: 10px;
  padding-right: 0.25em;
  padding-bottom: 2ex;
  padding-left: 20%;
  }
```

内边距的百分比数值

前面提到过，可以为元素的内边距设置百分数值。百分数值是相对于其父元素的 width 计算的，这一点与外边距一样。所以，如果父元素的 width 改变，它们也会改变。

下面这条规则把段落的内边距设置为父元素 width 的 10%：

```
p {padding: 10%;}
```

例如：如果一个段落的父元素是 div 元素，那么它的内边距要根据 div 的 width 计算。

```
<div style="width: 200px;">
<p>This paragragh is contained within a DIV that has a width of 200 pixels.</p>
</div>
```

注意：上下内边距与左右内边距一致；即上下内边距的百分数会相对于父元素宽度设置，而不是相对于高度。

CSS 内边距属性

属性	描述
padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
padding-bottom	设置元素的下内边距。
padding-left	设置元素的左内边距。
padding-right	设置元素的右内边距。
padding-top	设置元素的上内边距。

6.3 CSS边框

元素的边框 (border) 是围绕元素内容和内边距的一条或多条线。

CSS border 属性允许规定元素边框的样式、宽度和颜色。

CSS 边框

在 HTML 中，我们使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的的边框 (border)。元素的边框就是围绕元素内容和内边距的一条或多条线。

每个边框有 3 个方面：宽度、样式，以及颜色。在下面的篇幅，详细讲解这三个方面。

边框与背景

CSS 规范指出，边框绘制在“元素的背景之上”。这很重要，因为有些边框是“间断的”（例如，点线边框或虚线框），元素的背景应当出现在边框的可见部分之间。

CSS2 指出背景只延伸到内边距，而不是边框。后来 CSS2.1 进行了更正：元素的背景是内容、内边距和边框区的背景。大多数浏览器都遵循 CSS2.1 定义，不过一些较老的浏览器可能会有不同的表现。

边框的样式

样式是边框最重要的一个方面，这不是因为样式控制着边框的显示（当然，样式确实控制着边框的显示），而是因为如果没有样式，将根本没有边框。

CSS 的 border-style 属性定义了 10 个不同的非 inherit 样式，包括 none。

例如，可以为把一幅图片的边框定义为 outset，使之看上去像是“凸起按钮”：

```
a,img {border-style: outset;}
```

border-style 属性全部可能的值如下:

可能的值

值	描述
none	定义无边框。
hidden	与 "none" 相同。不过应用于表时除外，对于表，hidden 用于解决边框冲突。
dotted	定义点状边框。在大多数浏览器中呈现为实线。
dashed	定义虚线。在大多数浏览器中呈现为实线。
solid	定义实线。
double	定义双线。双线的宽度等于 border-width 的值。
groove	定义 3D 凹槽边框。其效果取决于 border-color 的值。
ridge	定义 3D 垄状边框。其效果取决于 border-color 的值。
inset	定义 3D inset 边框。其效果取决于 border-color 的值。
outset	定义 3D outset 边框。其效果取决于 border-color 的值。
inherit	规定应该从父元素继承边框样式。

范例:使用不同的边框样式

```
<html>
<head>
    <style type="text/css">
        p.dotted {
            border-style: dotted
        }
        p.dashed {
            border-style: dashed
        }
        p.solid {
            border-style: solid
        }
        p.double {
            border-style: double
        }
        p.groove {
            border-style: groove
        }
        p.ridge {
            border-style: ridge
        }
        p.inset {
            border-style: inset
        }
        p.outset {
```

```
            border-style: outset
        }
    </style>
</head>
<body>
    <p class="dotted">A dotted border</p>
    <p class="dashed">A dashed border</p>
    <p class="solid">A solid border</p>
    <p class="double">A double border</p>
    <p class="groove">A groove border</p>
    <p class="ridge">A ridge border</p>
    <p class="inset">An inset border</p>
    <p class="outset">An outset border</p>
</body>
</html>
```

定义多种样式

可以为一个边框定义多个样式，例如：

```
p.aside {border-style: solid dotted dashed double;}
```

上面这条规则为类名为 aside 的段落定义了四种边框样式：实线上边框、点线右边框、虚线下边框和一个双线左边框。

我们又看到了这里的值采用了 top-right-bottom-left 的顺序，讨论用多个值设置不同内边距时也见过这个顺序。

定义单边样式

如果希望为元素框的某一个边设置边框样式，而不是设置所有 4 个边的边框样式，可以使用下面的单边边框样式属性：

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

因此这两种方法是等价的：

```
p {border-style: solid solid solid none;}
p {border-style: solid; border-left-style: none;}
```

注意：如果要使用第二种方法，必须把单边属性放在简写属性之后。因为如果把单边属性放在 border-style 之前，简写属性的值就会覆盖单边值 none。

边框的宽度

可以通过 border-width 属性为边框指定宽度。

为边框指定宽度有两种方法：可以指定长度值，比如 2px 或 0.1em；或者使用 3 个关键字之一，它们分别是 thin 、medium（默认值） 和 thick。

注释：CSS 没有定义 3 个关键字的具体宽度，所以一个用户代理可能把 thin 、medium 和 thick 分别设置为等于 5px、3px 和 2px，而另一个用户代理则分别设置为 3px、2px 和 1px。

所以，我们可以这样设置边框的宽度：

```
p {border-style: solid; border-width: 5px;}
```

或者：

```
p {border-style: solid; border-width: thick;}
```

定义单边宽度

可以按照 top-right-bottom-left 的顺序设置元素的各边边框：

```
p {border-style: solid; border-width: 15px 5px 15px 5px;}
```

上面的例子也可以简写为（这样写法称为*值复制*）：

```
p {border-style: solid; border-width: 15px 5px;}
```

也可以通过下列属性分别设置边框各边的宽度：

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

因此，下面的规则与上面的例子是等价的：

```
p {
  border-style: solid;
  border-top-width: 15px;
  border-right-width: 5px;
  border-bottom-width: 15px;
  border-left-width: 5px;
  }
```

没有边框

在前面的例子中，您已经看到，如果希望显示某种边框，就必须设置边框样式，比如 solid 或 outset。

那么如果把 border-style 设置为 none 会出现什么情况：

```
p {border-style: none; border-width: 50px;}
```

尽管边框的宽度是 50px，但是边框样式设置为 none。在这种情况下，不仅边框的样式没有了，其宽度也会变成 0。边框消失了，为什么呢?

这是因为如果边框样式为 none，即边框根本不存在，那么边框就不可能有宽度，因此边框宽度自动设置为 0，而不论原先定义的是什么。

记住这一点非常重要。事实上，忘记声明边框样式是一个常犯的错误。根据以下规则，所有 h1 元素都不会有任何边框，更不用说 20 像素宽了：

```
h1 {border-width: 20px;}
```

由于 border-style 的默认值是 none，如果没有声明样式，就相当于 border-style: none。**因此，如果希望边框出现，就必须声明一个边框样式。**

边框的颜色

设置边框颜色非常简单。CSS 使用一个简单的 border-color 属性)，它一次可以接受最多 4 个颜色值。

可以使用任何类型的颜色值，例如可以是命名颜色，也可以是十六进制和 RGB 值：

```
p {
  border-style: solid;
  border-color: blue rgb(25%,35%,45%) #909090 red;
  }
```

如果颜色值小于 4 个，值复制就会起作用。例如下面的规则声明了段落的上下边框是蓝色，左右边框是红色：

```
p {
  border-style: solid;
  border-color: blue red;
  }
```

注释：默认的边框颜色是元素本身的前景色。如果没有为边框声明颜色，它将与元素的文本颜色相同。另一方面，如果元素没有任何文本，假设它是一个表格，其中只包含图像，那么该表的边框颜色就是其父元素的文本颜色（因为 color 可以继承）。这个父元素很可能是 body、div 或另一个 table。

定义单边颜色

还有一些单边边框颜色属性。它们的原理与单边样式和宽度属性相同：

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

要为 h1 元素指定实线黑色边框，而右边框为实线红色，可以这样指定：

```
h1 {
  border-style: solid;
  border-color: black;
  border-right-color: red;
  }
```

透明边框

我们刚才讲过，如果边框没有样式，就没有宽度。不过有些情况下可能希望创建一个不可见的边框。

CSS2 引入了边框颜色值 transparent。这个值用于创建有宽度的不可见边框。请看下面的例子：

```
<html>
<head>
<style type="text/css">
a:link, a:visited {
  border-style: solid;
  border-width: 5px;
  border-color: transparent;
  }
a:hover {border-color: gray;}
</style>
</head>
<body>
<a href="#">AAA</a>
</body>
</html>
```

CSS 边框属性

属性	描述
border	简写属性，用于把针对四个边的属性设置在一个声明。
border-style	用于设置元素所有边框的样式，或者单独地为各边设置边框样式。
border-width	简写属性，用于为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。
border-color	简写属性，设置元素的所有边框中可见部分的颜色，或为 4 个边分别设置颜色。
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中。
border-bottom-color	设置元素的下边框的颜色。
border-bottom-style	设置元素的下边框的样式。
border-bottom-width	设置元素的下边框的宽度。
border-left	简写属性，用于把左边框的所有属性设置到一个声明中。
border-left-color	设置元素的左边框的颜色。
border-left-style	设置元素的左边框的样式。
border-left-width	设置元素的左边框的宽度。

6.4 CSS外边距

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距的最简单的方法就是使用 margin 属性，这个属性接受任何长度单位、百分数值甚至负值。

CSS margin 属性

设置外边距的最简单的方法就是使用 margin 属性。

margin 属性接受任何长度单位，可以是像素、英寸、毫米或 em。

margin 可以设置为 auto。更常见的做法是为外边距设置长度值。下面的声明在 h1 元素的各个边上设置了 1/4 英寸宽的空白：

```
h1 {margin : 0.25in;}
```

下面的例子为 h1 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)：

```
h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距 (top) 开始围着元素顺时针旋转的：

```
margin: top right bottom left
```

另外，还可以为 margin 设置一个百分比数值：

```
p {margin : 10%;}
```

百分数是相对于父元素的 width 计算的。上面这个例子为 p 元素设置的外边距是其父元素的 width 的 10%。

margin 的默认值是 0，所以如果没有为 margin 声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，外边距会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 p 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要你特别作了声明，就会覆盖默认样式。

值复制

还记得吗？我们曾经在前两节中提到过值复制。下面讲解如何使用值复制。

有时，我们会输入一些重复的值：

```
p {margin: 0.5em 1em 0.5em 1em;}
```

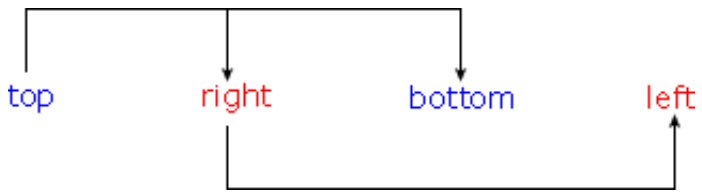
通过值复制，可以不必重复地键入这对数字。上面的规则与下面的规则是等价的：

```
p {margin: 0.5em 1em;}
```

这两个值可以取代前面 4 个值。这是如何做到的呢？CSS 定义了一些规则，允许为外边距指定少于 4 个值。规则如下：

- 如果缺少左外边距的值，则使用右外边距的值。
- 如果缺少下外边距的值，则使用上外边距的值。
- 如果缺少右外边距的值，则使用上外边距的值。

下图提供了更直观的方法来了解这一点：



换句话说，如果为外边距指定了 3 个值，则第 4 个值（即左外边距）会从第 2 个值（右外边距）复制得到。如果给定了两个值，第 4 个值会从第 2 个值复制得到，第 3 个值（下外边距）会从第 1 个值（上外边距）复制得到。最后一个情况，如果只给定一个值，那么其他 3 个外边距都由这个值（上外边距）复制得到。

利用这个简单的机制，我们只需指定必要的值，而不必全部都应用 4 个值，例如：

```
h1 {margin: 0.25em 1em 0.5em;}  /* 等价于 0.25em 1em 0.5em 1em */
h2 {margin: 0.5em 1em;}         /* 等价于 0.5em 1em 0.5em 1em */
p {margin: 1px;}                        /* 等价于 1px 1px 1px 1px */
```

这种办法有一个小缺点，您最后肯定会遇到这个问题。假设希望把 p 元素的上外边距和左外边距设置为 20 像素，下外边距和右外边距设置为 30 像素。在这种情况下，必须写作：

```
p {margin: 20px 30px 30px 20px;}
```

这样才能得到想要的结果。遗憾的是，在这种情况下，所需值的个数没有办法更少了。

再来看另外一个例子。如果希望除了左外边距以外所有其他外边距都是 auto（左外边距是 20px）：

```
p {margin: auto auto auto 20px;}
```

同样的，这样才能得到你想要的效果。问题在于，键入这些 auto 有些麻烦。如果只是希望控制元素单边上的外边距，请使用单边外边距属性。

单边外边距属性

可以使用单边外边距属性为元素单边上的外边距设置值。假设希望把 p 元素的左外边距设置为 20px。不必使用 margin（需要键入很多 auto），而是可以采用以下方法：

```
p {margin-left: 20px;}
```

可以使用下列任何一个属性来只设置相应上的外边距，而不会直接影响所有其他外边距：

- margin-top
- margin-right
- margin-bottom
- margin-left

一个规则中可以使用多个这种单边属性，例如：

```
h2 {
  margin-top: 20px;
  margin-right: 30px;
  margin-bottom: 30px;
  margin-left: 20px;
  }
```

当然，对于这种情况，使用 margin 可能更容易一些：

```
p {margin: 20px 30px 30px 20px;}
```

不论使用单边属性还是使用 margin，得到的结果都一样。一般来说，如果希望为多个边设置外边距，使用 margin 会更容易一些。不过，从文档显示的角度看，实际上使用哪种方法都不重要，所以应该选择对自己来说更容易的一种方法。

CSS 外边距合并

外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。

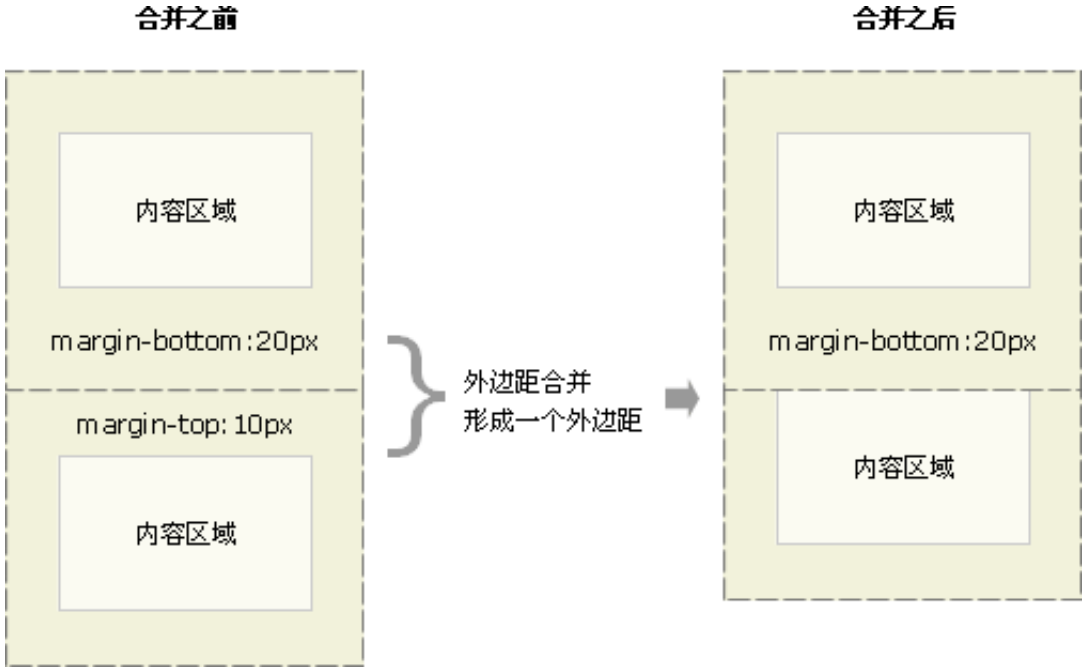
合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

外边距合并

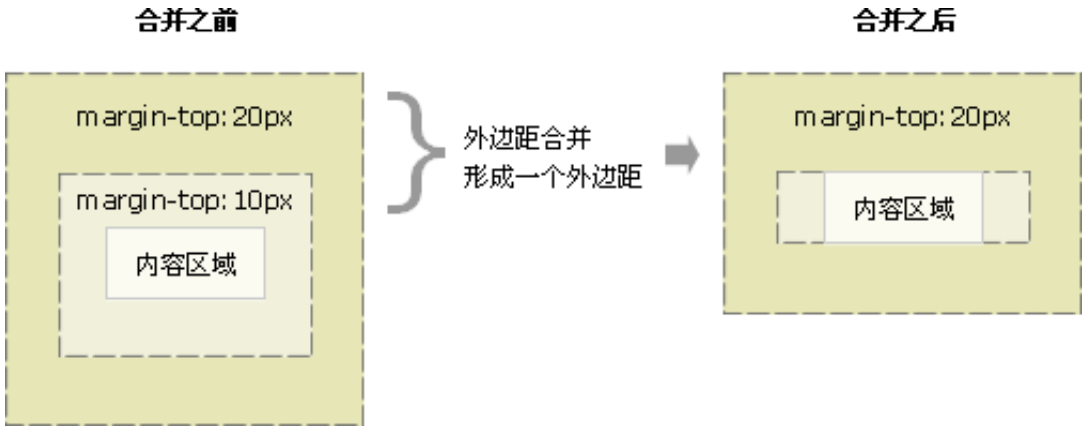
外边距合并（叠加）是一个相当简单的概念。但是，在实践中对网页进行布局时，它会造成许多混淆。

简单地说，外边距合并指的是，当两个垂直外边距相遇时，它们将形成一个外边距。合并后的外边距的高度等于两个发生合并的外边距的高度中的较大者。

当一个元素出现在另一个元素上面时，第一个元素的下外边距与第二个元素的上外边距会发生合并。请看下图：

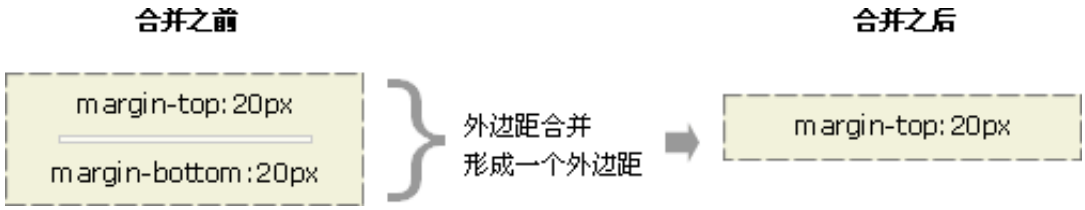


当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并。请看下图：

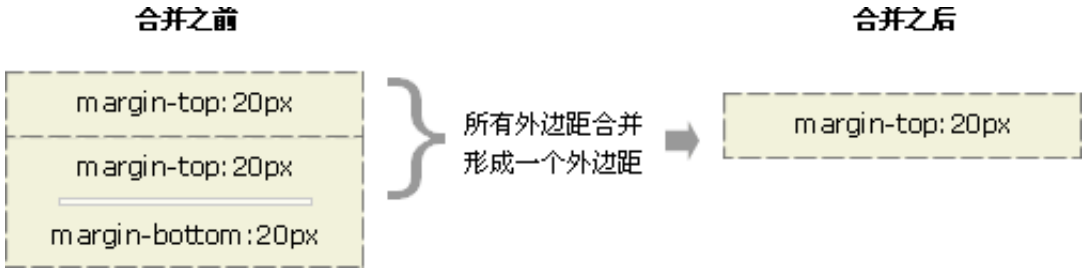


尽管看上去有些奇怪，但是外边距甚至可以与自身发生合并。

假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：



如果这个外边距遇到另一个元素的外边距，它还会发生合并：



这就是一系列的段落元素占用空间非常小的原因，因为它们的所有外边距都合并到一起，形成了一个小的外边距。

外边距合并初看上去可能有点奇怪，但是实际上，它是有意义的。以由几个段落组成的典型文本页面为例。第一个段落上面的空间等于段落的上外边距。如果没有外边距合并，后续所有段落之间的外边距都将是相邻上外边距和下外边距的和。这意味着段落之间的空间是页面顶部的两倍。如果发生外边距合并，段落之间的上外边距和下外边距就合并在一起，这样各处的距离就一致了。

6.5 盒模型与display属性

HTML组件中呈现一片空白区域的组件都可当做盒模型，比如<div../>元素、<span../>元素、<section../>等元素。而CSS则提供了display属性来控制盒模型的外观。

6.5.1 两种最基本的盒模型

就最基本的盒模型组件来说，HTML组件的盒模型可以分为两种。

- bolck类型 : 这种盒模型的组件默认占据一行，允许通过CSS设置宽度、高度。例如:<div.../>、<p../>元素。
- inline类型 : 这种盒模型的组件不会占据一行(默认允许在一行放置多个组件)。即使通过CSS设置宽度、高度也不会起作用。例如<span.../>、<a.../>、<img.../>元素。

范例:对比两种基本盒模型

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>基础盒模型</title>
    <style type="text/css">
        div,
        span {
            width: 300px;
            height: 40px;
            border: 1px solid black;
        }
    </style>
```

```
</head>
<body>
    <div>div元素1</div>
    <div>div元素2 </div>
    <span>span元素1</span>
    <span>span元素2</span>
</body>
</html>
```

上面页面中定义了两个<div../>元素和两个<span../>元素，页面的CSS代码设置了<div../>、<span../>元素的高度为40px，宽度为300px。但由于<span../>元素默认是inline盒模型，因此设置的宽度与高度对它不起作用。

CSS为display属性提供了block、inline两个属性值，用于改变HTML组件默认的盒模型。

范例:使用display属性破坏HTML组件默认的盒模型

```
<style type="text/css">
        div,span {
            width: 300px;
            height: 40px;
            border: 1px solid black;
        }
        body>div{
            display: inline;
        }
        body>span{
            display: block;
        }
    </style>
```

6.5.2 none值

display属性还可以指定为none值，用于设置目标对象隐藏，一旦该对象隐藏，其占用的页面空间也会释放。与此类似的还有visibility属性，该属性也可用于设置目标对象是否显示。与display属性不同，当通过visibility隐藏某个HTML元素后，该元素占用的页面空间依然会被保留。visibility属性的两个常用值为visible和hidden，分别用于控制目标对象的显示和隐藏。

范例：隐藏元素

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>隐藏元素</title>
    <style type="text/css">
        div{
            width: 300px;
            height: 40px;
            background-color: #dddddd;
```

```
            border: 2px solid black;
        }
    </style>
</head>
<body>
   <input type="button" value="隐藏"
        onclick="document.getElementById('test1').style.display='none';"/>
   <input type="button" value="显示"
        onclick="document.getElementById('test1').style.display='';"/>
    <div id="test1">
        使用display控制对象的显示与隐藏
    </div>
    <hr/>
    <input type="button" value="隐藏"
    onclick="document.getElementById('test2').style.visibility='hidden';"/>
    <input type="button" value="显示"
    onclick="document.getElementById('test2').style.visibility='visible';"/>
    <div id="test2">
        使用visibility来控制对象的显示与隐藏
    </div>
    <hr/>
</body>
</html>
```

6.5.3 inline-block盒模型

CSS还提供了一种inline-block盒模型，通过为display属性设置inline-block即可实现这种盒模型，这种盒模型是inline模型和block模型的综合体；inline-block盒模型的元素既不会占据一行，同时也通过width、height指定宽度及高度。

通过使用inline-block盒模型可以非常方便的实现多个<div../>元素并列显示。也就是说，使用inline-block盒模型可以实现多栏布局。

在默认情况下，多个inline-block盒模型的组件将会采用底端对齐的方式，也就是它们的底部将会位于同一条水平线上，这可能不是多栏布局期望的结果。为了让多个inline-block盒模型的组件在顶端对齐，为它们增加vertical-align:top；即可。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>多栏布局</title>
    <style type="text/css">
        body {
            margin: 0px;
        }
        div#container {
            width: 960px;
            margin: auto;
        }
```

```
        div>div {
            border: 1px solid #aaaaff;
            display: inline-block;
            vertical-align: top;
            /* 设置HTML组件的width属性包括边框 */
            box-sizing: border-box;
            -moz-box-sizing: border-box;
            border-radius: 12px 12px 0px 0px;
            background-color: #ffc;
            padding: 5px;
        }
    </style>
    <div id="container">
        <div style="width: 200px;">div1</div>
        <div style="width: 500px;">div2</div>
        <div style="width: 240px;">div3</div>
    </div>
</head>
<body>

</body>
</html>
```

除此之外，使用inline-block盒模型也可以非常方便的实现水平菜单。下面例子无须js脚本就实现了一个横向排列的导航菜单

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>导航菜单</title>
    <style type="text/css">
        body>div{
            text-align: center;
            margin: auto;
        }
        div>div{
            display: inline-block;
            border: 1px solid black;
        }
        a {
            text-decoration: none;
            display: block;
            width: 120px;
            padding: 10px;
            background-color: #eee;
        }
        a:hover {
            background-color: #aaa;
```

```
            font-weight: bold;
        }
    </style>
    <div>
        <div>
            <a href="#">前端从入门到精通</a>
        </div>
        <div>
            <a href="#">html从入门到精通</a>
        </div>
        <div>
            <a href="#">css3从入门到精通</a>
        </div>
        <div>
            <a href="#">javascript从入门到精通</a>
        </div>
    </div>
</head>
<body>

</body>
</html>
```

上面页面设置了4个子<div../>元素显示为inline-block盒模型，这样可以保证这4个子元素在同一行内显示。每个div子元素内包含一个超链接，这些超链接以block盒模型显示，它们会占满整个父容器，而且css为这些超链接设置了背景色，浏览者将会看到整个<div../>都会显示它所包含的超链接的背景色。

6.5.4 inline-table盒模型

在默认情况下，<table../>元素属于block盒模型，也就是说，该元素默认占据一行:它的左边不允许出现任何元素，右边也不允许出现其他内容。该元素可以通过width、height设置宽度和高度。

CSS为<table../>元素提供了一个inline-table盒模型，这个盒模型允许表格通过width、height设置宽度和高度，而且允许它的左右两边出现其他内容。

为了控制表格与前、后内容垂直对齐，可以通过添加vertical-align属性来实现，设置该属性为top，表示让表格与前后内容顶端对齐；设置该属性为bottom，表示让表格与前后内容底端对齐。

范例:使用table的inline模型

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>inline-table盒模型</title>
    <style type="text/css">
        td {
            border: 1px solid black;
        }
        table {
```

```
            width: 360px;
            border-collapse: collapse;
            display: inline-table;
            vertical-align: top;
        }
    </style>
</head>
<body>
     前面内容
        <table>
            <tr>
                <td>java从入门到精通</td>
                <td>多线程从入门到精通</td>
            </tr>
            <tr>
                <td>js从入门到精通</td>
                <td>node.js从入门到精通</td>
            </tr>
        </table>
        后面内容
</body>
</html>
```

6.5.5 表格相关的盒模型

除了上一节介绍的inline-table盒模型之外，CSS3还为display提供如下属性值。

> - table:将目标HTML组件显示为表格
> - table-caption:将目标HTML组件显示为表格标题
> - table-cell:将目标HTML组件显示为单元格
> - table-column:将目标HTML组件显示为表格列
> - table-column-group:将目标HTML组件显示为表格列组
> - table-header-group:将目标HTML组件显示为表格头部分
> - table-footer-group:将目标HTML组件显示为表格页脚部分
> - table-row:将目标HTML组件显示为表格行
> - table-row-group:将目标HTML表格显示为表格行组

通过上面这些盒模型，可以使用<div../>元素构建表格

范例:使用div构建表格

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>inline-table盒模型</title>
    <style type="text/css">
        div>div {
            display: table-row;
            padding: 10px;
```

```
        }
        div>div>div {
            display: table-cell;
            border: 1px solid black;
        }
    </style>
</head>
<body>
   <div style="display: table;width: 400px;">
       <div style="display: table-caption;">Java课程体系</div>
        <div>
            <div>Java从入门到精通</div>
            <div>多线程从入门到精通</div>
        </div>
        <div>
            <div>js从入门到精通</div>
            <div>node.js从入门到精通</div>
        </div>
    </div>
</body>
</html>
```

在上述页面中，虽然都是<div../>元素，但由于这些元素的display属性设置为表格相关的盒模型，因此各

div元素将会组成一个表格。

6.5.6 list-item盒模型

list-item模型可以将目标组件转换为类似于<ul../>的列表元素，也可以同时在元素前面添加列表标志。

范例:将div元素变为列表展示

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        div {
            display: list-item;
            list-style-type: square;
            margin-left: 20px;
        }
    </style>
</head>
<body>
   <div>Java从入门到精通</div>
   <div>Spring从入门到精通</div>
   <div>JavaScript从入门到精通</div>
</body>
```

```
</html>
```

实际上，如果为不同元素添加不同的列表符号，并使用不同的margin-left，就可以通过list-item盒模型实现多级列表的效果。

范例:list-item实现的多级列表

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        body>div {
            display: list-item;
            list-style-type: disc;
            margin-left: 20px;
        }
        div>div {
            display: list-item;
            list-style-type: square;
            margin-left: 40px;
        }
    </style>
</head>
<body>
   <div id="div1">Java课程体系
       <div>JavaSE</div>
       <div>Java多线程</div>
       <div>Java类集</div>
   </div>
   <div id="div2">JavaEE课程体系
       <div>SpringCore</div>
       <div>SpringMVC</div>
       <div>SpringBoot</div>
   </div>
</body>
</html>
```

6.6 布局相关属性

6.6.1 通过float属性实现多栏布局

通过使用float属性，可以很方便的基于<div..../>元素来设计导航菜单、多栏布局等效果。

> float属性控制目标HTML组件是否浮动以及如何浮动。当通过该属性设置某个对象浮动后，该对象将被当做块(block-level)组件处理，即相当于display属性被设置为block。也就是说，即使为浮动组件的display设置了其他属性值，该属性值依然是block。浮动HTML组件将会漂浮紧紧跟随它的前一个组件，直到遇到边框、padding、margin、另一个块组件为止。该属性支持left、right

> 两个属性值，分别指定对象向左、向右浮动。

范例:使用float属性实现多栏布局

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>多栏布局</title>
    <style type="text/css">
        body {
            margin: 0px;
        }
        div#container {
            width: 960px;
            margin: auto;
        }
        div>div {
            border: 1px solid #aaf;
            box-sizing: border-box;
            border-radius: 12px 12px 0px 0px;
            background-color: #ffc;
            padding: 5px;
        }
    </style>
</head>
<body>
    <div id="container">
        <div style="float: left;width: 220px">
            <h2>JavaSE课程体系</h2>
            <ul>
                <li>Java Thread</li>
                <li>Java Collections</li>
                <li>JVM</li>
            </ul>
        </div>
        <div style="float: left;width: 500px;">
            <h2>JavaEE课程体系</h2>
            <ul>
                <li>Mybatis</li>
                <li>SpringCore</li>
                <li>SpringMVC</li>
                <li>SpringBoot</li>
            </ul>
        </div>
        <div style="float: left;width: 240px;">
            <h2>前端课程体系</h2>
            <ul>
                <li>HTML</li>
```

```
                <li>CSS</li>
                <li>JavaScript</li>
            </ul>
        </div>
    </div>
</body>
</html>
```

6.6.2 使用clear属性实现换行

clear属性用于设置HTML组件的左、右是否允许出现"浮动"组件，如果该属性指定为left，则左边不允许出现"浮动"组件。如果指定为both，则两边都不允许出现浮动组件。借助于clear属性可以让"浮动"组件换行。

范例:clear实现换行。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>clear属性</title>
    <style type="text/css">
        div>div {
            width: 220px;
            padding: 5px;
            margin: 2px;
            float: left;
            background-color: #ddd;
        }
    </style>
</head>
<body>
    <div>
        <div>比特C语言</div>
        <div>比特C++</div>
        <div style="clear: both;">比特Java</div>
        <div>比特JS</div>
    </div>
</body>
</html>
```

上述代码为4个子div元素都设置了向左浮动，这会让他们都浮向左边，如果宽度足够，它们会并排排成一行。但由于将第三个div元素设置了clear:both，这意味着它的左右两边不允许出现浮动元素，因此会在该元素前后换行。

6.6.3 控制组件的滚动条

CSS提供了overflow、overflow-x、overflow-y三个属性值来控制HTML组件不够容纳内容时的显示方式，这三个属性的功能基本类似，区别只是overflow同时控制两个方向，而overflow-x只控制水平方向，overflow-y只控制垂直方向。

范例:使用overflow属性

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>overflow属性</title>
    <style type="text/css">
       div {
           width: 300px;
           height: 70px;
           white-space: nowrap;
           border: 1px solid black;
           margin: 15px;
       }
    </style>
</head>
<body>
   <div>
       <h3>不设置overflow属性</h3>
       测试文字测试文字测试文字测试文字测试文字测试文字
   </div>
   <div style="overflow: hidden;">
        <h3>overflow: hidden;</h3>
        测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字
    </div>
    <div style="overflow: auto;">
            <h3>overflow: auto;</h3>
            测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字
    </div>
    <div style="overflow-x: hidden;">
            <h3>overflow-x: hidden;</h3>
            测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字
    </div>
    <div style="overflow-y: hidden;">
            <h3>overflow-y: hidden;</h3>
            测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字测试文字
    </div>
</body>
</html>
```

6.6.4 CSS3新增的box-shadow属性

CSS3新增了box-shadow属性为盒模型添加阴影，该属性可用于为整个盒模型添加阴影。

box-shadow属性可以为所有盒模型的元素整体添加阴影。这是一个复合属性，包含有如下5个值:

> - hOffset:该属性值控制阴影在水平方向的偏移
> - vOffset:该属性值控制阴影在垂直方向的偏移
> - blurLength:该属性值控制阴影的模糊程度

> - scaleLength:该属性值控制阴影的缩放程度
> - color:该属性值控制阴影的颜色

范例:使用box-shadow为盒模型添加阴影

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>overflow属性</title>
    <style type="text/css">
        div {
            width: 300px;
            height: 50px;
            border: 1px solid black;
            margin: 30px;
        }
    </style>
</head>
<body>
   <div style="box-shadow: -10px -8px 6px #444;">
        box-shadow: -10px -8px 6px #444;(左上阴影)
   </div>
   <div style="box-shadow: 10px -8px 6px #444;">
        box-shadow: 10px -8px 6px #444;(右上阴影)
   </div>
   <div style="box-shadow:-10px 8px 6px #444;">
        box-shadow:10px 8px 6px #444;(左下阴影)
   </div>
   <div style="box-shadow:10px 8px 6px #444;">
        box-shadow:10px 8px 6px #444;(右下阴影)
   </div>
   <div style="box-shadow: 10px 8px #444;">
        box-shadow:10px 8px #444;(右下阴影,不指定模糊程度)
   </div>
   <div style="box-shadow: 10px 8px 20px #444;">
        box-shadow:10px 8px 20px #444;(右下阴影,增大模糊程度)
   </div>
   <div style="box-shadow: 10px 8px 10px -10px blueviolet;">
        box-shadow: 10px 8px 10px -10px red;(右下阴影,缩小阴影区域)
   </div>
   <div style="box-shadow: 10px 8px 10px 15px blueviolet;">
        box-shadow: 10px 8px 10px 10px red;(右下阴影,增大阴影区域)
   </div>
</body>
</html>
```

通过box-shadow还可以为表格、单元格添加阴影

范例:为表格添加阴影

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>box-shadow属性</title>
    <style type="text/css">
        table {
            width: 500px;
            border-spacing: 10px;
            box-shadow: 10px 10px 6px #444;
        }
        td {
            box-shadow: 6px 6px 4px #444;
            padding: 5px;
        }
    </style>
</head>
<body>
    <table>
        <tr>
            <td>JavaSE Thread</td>
            <td>JavaSE Collections</td>
        </tr>
        <tr>
            <td>JavaEE SpringCore</td>
            <td>JaveEE SpringMVC</td>
        </tr>
    </table>
</body>
</html>
```