CSS 语法

CSS 语法由三部分构成：选择器、属性和值：

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

选择器 (selector) 通常是你希望定义的 HTML 元素或标签，属性 (property) 是你希望改变的属性，并且每个属性都有一个值。属性和值被冒号分开，并由花括号包围，这样就组成了一个完整的样式声明（declaration）：

```
body {color: blue}
```

上面这行代码的作用是将 body 元素内的文字颜色定义为蓝色。在上述例子中，body 是选择器，而包括在花括号内的的部分是声明。声明依次由两部分构成：属性和值，color 为属性，blue 为值。

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

提示：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，这么的好处是，当你从现有的规则中增减声明时，会尽可能的减少出错的可能性。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，及 XHTML 不同，CSS 对大小写不敏感。不过存在一个例外：如果涉及到及 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的。

CSS 高级语法

- Previous Page
- Next Page

选择器的分组

你可以对选择器进行分组，这样，被分组的选择器就可以分享相同的声明。用逗号将需要分组的选择器分开。在下面的例子中，我们对所有的标题元素进行了分组。所有的标题元素都是绿色的。

```
h1,h2,h3,h4,h5,h6 {
  color: green;
  }
```

继承及其问题

根据 CSS，子元素从父元素继承属性。但是它并不总是按此方式工作。看看下面这条规则：

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

根据上面这条规则，站点的 body 元素将使用 Verdana 字体（假如访问者的系统中存在该字体的话）。

通过 CSS 继承，子元素将继承最高级元素（在本例中是 body）所拥有的属性（这些子元素诸如 p, td, ul, ol, ul, li, dl, dt,和 dd）。不需要另外的规则，所有 body

的子元素都应该显示 Verdana 字体，子元素的子元素也一样。并且在大部分的现代浏览器中，也确实是这样的。

但是在那个浏览器大战的血腥年代里，这种情况就未必会发生，那时候对标准的支持并不是企业的优先选择。比方说，Netscape 4 就不支持继承，它不仅忽略继承，而且也忽略应用于 body 元素的规则。IE/Windows 直到 IE6 还存在相关的问题，在表格内的字体样式会被忽略。我们又该如何是好呢？

友善地对待 Netscape 4

幸运地是，你可以通过使用我们称为 "Be Kind to Netscape 4" 的冗余法则来处理旧式浏览器无法理解继承的问题。

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

p, td, ul, ol, li, dl, dt, dd  {
     font-family: Verdana, sans-serif;
     }
```

4.0 浏览器无法理解继承，不过他们可以理解组选择器。这么做虽然会浪费一些用户的带宽，但是如果需要对 Netscape 4 用户进行支持，就不得不这么做。

继承是一个诅咒吗？

如果你不希望 "Verdana, sans-serif" 字体被所有的子元素继承，又该怎么做呢？比方说，你希望段落的字体是 Times。没问题。创建一个针对 p 的特殊规则，这样它就会摆脱父元素的规则：

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

td, ul, ol, ul, li, dl, dt, dd  {
```

```
    font-family: Verdana, sans-serif;
```

```
    }


p  {
    font-family: Times, "Times New Roman", serif;
    }
```

CSS 派生选择器

- Previous Page
- Next Page

派生选择器

通过依据元素在其位置的上下文关系来定义样式，你可以使标记更加简洁。

在 CSS1 中，通过这种方式来应用规则的选择器被称为上下文选择器 (contextual selectors)，这是由于它们依赖于上下文关系来应用或者避免某项规则。在 CSS2 中，它们称为派生选择器，但是无论你如何称呼它们，它们的作用都是相同的。

派生选择器允许你根据文档的上下文关系来确定某个标签的样式。通过合理地使用派生选择器，我们可以使 HTML 代码变得更加整洁。

比方说，你希望列表中的 strong 元素变为斜体字，而不是通常的粗体字，可以这样定义一个派生选择器：

```
li strong {
    font-style: italic;
    font-weight: normal;
  }
```

请注意标记为 的蓝色代码的上下文关系：

```
<p><strong>我是粗体字，不是斜体字，因为我不在列表当中，所以这个规则对我不起作用</strong></p>


<ol>
<li><strong>我是斜体字。这是因为 strong 元素位于 li 元素内。</strong></li>
```

```
<li>我是正常的字体。</li>
```

```
</ol>
```

在上面的例子中，只有 li 元素中的 strong 元素的样式为斜体字，无需为 strong 元素定义特别的 class 或 id，代码更加简洁。

再看看下面的 CSS 规则：

```
strong {
    color: red;
    }

h2 {
    color: red;
    }

h2 strong {
    color: blue;
    }
```

下面是它施加影响的 HTML：

```
<p>The strongly emphasized word in this paragraph
is<strong>red</strong>.</p>
<h2>This subhead is also red.</h2>
```

CSS id 选择器

- Previous Page
- Next Page

id 选择器

id 选择器可以为标有特定 id 的 HTML 元素指定特定的样式。

id 选择器以 "#" 来定义。

下面的两个 id 选择器，第一个可以定义元素的颜色为红色，第二个定义元素的颜色为绿色：

```
#red {color:red;}
```

```
#green {color:green;}
```

下面的 HTML 代码中，id 属性为 red 的 p 元素显示为红色，而 id 属性为 green 的 p 元素显示为绿色。

```
<p id="red">这个段落是红色。</p>
<p id="green">这个段落是绿色。</p>
```

注意：id 属性只能在每个 HTML 文档中出现一次。想知道原因吗，请参阅 XHTML:网站重构。

id 选择器和派生选择器

在现代布局中，id 选择器常常用于建立派生选择器。

```
#sidebar p {
        font-style: italic;
        text-align: right;
        margin-top: 0.5em;
        }
```

上面的样式只会应用于出现在 id 是 sidebar 的元素内的段落。这个元素很可能是 div 或者是表格单元，尽管它也可能是一个表格或者其他块级元素。它甚至可以是一个内联元素，比如 或者 ，不过这样的用法是非法的，因为不可以在内联元素 中嵌入 <p> （如果你忘记了原因，请参阅 XHTML:网站重构）。

一个选择器，多种用法

即使被标注为 sidebar 的元素只能在文档中出现一次，这个 id 选择器作为派生选择器也可以被使用很多次：

```
#sidebar p {
        font-style: italic;
        text-align: right;
        margin-top: 0.5em;
```

```
        }
```

```
#sidebar h2 {
        font-size: 1em;
        font-weight: normal;
        font-style: italic;
        margin: 0;
        line-height: 1.5;
        text-align: right;
        }
```

在这里，及页面中的其他 p 元素明显不同的是，sidebar 内的 p 元素得到了特殊的处理，同时，及页面中其他所有 h2 元素明显不同的是，sidebar 中的 h2 元素也得到了不同的特殊处理。

单独的选择器

id 选择器即使不被用来创建派生选择器，它也可以独立发挥作用：

```
#sidebar {
        border: 1px dotted #000;
        padding: 10px;
        }
```

根据这条规则，id 为 sidebar 的元素将拥有一个像素宽的黑色点状边框，同时其周围会有 10 个像素宽的内边距（padding，内部空白）。老版本的 Windows/IE 浏览器可能会忽略这条规则，除非你特别地定义这个选择器所属的元素：

```
div#sidebar {
        border: 1px dotted #000;
        padding: 10px;
        }
```

CSS 类选择器

- Previous Page

- Next Page

在 CSS 中，类选择器以一个点号显示：

```
.center {text-align: center}
```

在上面的例子中，所有拥有 center 类的 HTML 元素均为居中。

在下面的 HTML 代码中，h1 和 p 元素都有 center 类。这意味着两者都将遵守 ".center" 选择器中的规则。

```
<h1 class="center">
This heading will be center-aligned
</h1>

<p class="center">
This paragraph will also be center-aligned.
</p>
```

注意：类名的第一个字符不能使用数字！它无法在 Mozilla 或 Firefox 中起作用。

和 id 一样，class 也可被用作派生选择器：

```
.fancy td {
        color: #f60;
        background: #666;
        }
```

在上面这个例子中，类名为 fancy 的更大的元素内部的表格单元都会以灰色背景显示橙色文字。（名为 fancy 的更大的元素可能是一个表格或者一个 div）

元素也可以基于它们的类而被选择：

```
td.fancy {
        color: #f60;
```

```
        background: #666;
```