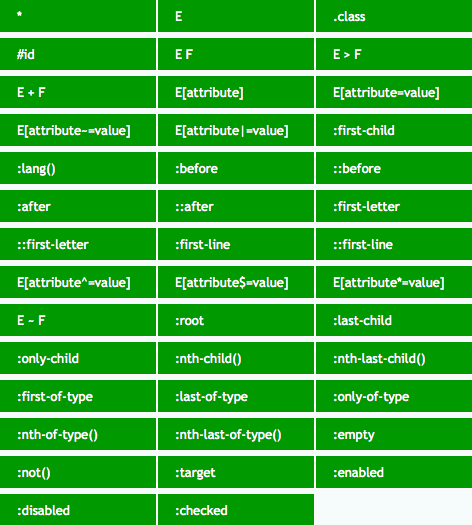
**基本选择器**

常用的选择器列表图



下面我们先看看上表中基本的选择器的使用方法和其所起的作用，为了更好的说明问题，先创建一个简单的DOM结构，如下：

<div class="demo">

<ul class="clearfix">

<li id="first" class="first">1</li>

<li class="active important">2</li>

<li class="important items">3</li>

<li class="important">4</li>

<li class="items">5</li>

<li>6</li>

<li>7</li>

<li>8</li>

<li>9</li>

<li id="last" class="last">10</li>

</ul>

</div>

给这个demo加上一些样式，让他好看一点

.demo {

width: 300px;

border: 1px solid #ccc;

padding: 10px;

}

li {

float: left;

height: 20px;

line-height: 20px;

width: 20px;

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

border-radius: 10px;

text-align: center;

background: #f36;

color: green;

margin-right: 5px;

}

初步效果如下所示：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-status.png

**一、通配符选择器（＊）**

通配符选择器是用来选择所有元素，，也可以选择某个元素下的所有元素。如：

\*{

margin: 0;

padding: 0;

}

上面代码大家在reset样式文件中看到的肯定不少，他所表示的是，所有元素的margin和padding都设置为0，另外一种就是选择某个元素下的所有元素：

.demo \* {border:1px solid blue;}

效果如下；

http://cdn2.w3cplus.com/sites/default/files/css3-selector-tp.png

从上面的效果图看，只要是div.demo下的元素边框都加上了新的样式。**所有浏览器支持通配符选择器。**

**二、元素选择器(E)**

元素选择器，是css选择器中最常见而且最基本的选择器。元素选择器其实就是文档的元素，如html,body,p,div等等，比如我们这个demo：中元素包括了div,ul,li等。

li {background-color: grey;color: orange;}

上在表示选择页面的li元素，并设置了背景色和前景色，效果如下：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-e.png

**所有浏览器支持元素选择器(E)。**

**三、类选择器（.className）**

类选择器是以一独立于文档元素的方式来指定样式，使用类选择器之前需要在html元素上定义类名，换句话说需要保证类名在html标记中存在，这样才能选择类，如：

<li class="active important items">2</li>

共中“active,important, items”就是我们以类给li加上一个类名，以便类选择器能正常工作，从而更好的将类选择器的样式与元素相关联。

.important {font-weight: bold; color: yellow;}

上面代码表示是给有important类名的元素加上一个“字体为粗体，颜色为黄色”的样式，如

http://cdn2.w3cplus.com/sites/default/files/css3-selector-classname.png

类选择器还可以结合元素选择器来使用，比如说，你文档中有好多个元素使用了类名“items”，但你只想在p元素这个类名上修改样式，那么你可以这样进行选择并加上相应的样式：

p.items {color: red;}

上面代码只会匹配class 属性包含important 的所有p 元素，但其他任何类型的元素都不匹配，包括有“items”这个类名的元素，上面也说过了“p.items”只会对p元素并且是其有一个类名叫“items”。不符合这两个条件的都不会被选择。

类选择器还有可以具备多类名，上面我们也看到了，我们li元素中同时有两个或多少类名，其中他们以空格隔开，那么选择器也可以使用多类连接在一起，如：

.important {font-weight: bold;}

.active {color: green;background: lime;}

.items {color: #fff;background: #000;}

.important.items {background:#ccc;}

.first.last {color: blue;}

正如上面的代码所示，".important.items"这个选择器只对元素中同时包含了"important"和"items"两个类才能起作用，如图所示：

http://cdn2.w3cplus.com/sites/default/files/css-select-calssName-2.png

有一点大家需要注意，如果一个多类选择器包含的类名中其中有一个不存在，那么这个选择器将无法找到相匹配的元素 比如说下在这句代码，他就无法到找相对应的的元素标签，因为我们列表中只有一个li.first和一个li.last，不存在有一个叫li.first.last的列表项：

.first.last {color: blue;}

**所有浏览器都支持类选择器，但多类选择器（.className1.className2）不被ie6支持。**

**四、id选择器（#ID）**

ID选择器和上面说的类选择器是很相似的，在使用ID选择器之前也需要先在html文档中加注ID名称，这样在样式选择器中才能找到相对应的元素，不同的是ID选择器是一个页面中唯一的值，我们在类使用时是在相对应的类名前加上一个“.”号（.className）而id选择器是在名称前使用"#"如(#id),

#first {background: lime;color: #000;}

#last {background: #000;color: lime;}

上在的代码就是选择了id为"first"和"last"的列表项，其效果如下

http://cdn.w3cplus.com/sites/default/files/css-select-id.png

ID选择器有几个地方需要特别注意的，第一：一个文档中一个id选择器只充许使用一次，因为id在页面中是唯一的；第二，id选择器不能像类选择器一样多个合并使用，一个元素只能命名一个id名；第三，可以在不同的文档中使用相同的id名，比如说在“test.html”中给h1定了“＃important”,也可以给“test1.html”中定义p的id为"#important"，但前提是不管在test.html还是test1.html中只充许有一个id叫"#important"的存在。

**所有浏览器都支持ID选择器。**

那么什么时候采用id命名？什么时候采用类命名呢？我个人认为就是关键的一点就是**具有唯一性使用id选择器；公用的，类似的使用类选择器。**使用这两个选择器时，最好区别大小写。

**五、后代选择器（Ｅ Ｆ）**

后代选择器也被称作包含选择器,所起作用就是可以选择某元素的后代元素，比如说:E F，前面E为祖先元素，Ｆ为后代元素，所表达的意思就是选择了Ｅ元素的所有后代Ｆ元素，请注意他们之间需要一个空格隔开。这里Ｆ不管是E元素的子元素或者是孙元素或者是更深层次的关系，都将被选中，换句话说，不论Ｆ在Ｅ中有多少层关系，都将被选中：

.demo li {color: blue;}

上面表示的是，选中div.demo中所有的li元素

http://cdn2.w3cplus.com/sites/default/files/css3-selector-ef.png

**所有浏览器都支的后代选择器。**

**六、子元素选择器(E>F)**

子元素选择器只能选择某元素的子元素，其中E为父元素，而F为子元素，其中E>F所表示的是选择了Ｅ元素下的所有子元素Ｆ。这和后代选择器（Ｅ Ｆ）不一样，在后代选择器中Ｆ是Ｅ的后代元素，而子元素选择器Ｅ > F，其中Ｆ仅仅是Ｅ的子元素而以。

ul > li {background: green;color: yellow;}

上在代码表示选择ul下的所有子元素li。如：

http://cdn.w3cplus.com/sites/default/files/css3-selector-chirld.png

**IE6不支持子元素选择器。**

**七、相邻兄弟元素选择器(E + F)**

相邻兄弟选择器可以选择紧接在另一元素后的元素，而且他们具有一个相同的父元素，换句话说，EF两元素具有一个相同的父元素，而且Ｆ元素在Ｅ元素后面，而且相邻，这样我们就可以使用相邻兄弟元素选择器来选择Ｆ元素。

li + li {background: green;color: yellow; border: 1px solid #ccc;}

上面代码表示选择li的相邻元素li，我们这里一共有十个li，那么上面的代码选择了从第2个li到 10 个li，一共九个，请看效果：

http://cdn.w3cplus.com/sites/default/files/css-selector-xl.png

因为上面的li+li其中第二li是第一li的相邻元素，第三个又是第二个相邻元素，因此第三个也被选择，依此类推，所以后面九个li都被选中了，如果我们换过一种方式来看，可能会更好的理解一点：

.active + li {background: green;color: yellow; border: 1px solid #ccc;}

按照前面所讲的知识，这句代码很明显选择了li.active后面相邻的li元素，注意了和li.active后面相邻的元素仅只有一个的。如图：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-next.png

**IE6不支持这个选择器**

**八、通用兄弟选择器（Ｅ 〜 Ｆ）**

通用兄弟元素选择器是CSS3新增加一种选择器，这种选择器将选择某元素后面的所有兄弟元素，他们也和相邻兄弟元素类似，需要在同一个父元素之中，换句话说，E和F元素是属于同一父元素之内，并且F元素在Ｅ元素之后，那么E ~ F 选择器将选择中所有Ｅ元素后面的Ｆ元素。比如下面的代码：

.active ~ li {background: green;color: yellow; border: 1px solid #ccc;}

上面的代码所表示的是，选择中了li.active 元素后面的所有兄弟元素li,如图所示：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-tongyong.png

通用兄弟选择器和相邻兄弟选择器极其相似，只不过，相邻兄弟选择器仅选择是元素的仅与其相邻的后面元素（选中的仅一个元素）；而通用兄弟元素选择器，选中的是元素相邻的后面兄弟元素，这样说起来可能会有迷糊，大家可以仔细看看其相邻兄弟的效果图。

**IE6不支持这种选择器的用法。**

**九、群组选择器（selector1,selector2,...,selectorN）**

群组选择器是将具有相同样式的元素分组在一起，每个选择器之间使用逗号“，”隔开，如上面所示selector1,selector2,...,selectorN。这个逗号告诉浏览器，规则中包含多个不同的选择器，如果不有这个逗号，那么所表达的意就完全不同了，省去逗号就成了我们前面所说的后代选择器，这一点大家在使用中千万要小心加小心。我们来看一个简单的例子：

.first, .last {background: green;color: yellow; border: 1px solid #ccc;}

因为li.first和li.last具有相同的样式效果，所以我们把他们写到一个组里来：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-group.png

**所有浏览器都支持群组选择器。**

上面九种选择器是CSS3中的基本选择器，而我们最常用的是**元素选择器、类选择器、ID选择器、后代选择器、群组选择器**，同时大家可以在实际应用中把这些选择器结合起来使用，达到目的就行了。

**属性选择器**

CSS3的属性选择器主要包括以下几种：

1. **E[attr]**：只使用属性名，但没有确定任何属性值；
2. **E[attr="value"]**：指定属性名，并指定了该属性的属性值；
3. **E[attr~="value"]**：指定属性名，并且具有属性值，此属性值是一个词列表，并且以空格隔开，其中词列表中包含了一个value词，而且等号前面的“〜”不能不写；
4. **E[attr^="value"]**：指定了属性名，并且有属性值，属性值是以value开头的；
5. **E[attr$="value"]**：指定了属性名，并且有属性值，而且属性值是以value结束的；
6. **E[attr\*="value"]**：指定了属性名，并且有属性值，而且属值中包含了value；
7. **E[attr|="value"]**：指定了属性名，并且属性值是value或者以“value-”开头的值（比如说zh-cn）;

把上面的demo换成别的结构，如下所示：

<div class="demo clearfix">

<a href="http://www.w3cplus.com" target="\_blank" class="links item first" id="first" title="w3cplus">1</a>

<a href="" class="links active item" title="test website" target="\_blank" lang="zh">2</a>

<a href="sites/file/test.html" class="links item" title="this is a link" lang="zh-cn">3</a>

<a href="sites/file/test.png" class="links item" target="\_balnk" lang="zh-tw">4</a>

<a href="sites/file/image.jpg" class="links item" title="zh-cn">5</a>

<a href="mailto:w3cplus@hotmail" class="links item" title="website link" lang="zh">6</a>

<a href="" class="links item" title="open the website" lang="cn">7</a>

<a href="" class="links item" title="close the website" lang="en-zh">8</a>

<a href="" class="links item" title="http://www.sina.com">9</a>

<a href="" class="links item last" id="last">10</a>

</div>

初步美化一下上面的代码

.demo {

width: 300px;

border: 1px solid #ccc;

padding: 10px;

}

.demo a {

float: left;

display: block;

height: 20px;

line-height: 20px;

width: 20px;

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

border-radius: 10px;

text-align: center;

background: #f36;

color: green;

margin-right: 5px;

text-decoration: none;

}

最初效果如下所示：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-status.png

下面就开始针对上面列出的每个属性选择器来，具体分析其使用方法。

#### 一、E[attr]

E[attr]属性选择器是CSS3属性选择器中最简单的一种。如果你希望选择有某个属性的元素，而不论这个属性值是什么，你就可以使用这个属性选择器，如：

.demo a[id] {background: blue; color:yellow;font-weight:bold;}

上面代码所表示的，选择了div.demo下所有带有id属性的a元素，并在这个元素上使用背景色为兰色，前景色为黄色，字体加粗的样式，对照上面的html，我们不难发现，只有第一个和最后一个链接使用了id属性，所以选中了这两个a元素，效果如下所示：

http://cdn.w3cplus.com/sites/default/files/css3-selector-attr.png

上面是单一属性的使用，你也可以使用多属性进行选择元素，如E[attr1][attr2], 这样只要是同时具有这两属性的元素都将被选中：

.demo a[href][title] {background: yellow; color:green;}

不用我说，上面的代码大家都知道是什么意思了，他表示的是选择div.demo下的同时具有href,title两个属性的a元素，并且应用相对应的样式，如下所示：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-attr1.png

**IE6不支持这个选择器。**

#### 二、Ｅ[attr="value"]

E[attr="value"]选择器和E[attr]选择器，从字面上就能很清楚的理解出来，E[attr="value"]是指定了属性值“value”，而E[attr]只是选择了有对应的属性，并没有明确指其对应的属性值"value"，这也是这两种选择器的最大区是之处。从而缩小了选择 围，更能精确选择自己需要的元素，在前面实例基础上我们进行一下简单的修改：

.demo a[id="first"] {background: blue; color:yellow;font-weight:bold;}

和前面代码相比较，此处在id的属性基础上指定了相应的value值为“first”,这样一来我们选中的是div.demo中的a元素，并且这个元素有一个"id="first""属性值，请看下在的效果图：

http://cdn.w3cplus.com/sites/default/files/css3-selector-attr2.png

E[attr="value"]属性选择器也可以多个属性并写，进一步缩小选择范围：

.demo a[href="http://www.w3cplus.com"][title] {background: yellow; color:green;}

效果如下：

http://cdn1.w3cplus.com/sites/default/files/css-selector-attr3.png

**对于E[attr="value"]这种属性值选择器有一点需要注意：属性和属性值必须完全匹配，特别是对于属性值是词列表的形式时，如：**

<a href="" class="links item" title="open the website">7</a>

例如上面的代码，如果你写成：

.demo a[class="links"]{color:red};/\*这是一种写法不能和上面的html所匹配\*/

上面的属性选择器并不会和上在的html匹配,因为他们的属性和属性值没有完全匹配，需要改成如下所示的代码，才能正确匹配：

.demo a[class="links item"]{color:red};/\*这样才是匹配的，记得中间的空格不能少的哟\*/

**IE6浏览器不支持这种选择器。**

#### 三、E[attr~="value"]

如果你想根据属性值中的词列表的某个词来进行选择元素，那么就需要使用这种属性选择器：E[attr~="value"]，这种属性选择器是属性值是一个或多个词列表，如果是列表时，他们需要用空格隔开，只要属性值中有一个value相匹配就可以选中该元素，而我们前面所讲的E[attr="value"]是属性值需要完全匹配才会被选中，他们两者区别就是一个有“〜”号，一个没有“〜”号。我们来看一个这方面的实例：

.demo a[title~="website"]{background:orange;color:green;}

上面代码表示的是，div.demo下的a元素的title属性中，只要其属性值中含有"website"这个词就会被选择，回头看看我们的html，不难发现所有a元素中“2，6，7，8”这四个a元素的title中都含有,所以被选中，请看效果：

http://cdn.w3cplus.com/sites/default/files/css3-selector-attr4.png

如果我们在上面的代码中，把那个“〜”号省去，大家看看他们不同之处：

.demo a[title="website"]{background:orange;color:green;}

这样将不会选择中任何元素，因为在所有a元素中无法找到完全匹配的"title='website'",换句话说就没有选中任何元素，效果如下：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-status.png

这个实例再次证明了E[attr="value"]和E[attr~="value"]之间的区别，和其中“〜”所取的作用，我总结了一句话：**属性选择器中有波浪（〜）时属性值有value时就相匹配，没有波浪（〜）时属性值要完全是value时才匹配。**

**IE6不支持E[attr~="value"]属性选择器。**

#### 四、E[attr^="value"]

E[attr^="value"]属性选择器，指的是选择attr属性值以“value”开头的所有元素，换句话说，选择的属性其以对应的属性值是以“value”开始的，一起来看个实例：

.demo a[href^="http://"]{background:orange;color:green;}

.demo a[href^="mailto:"]{background:green;color:orange;}

上面代码表示的是选择了以href属性，并且以"http://"和"mailto:"开头的属性值的所有a元素，换过更简单一点的呢？只要a元素中的href属性值是以"http://"或"mailto:"开头的a元素都会以选中，那么下面大家请对照上面的html和下面的效果图，看看是不是那么一回事：

http://cdn.w3cplus.com/sites/default/files/css3-selector-attr5.png

**IE6不支持E[attr^="value"]选择器。**

#### 五、E[attr$="value"]

Ｅ[attr$="value"]属性选择器刚好与E[attr^="value"]选择器相反，E[attr$="value"]表示的是选择attr属性值以"value"结尾的所有元素，换句话说就是选择元素attr属性，并且他的属性值是以value结尾的，这个运用在给你一些特殊的链接加背景图片很方便的，比如说给pdf,png,doc等不同文件加上不同icon，我们就可以使用这个属性来实现，如：

.demo a[href$="png"]{background:orange;color:green;}

上面代码表示的是，选择div.demo中元素有href属性，并以png值结尾的a元素。（正如上面所说，只不过这里使用的是改变元素的背景色），效果如下：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-attr6.png

**IE6不支持E[attr$="value"]属性选择器。**

#### 六、E[attr\*="value"]

E[attr\*="value"]属性选择器表示的是选择attr属性值中包含子串"value"的所有元素。也就是说，只要你所选择的属性，其属性值中有这个"value"值都将被选中，如：

.demo a[title\*="site"]{background:black;color:white;}

上面代码表示的是：选择了div.demo中a元素，而a元素的title属性中只要有"site"就符合选择条件。效果如下：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-attr7.png

**IE6不支持E[attr\*="value"]选择器。**

#### 七、E[attr|="value"]

E[attr|="value"]是属性选择器中的最后一种，在说这个选择器使用之前先提醒大家attr后面的是一个竖线“｜”而不是l，小心搞错了。E[attr|="value"]被称作为特定属性选择器。这个选择器会选择attr属性值等于value或以value-开头的所有元素,我们来看个实例：

.demo a[lang|="zh"]{background:gray;color:yellow;}

上面的代码会选中了div.demo中lang属性等于zh或以zh-开头的所有a元素，大家可以对照前面的html代友，其中"2,3,4,6"被选中，因为他们都有一个lang属性，并且他们的属性值都符合以"zh"或"zh-"开始的元素。具体效果如下：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-attr8.png

所以这种属性选择器用来匹配以“女value-1”,"value-2"的属性是很方便的，比如说你页面中有很多图片，图片文件名都是以"figure-1","figure-2"这样的方式来命名的，那么使用这种选择器选中图片就很方便了，大家可以在本地尝试一下，这种属性选择器最常常用的地方是如上面的示例用来匹配语言。

**IE6不支持E[attr|="value"]选择器。**

有关于属性选择器就上面这些内容了，属性选择器除了IE6不支持外，其他的浏览器都能支持，这样一来，如果你在你的页面上使用了属性选择器，而且你需要处理ie6兼容问题，那你就需要确保IE6用别的方法来实现或者你应该确保IE6用户将能获得一个可用的页面。七种属性选择器中**E[attr="value"]和E[attr\*="value"]是最实用的，其中E[attr="value"]能帮我们定位不同类型的元素，特别是表单form元素的操作，比如说input[type="text"],input[type="checkbox"]等，而E[attr\*="value"]能在网站中帮助我们匹配不同类型的文件，比如说你的网站上不同的文件类型的链接需要使用不同的icon图标，用来帮助你的网站提高用户体验，就像前面的实例，可以通过这个属性给".doc",".pdf",".png",".ppt"配置不同的icon图标。**

**伪类选择器**

伪类选择器对于大家来说最熟悉的莫过于:link,:focus,:hover之类的了，因为这些在平时中是常用到的伪类选择器，那么先和大家一起简单总结一下CSS中常用的伪类选择器的使用方法，最后把重心放到CSS3新增加的":nth-child"选择器的使用方法。

和前面一样，在开始之前先创建一个Dom:

<div class="demo clearfix">

<ul class="clearfix">

<li class="first links odd" id="first"><a href="">1</a></li>

<li class="links even"><a href="">2</a></li>

<li class="links odd"><a href="">3</a></li>

<li class="links even"><a href="">4</a></li>

<li class="links odd"><a href="">5</a></li>

<li class="links even"><a href="">6</a></li>

<li class="links odd"><a href="">7</a></li>

<li class="links even"><a href="">8</a></li>

<li class="links odd"><a href="">9</a></li>

<li class="links even last" id="last"><a href="">10</a></li>

</ul>

</div>

同样先加上一些样式，让他看起来好看一点

.demo {

width: 300px;

border: 1px solid #ccc;

padding: 10px;

}

.demo li {

border: 1px solid #ccc;

padding: 2px;

float: left;

margin-right:4px;

}

.demo a {

float: left;

display: block;

height: 20px;

line-height: 20px;

width: 20px;

-moz-border-radius: 10px;

-webkit-border-radius: 10px;

border-radius: 10px;

text-align: center;

background: #f36;

color: green;

text-decoration: none;

}

最出效果：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-pausse1.png

CSS的伪类语法和别的语法有点不一样，其主要有两种语法表达方式

E:pseudo-class {property:value}/\*其中E为元素;pseudo-class为伪类名称;property是css的属性;value为css的属性值\*/

示例：

a:link {color:red;}

第二种写法

E.class:pseudo-class{property:value}

如：

a.selected:hover {color: blue;}

下面开始我们一起来看看这些伪类的具体应用：

一、动态伪类

动态伪类，因为这些伪类并不存在于HTML中,而只有当用户和网站交互的时候才能体现出来，动态伪类包含两种，第一种是我们在链接中常看到的锚点伪类，如":link",":visited";另外一种被称作用户行为伪类，如“:hover”,":active"和":focus"。先来看最常见的锚点伪类

.demo a:link {color:gray;}/\*链接没有被访问时前景色为灰色\*/

.demo a:visited{color:yellow;}/\*链接被访问过后前景色为黄色\*/

.demo a:hover{color:green;}/\*鼠标悬浮在链接上时前景色为绿色\*/

.demo a:active{color:blue;}/\*鼠标点中激活链接那一下前景色为蓝色\*/

对于这四个锚点伪类的设置，有一点需要特别注意，那就是他们的先后顺序，要让他们遵守一个爱恨原则LoVe/HAte,也就是Link--visited--hover--active。如果你把顺序搞错了会给你带来意想不到的错误，大家对于这个应该很熟悉的，如果是初学的朋友，可以私下练习一下。其中:hover和:active又同时被列入到用户行为伪类中，他们所表达的意思是：

:hover用于当用户把鼠标移动到元素上面时的效果；

:active用于用户点击元素那一下的效果（正发生在点的那一下，松开鼠标左键此动作也就完成了）

:focus用于元素成为焦点，这个经常用在表单元素上。

本站的button就采用了这几处效果：

.form-submit {

-moz-transition: border-color 0.218s ease 0s;

-webkit-transition: border-color 0.218s ease 0s;

-o-transition: border-color 0.218s ease 0s;

-ms-transition: border-color 0.218s ease 0s;

transition: border-color 0.218s ease 0s;

background: none repeat scroll 0 0 #F5F5F5;

border: 1px solid #DCDCDC;

-moz-border-radius: 2px 2px 2px 2px;

-webkit-border-radius: 2px 2px 2px 2px;

border-radius: 2px 2px 2px 2px;

color: #333333;

font: 11px/27px arial,sans-serif;

height: 27px;

padding: 0 8px;

text-align: center;

text-shadow: 0 1px 0 rgba(0, 0, 0, 0.1);

}

.form-submit:hover {

background-color: #F8F8F8;

border-color: #C6C6C6;

-moz-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.15);

-webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.15);

box-shadow: 0 1px 2px rgba(0, 0, 0, 0.15);

color: #333333;

}

.form-submit:active {

border-color: #4D90FE;

-webkit-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.3) inset;

-moz-box-shadow: 0 1px 2px rgba(0, 0, 0, 0.3) inset;

box-shadow: 0 1px 2px rgba(0, 0, 0, 0.3) inset;

color: #000000;

}

.form-submit:focus {

border: 1px solid #4D90FE !important;

}

你可以点击[这里](http://www.w3cplus.com/contact" \t "_blank)体验一下这种效果。

对于:hover在IE6下只有a元素支持，:active只有IE7-6不支持，:focus在IE6-7下不被支持。

二、UI元素状态伪类

我们把":enabled",":disabled",":checked"伪类称为UI元素状态伪类，这些主要是针对于HTML中的Form元素操作，最常见的比如我们"type="text"有enable和disabled两种状态，前者为可写状态后者为不可状态；另外"type="radio"和"type="checkbox""有"checked"和"unchecked"两种状态。来看两个实例，比如说你想将"disabled"的文本框与别的文本框区别出来，你就可以这样应用

input[type="text"]:disabled {border:1px solid #999;background-color: #fefefe;}

这样一来就把页面中禁用的文本框应用了一个不同的样式。那么对于其他几个用法是一样的，这里就不在讲述。IE6-8不支持":checked",":enabled",":disabled"这三种选择器。

三、CSS3的:nth选择器

这节内容才是关键，也是CSS3选择器最新部分，有人也称这种选择器为CSS3结构类，下面我们通过实际的应用来具体了解他们的使用和区别，首先列出他具有的选择方法：

:first-child选择某个元素的第一个子元素；

:last-child选择某个元素的最后一个子元素；

:nth-child()选择某个元素的一个或多个特定的子元素；

:nth-last-child()选择某个元素的一个或多个特定的子元素，从这个元素的最后一个子元素开始算；

:nth-of-type()选择指定的元素；

:nth-last-of-type()选择指定的元素，从元素的最后一个开始计算；

:first-of-type选择一个上级元素下的第一个同类子元素；

:last-of-type选择一个上级元素的最后一个同类子元素；

:only-child选择的元素是它的父元素的唯一一个了元素；

:only-of-type选择一个元素是它的上级元素的唯一一个相同类型的子元素；

:empty选择的元素里面没有任何内容。

下面我们针对上面所列的各种选择器，一个一个来介绍：

1、:first-child

:first-child是用来选择某个元素的第一个子元素，比如我们这里的这个demo，你想让列表中的"1"具有与从不同的样式，我们就可以使用:first-child来实现：

.demo li:first-child {background: green; border: 1px dotted blue;}

在没有这个选择器出现之前，我们都需在要第一个li上加上一个不同的class名，比如说“first”，然后在给他应用不同的样式

.demo li.first {background: green; border: 1px dotted blue;}

其实这两种最终效果是一样的，只是后面这种，我们需要在html增加一个额外的class名，请看效果：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth1.png

IE6不支持:first-child选择器。

2、:last-child

:last-child选择器与:first-child选择器的作用类似，不同的是":last-child"选择是的元素的最后一个子元素。比如说，我们需要单独给列表最后一项一个不同的样式，我们就可以使用这个选择器，如：

.demo li:last-child {background: green; border: 2px dotted blue;}

这个效果和以前在列表上的“last”的class是一样的

.demo li.last {background: green; border: 2px dotted blue;}

他们效显示的效果都是一致的，如图所示；

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth2.png

3、:nth-child()

:nth-child()可以选择某个的一个或多个特定的子元素，你可以按这种方式进行选择：

:nth-child(length);/\*参数是具体数字\*/

:nth-child(n);/\*参数是n,n从0开始计算\*/

:nth-child(n\*length)/\*n的倍数选择，n从0开始算\*/

:nth-child(n+length);/\*选择大于length后面的元素\*/

:nth-child(-n+length)/\*选择小于length前面的元素\*/

:nth-child(n\*length+1);/\*表示隔几选一\*/

//上面length为整数

:nth-child()可以定义他的值（值可以是整数，也可以是表达式），如上面所示，用来选择特定的子元素，对于这个我们直接看实例，比我说的更好理解。

:nth-child(3),选择某元素下的第三个子元素，（这里的3可以是你自己需要的数字），比如说，我需要选择列表中的第三个li元素，那么我们可以直接这样使用：

.demo li:nth-child(3) {background: lime;}

效果如下所示：

http://cdn.w3cplus.com/sites/default/files/css3-selector-nth3.png

这种不式不能引用负值，也就是说li:nth-child(-3)是不正确的使用方法。

:nth-child(n)，其中n是一个简单的表达式，那么"n"取值是从“0”开始计算的，到什么时候结束我也不知道，如果你在实际应用中直接这样使用的话，将会选中所有子元素，比如说，在我们的demo中，你在li中使用":nth-child(n)",那么将选中所有的"li",如：

.demo li:nth-child(n) {background: lime;}

等于

.demo li {background: lime;}

他其实是这样计算的

n=0 －－》 没有选择元素

n=1 －－》 选择第一个li,

n=2 －－》 选择第二个li,后在的依此类推，这样下来就选中了所有的li

请看效果：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth4.png

请注意了，这里的“n”只能是"n"，不能使用其他字母代替，不然会没有任何效果的。

:nth-child(2n),这中方式是前一种的变身，我们可以选择n的2倍数，当然其中“2”可以换成你自己需要的数字，如：

.demo li:nth-child(2n) {background: lime;}

等于

.demo li:nth-child(even) {background: lime;}

我们来看一下其计算的过程：

n=0 －－》 2n=0 －－》 没有选中任何元素，

n=1 －－》 2n=2 －－》 选择了第二个li

n=2 －－》 2n＝4 －－》 选择了第四个li，后面的依此类推

如果是“2n”这样跟我们以使用"even"命名class定义样式，所起到的效果是一样的，如图所示：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth5.png

“:nth-child(2n)”也等于":nth-child(even)"效果。

:nth-child(2n-1),这个选择器是在":nth-child(2n)"基础上演变过来的，上面说了人是选择偶数，那么我们在他的基础上减去“1”就变成奇数选择，如：

.demo li:nth-child(2n-1) {background: lime;}

我们来看看其实现过程

n=0 －－》 2n-1=-1 －－》 也没有选中任何元素，

n=1 －－》 2n-1=1 －－》 选择第一个li

n=2 －－》 2n-1=3 －－》 选择第三个li,后面的依此类推

其实实现这种奇数效果，我们还可以使用":nth-child(2n+1)"和":nth-child(odd)",一起来看他们的效果

http://cdn.w3cplus.com/sites/default/files/css3-selector-nth6.png

:nth-child(n+5)这个选择器是选择从第五个元素开始选择，这里的数字你可以自己定义，如：

.demo li:nth-child(n+5) {background: lime;}

按前面的计算方法，我们来看看，

n=0 －－》 n+5=5 －－》 选中第5个li

n=1 －－》 n+5=6 －－》 选择第6个li，后面的就不列出来了，原理一样

你可以使用这种方法选择你需要开始选择的元素位置，也就是说换了数字，起始位置就变了，看下在的效果图：



:nth-child(-n+5)这种选择器刚好和上面的选择器相反，这个是选择第5个前面的，如：

.demo li:nth-child(-n+5) {background: lime;}

如果不清楚怎么一回事，你只要计算一下就明白了

n=0 －－》 -n+5=5 －－》 选择了第5个li

n=1 －－》 -n+5=4 －－》 选择了第4个li

n=2 －－》 -n+5=3 －－》 选择了第3个li

n=3 －－》 -n+5=2 －－》 选择了第2个li

n=4 －－》 -n+5=1 －－》 选择了第1个li

n=5 －－》 -n+5=0 －－》 没有选择任何元素

从上面的计算方法中，大家很清楚的知道是怎么得来的，最后我们一起看看效果吧：

http://cdn1.w3cplus.com/sites/default/files/css3-selector-nth8.png

:nth-child(4n+1)这种方法是实现隔几选一的效果，比如我们这里是隔三选一，如果你把"4"换成别的数字那就是另外的一种隔法了，比如这个实例

.demo li:nth-child(4n+1) {background: lime;}

我们主要来看其计算出来的结果

n=0 －－》4n+1=1 －－》选择了第一个li

n=1 －－》4n+1=5 －－》选择了第五个li

n=2 －－》4n+1=9 －－》选择了第九个li

效果如下

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth9.png

IE6-8和FF3-浏览器不支持":nth-child"选择器。

4、:nth-last-child()

":nth-last-child()"选择器和前面的":nth-child()"很相似，只是这里多了一个last，所以他起的作用就和前面的":nth-child"不一样了，他只要是从最后一个元素开始算，来选择特定元素。我们来看几个实例：

.demo li:nth-last-child(4) {background: lime;}

上面代码表示选择倒数第四个列表项，效果如下：

http://cdn2.w3cplus.com/sites/default/files/css3-selector-nth10.png

其中":nth-last-child(1)"和":last-child"所起作用是一样的，都表示的是选择最后一个元素。

另外":nth-last-child()"也可以像“:nth-child()”一样，可以使用表达式来选择特定元素，下面我们来看几个特殊的表达式所起的作用

:nth-last-child(2n),这个表示的是从元素后面计算，选择的是偶数个数，从而反过来说就是选择元素的奇数，和前面的":nth-child(2n+1)",":nth-child(2n-1)",":nth-child(odd)"所起的作用是一样的。如：

.demo li:nth-last-child(2n) {background: lime;}

.demo li:nth-last-child(even) {background: lime;}

等于

.demo li:nth-child(2n+1) {background: lime;}

.demo li:nth-child(2n-1) {background: lime;}

.demo li:nth-child(odd) {background: lime;}

请看效果：



:nth-last-child(2n-1)这个选择器刚好跟上面的相反，从后面计算选择的是奇数，而从前面计算选择的就是偶数位了，这个前面的":nth-child(2n)"之类是相同的效果，如：

.demo li:nth-last-child(2n+1) {background: lime;}

.demo li:nth-last-child(2n-1) {background: lime;}

.demo li:nth-last-child(odd) {background: lime;}

等于：

.demo li:nth-child(2n) {background: lime;}

.demo li:nth-child(even) {background: lime;}

效果如下

http://cdn1.w3cplus.com/sites/default/files/css3-selector-nth12.png

看了这几个实例，大家都知道":nth-last-child()"和"nth-child()"使用方法是一样的，只不过他们的区别是“:nth-child()”是从元素的第一个开始计算，而“:nth-last-child()”是从元素的最后一个开始计算，他们的计算方法都是一样的。同样在IE6-8和FF3.0-浏览器不支持“:nth-last-child()”选择器。

5、:nth-of-type

:nth-of-type类似于:nth-child，不同的是他只计算选择器中指定的那个元素,其实我们前面的实例都是指定了具体的元素，这个选择器主要对用来定位元素中包含了好多不同类型的元素是很有用处，比如说，我们div.demo下有好多p元素，li元素，img元素等，但我只需要选择p元素，并让他每隔一个p元素就有不同的样式，那我们就可以简单的写成：

.demo p:nth-of-type(even) {background-color: lime;}

其实这种使用和:nth-child使用是一样的，也可以使用:nth-child的那些表达式和使用方法，唯一不同的是这种指定了元素的类型而以。同样在IE6-8和FF3.0-浏览器不支持这种选择器

6、:nth-last-of-type

这个选择器不用说大家都能想得到了，他和前面的:nth-last-child一样使用，只是他指一了元素的类型而以。

同样在IE6-8和FF3.0-浏览器不支持这种选择器

7、:first-of-type和:last-of-type

:first-of-type和:last-of-type这两个选择器就类似于:first-child和:last-child;不同之处就是指定了元素的类型。

:nth-of-type,:nth-last-of-type;:first-of-type和:last-of-type实际意义并不是很大，我们前面讲的:nth-child之类选择器就能达到这此功能，不过大家要是感兴趣还是可以了解一下，个人认为实用价值并不是很大。此类说法仅供参考。

8、:only-child和:only-of-type

":only-child"表示的是一个元素是它的父元素的唯一一个子元素。我们一起来看一个实例更好理解

<div class="post">

<p>Lorem ipsum dolor sit amet, consectetur</p>

<p>Lorem ipsum dolor sit amet, consectetur</p>

</div>

<div class="post">

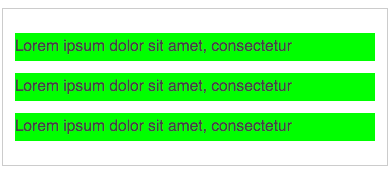
<p>Lorem ipsum dolor sit amet, consectetur</p>

</div>

css样式

.demo .post p {background: lime;}

初步效果

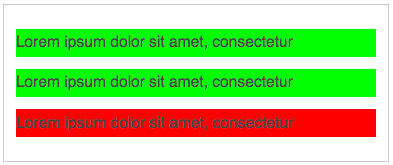


如果我需要在div.post只有一个p元素的时候，改变这个p的样式，那么我们现在就可以使用:only-child,如：

.demo .post p {background: lime;}

.demo .post p:only-child {background: red;}

此时只有div.post只有一个子元素p时，那么他的背景色将会改变,如图所示：



:only-of-type是表示一个元素他有很多个子元素，而其中只有一个子元素是唯一的，那么我们使用这种选择方法就可以选择中这个唯一的子元素，比如说

<div class="post">

<div>Lorem ipsum dolor sit amet, consectetur</div>

<p>Lorem ipsum dolor sit amet, consectetur</p>

<div>Lorem ipsum dolor sit amet, consectetur</div>

</div>

如果我们想只选择中上面中的p元素，我们就可以这样写，

p:only-of-type{background-color:red;}

IE6-8浏览器不支持:only-child选择器；IE6-8和FF3.0-浏览器不支持:only-of-type选择器。

9、:empty

:empty是用来选择没有任何内容的元素，这里没有内容指的是一点内容都没有，哪怕是一个空格，比如说，你有三个段落，其中一个段落什么都没有，完全是空的，你想这个p不显示，那你就可这样来写：

p:empty {display: none;}

IE6-8浏览器不支持:empty选择器

三、否定选择器（:not）

否定选择器和jq中的:not选择器一模一样，就拿form中的元素来说明这个选择器的用法，比如你想对form中所有input加边框，但又不想submit也起变化，此时就可以使用:not为实现

input:not([type="submit"]) {border: 1px solid red;}

否定选择器 :not()，可以让你定位不匹配该选择器的元素。IE6-8浏览器不支持:not()选择器

四、伪元素

CSS中的伪元素大家以前看过：:first-line,:first-letter,:before,:after;那么在CSS3中，他对伪元素进行了一定的调整，在以前的基础上增加了一个“：”也就是现在变成了“::first-letter,::first-line,::before,::after”另外他还增加了一个“::selection”,两个“：：”和一个“：”css3中主要用来区分伪类和伪元素，到目前来说，这两种方式都是被接受的，也就是说不管使用哪种写法所起的作用都是一样的，只是一个书写格式不同而以。

那么我们简单了解一下他们的作用

::first-line选择元素的第一行，比如说改变每个段落的第一行文本的样式，我们就可以使用这个

p::first-line {font-weight:bold;}

::first-letter选择文本块的第一个字母，除非在同一行里面包含一些其它元素，不过这个主要运用于段落排版上多，比如说首字下沉，

p::first-letter {font-size: 56px;float:left;margin-right:3px;}

::before和::after这两个主要用来给元素的前面或后面插入内容，这两个常用"content"配合使用，见过最多的就是清除浮动，

.clearfix:before,

.clearfix:after {

content: ".";

display: block;

height: 0;

visibility: hidden;

}

.clearfix:after {clear: both;}

.clearfix {zoom: 1;}

当然可以利用他们制作出其他更好的效果，比如本站首面的那个三角纸张效果，也是通过这个来实现的。

::selection用来改变浏览网页选中文的默认效果，有关于“::selection”使用，大家可以点击《[CSS ::Selection](http://www.w3cplus.com/content/css-selection" \t "_blank)》。

**CSS选择器方面的性能优化**

**浏览器如何识别你的选择器**

首先我们需要清楚，浏览器是如何读取选择器，以识别样式，并将相应的样式附于对应的HTML元素，达到美化页面的效果。[Chris Coyier](http://www.css-tricks.com/" \t "_blank)曾在《[Efficiently Rendering CSS](http://css-tricks.com/6386-efficiently-rendering-css/" \t "_blank)》一文中说过“**浏览器读取你的选择器，遵循的原则是从选择器的右边到左边读取。换句话说，浏览器读取选择器的顺序是由右到左进行**”。比如说：

div.nav > ul li a[title]

上面的实例来说，浏览器首先会尝试在你的HTML标签中寻找“**a[title]**”元素，接着在匹配**“li和ul”**，最后在去匹配“**div.nav**”。这就是前成所主的“**选择器从右到左的原则**”。

选择器的最后一部分，也就是选择器的最右边（在这个例子中就是a[title]部分）部分被称为“**关键选择器**”，它将决定你的选择器的效率如何？是高还是低。

老版本的浏览器可以过滤掉不匹配的选择器，而直接匹配更高效的选择器。记得David Hyatt在《[Writing efficient CSS for use in the Mozilla UI](https://developer.mozilla.org/en/Writing_Efficient_CSS" \t "_blank)》说过：“这个关键选择器可以大大提高选择器的性能，少检查一个给定的元素规则，就可以更有效的将样式匹配给对应的HTML元素。”那么如何让关键选择器更有效，性能化更高呢？其实很简单，主要把握一点“**越具体的关键选择器，其性能越高**”那么什么样类型的选择器，其性能高？什么样的类型的选择器性能低呢？

**CSS选择器的效率**

如果你阅读了本站的有关于选择器类型的介绍的话，你对选择器并不会感到陌生。就算你没读过，我想CSS选择器不会让我们觉得是新东西，比如我们常用的基本选择器“元素标签选择器div”、“id选择器#header”、“类选择器.class”，或者说我们很少见的伪类选择器“:focus”以及更复杂的css3选择器“:nth-child”等等。

选择器有一个固有的效率，我们来看[Steve Souders](http://stevesouders.com/)给排的一个顺序：

1. id选择器（#myid）
2. 类选择器（.myclassname）
3. 标签选择器（div,h1,p）
4. 相邻选择器（h1+p）
5. 子选择器（ul > li）
6. 后代选择器（li a）
7. 通配符选择器（\*）
8. 属性选择器（a[rel="external"]）
9. 伪类选择器（a:hover,li:nth-child）

上面九种选择器的效率是从高到低排下来的，基中ID选择器的效率是最高，而伪类选择器的效率则是最底。详细的介绍大家还可以点击[Writing efficient CSS selectors](http://csswizardry.com/2011/09/writing-efficient-css-selectors/" \t "_blank)。

**综合上面的顺序，我们清楚的知道，id和类名用于关键选择器上效率是最高的，而CSS3的仿伪类和属性选择器，虽然使用方便，但其效率却是最低的。**我们下面一起来看几个实例的对比：

**div *#myid***

效率要比下面的高：

***#myid div***

第一种选择器比第二种选择器效率高，大家或许会问为什么？其实根据前面所介绍的我们就不难理解了，因为第一个选择器的“关键选择器”使用了“ID选择器”，而第二个选择器的“关键选择器”使用的是“标签选择器”，对比下来，“ID选择器”效率高过“标签选择器”，所以说第一个选择器的效率要高于第二个选择器。

在类名或ID名前面加上标签也会致使选择器效率变低的，比如说：

***#myid***

**.myclassname**

上面两个选择的效率要高于下面的选择器：

**p*#myid***

**p.myclassname**

**来自Mozilla的几点建议**

David在《[Use efficient CSS selectors](http://code.google.com/speed/page-speed/docs/rendering.html" \t "_blank)》中介绍了几种书写高效率的CSS选择器的方法，下面我将他们移到这里来让大家参考：

1. 避免普遍规则
2. 不要在ID选择器前加标签名或类名
3. 不要在类名选择器前加标签名
4. 尽可能使用具体的类别
5. 避免使用后代选择器
6. 标签分类规则中不应该包含一个子选择器
7. 子选择器的问题
8. 借助相关继承关系
9. 使用范围内的样式表

如果你不够清楚上面所讲的是什么，你可以点击[这里](https://developer.mozilla.org/en/Writing_Efficient_CSS)，他会让你更容易了解这些规则。

**我们应该怎么做**

前面说“ID选择器”的效率是最高的，那么今天我们写样式，为了提高选择器的效率，是不是我们要在每一个文档的HTML元素中都加入ID名呢？我想这样的做法是没有的。对于一个有语义的代码编写和如何提高性能，以前他们之间如何的平衡？其实这个选择器的效率低一点，对于大多数网站来说并不会有太大的影响，但对于一个大型的网站，产生大量的流量这就会有差别了，也就很值得我们去对他进行优化。那么我们就很有必要的去了解他们是如何工作，比如说，一般情况下哪些选择器的使用效率更高。来看两个简单的例子：

***#myid***

上面的选择器高于下面的：

**p*#myid***

后者的写法我发现还是有很多朋友这样写，但我不知道你为什么需要在ID前面加一个标签？难道你同一个页面会有多个相同的ID不成？

我们接下来在来看一个实例，用于列表上的，比如说我们制作导航菜单的：

***#nav a***

高效于：

***#nav li a***

上面只是介绍了两个常碰到的实例，在这里说这两个实例，主要目的是让你在今后的编写样式时，能注意这方面的的细节，从而加快你的代码效率。

**测试你的选择器**

[Steve Souders](http://stevesouders.com/)给我们提供了一个[选择器效率测试](http://stevesouders.com/efws/css-selectors/csscreate.php" \t "_blank)，您 可以测试不同的选择器，比较他们之间的效率。

我们一起来看一个[Jon Sykes](http://blog.archive.jpsykes.com/)在前几年对后代选择器性能做的测试的对比：

直接类选择器：

.tdxx{

background: red;

}

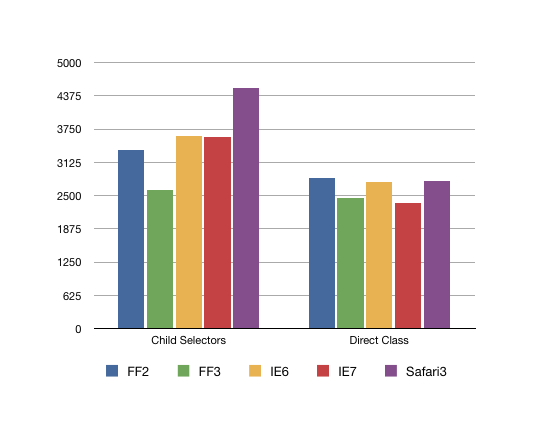
与后代选择器：

table tr td.tdxx{

background: red;

}

上面两个选择器所做的性能测试结果：



我想上面的图表就能很清楚的说明问题所在了。（注：上面的测试是几年前的数据；纵向是时间ms，模向前面是后代选择器，后面是直接类选择器）。如果你对这个感兴趣可以参阅：

1. [Testing CSS Performance](http://blog.archive.jpsykes.com/151/testing-css-performance/)
2. [Testing CSS Performance (pt 2)](http://blog.archive.jpsykes.com/152/testing-css-performance-pt-2/)
3. [More CSS Performance Testing (pt 3)](http://blog.archive.jpsykes.com/153/more-css-performance-testing-pt-3/)

上面扯了这么多，其实就是想说，CSS的选择器对一个网站的性能也是有关系的，学习之后，希望大家在平时的编写之时能尽量克服一些没必要的错误，从而提高这方面的性能。希望这篇文章对大家有所帮助。

**参考阅读：**

1. [Performance Impact of CSS Selectors](http://www.stevesouders.com/blog/2009/03/10/performance-impact-of-css-selectors/)
2. [Writing efficient CSS selectors](http://csswizardry.com/2011/09/writing-efficient-css-selectors/)
3. [101 on Prototype CSS Selectors](http://blog.dynatrace.com/2009/11/30/101-on-prototype-css-selectors/)
4. [More on CSS Selector Performance](http://css-tricks.com/7300-more-on-css-selector-performance/)
5. [Don’t use IDs in CSS selectors?](http://oli.jp/2011/ids/)
6. [On CSS Selectors and Performance](http://dilated.cc/css-selectors)
7. [Why you should deeply nest your CSS selectors.](http://coryschires.com/why-you-should-deeply-nest-your-css-selectors/)
8. [Selector Performance](http://smacss.com/book/selectors)
9. [CSS performance, who cares?](http://www.thebrightlines.com/2010/07/28/css-performance-who-cares/)
10. [The 30 CSS Selectors you Must Memorize](http://net.tutsplus.com/tutorials/html-css-techniques/the-30-css-selectors-you-must-memorize/)