**天池洗髓：CSS选择器之复合选择器**

上文讲了基础选择器，它们的功能比较单一，不能覆盖所有的业务需求。我们就来讲一下复合选择器，复合选择器可以说是CSS中经常使用的内容了。复合选择器是通过对基础选择器的组合，让选择器变得功能更强大。在样式开发中，复合选择器是最常见的，这一节的内容也是选择器这部分知识的重点。到实际开发时，我们就能感受到复合选择器到底有多常见。

## **后代选择器**

后代选择器的语法是用空格分隔的多个选择器组合，它的作用是在 A 选择器的后代元素中找到 B 选择器所指的元素。它的语法形式就是：“选择器 A 选择器 B” ，例如：

<!-- HTML -->

<div class="page">

<div class="article">

<h1>我是标题部分</h1>

<p>黑化肥发灰，灰化肥发黑</p>

<p>黑化肥发灰会挥发；灰化肥挥发会发黑</p>

<p>黑化肥挥发发灰会花飞；灰化肥挥发发黑会飞花</p>

</div>

<p class="footer">版权 © tuituitech.com</p>

</div>

// CSS:

.page p{

font-size: 18px;

}

上面的例子中就是一个标准的后代选择器，“.page p” 就表示在 .page 选定的区域里去找 p 标签。上面的后代选择器就会把 class 为 page 元素里面的所有 p 元素的字体都改成 18px。后代选择器最主要的目的是在给元素加一个范围的限制，上面例子中如果直接用一个标签选择器 p 效果也是一样的，但这样容易影响到页面里其他部分的 p 元素，所以前面加个 .page 给它限定住。

我们要注意，后代选择器是可以叠加使用的。假如我们只想限定 class=“article” 元素里的 p 标签字号是 18px，也可以用下面的方式：

// CSS:

.page .article p{

font-size: 18px;

}

使用这样的选择器以后，.article 元素外部的 p 标签，就不会应用这一条样式了。刚才这个例子中的选择器会有效率问题，后面的章节我们会讲到这么用有什么问题。

Tips：后代选择器通常用来限制选择器生效的范围，防止因为选择器使用不当或者对元素命名出现重复造成的样式冲突。

## **子元素选择器**

子元素选择器和后代选择器类似，也是为选择器限定范围。不同的是子元素选择器只找子元素，而不会把所有的后代都找一遍。它的语法是 “选择器 A> 选择器 B”，例如：

<!-- HTML -->

<div class="page">

<div class="article">

<h1>我是标题部分</h1>

<p>黑化肥发灰，灰化肥发黑</p>

<p>黑化肥发灰会挥发；灰化肥挥发会发黑</p>

<p>黑化肥挥发发灰会花飞；灰化肥挥发发黑会飞花</p>

</div>

<p class="footer">版权 © tuituitech.com</p>

</div>

// CSS:

.page > p{

color: grey;

}

用 “.page> p” 这个选择器就会把 .page 元素里面的 .footer 元素的字体变成灰色，但不会对 .article 元素下的 p 元素产生影响。

子元素选择器也是可以叠加使用的，还按上面的例子，假如我们想使用子元素选择器仅设置 .article 元素里面的 p 元素样式，我们就可以用下面的选择器来实现：

// CSS:

.page > .article > p{

color: black;

}

这样用叠加的子元素选择器，就可以只改变 .article 元素里面 p 元素的字体颜色了。

Tips：  
子元素选择器的作用和后代选择器相似，也是用来限制选择器生效的范围。它和后代选择器不同的是：  
1. 子元素选择器只匹配子元素，不会匹配后代元素。在有确定的父子关系时，尽量使用子元素选择器，效率会比后代选择器高。  
2. 使用子元素选择器还可以避免对非直接后代的样式影响，在只想给子元素设置样式时会比后代选择器安全。

## **兄弟选择器**

在 CSS 中，还有一种选择器是用来选取同级元素的，叫做兄弟选择器。兄弟选择器有两种，一种是相邻兄弟选择器，另外一种是通用兄弟选择器。下面我们来对两种兄弟选择器分开来介绍。

****一、相邻兄弟选择器****

相邻兄弟选择器是用来选取某个元素紧邻的兄弟元素，它的语法是 “选择器 A + 选择器 B”，表示找到与 A 元素相邻的 B 元素。其实就是对选择器 B 加上 “紧邻着选择器 A” 的限制。一下面的代码为例：

<!-- HTML -->

<div class="article">

<p>黑化肥与灰化肥</p>

<h1>我是标题部分</h1>

<p>黑化肥发灰，灰化肥发黑</p>

<p>黑化肥发灰会挥发；灰化肥挥发会发黑</p>

<p>黑化肥挥发发灰会花飞；灰化肥挥发发黑会飞花</p>

</div>

// CSS

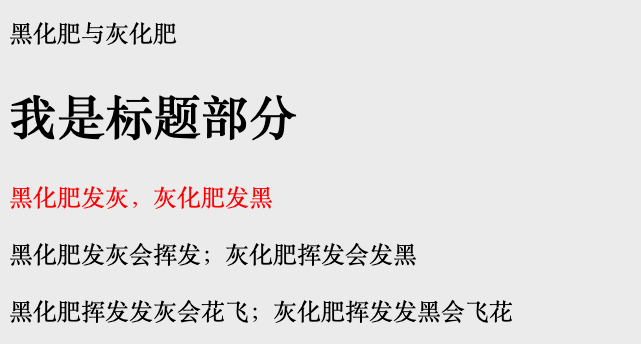
h1 + p{

margin-top: 10px;

color: red;

}

上面的例子中，h1 + p 选择器就表示选择紧邻 h1 元素的 p 元素，让这个 p 元素距离标题隔开 10px，并且字体设置为红色。这里要注意，相邻选择器只能选择紧挨在 h1 ****后面****的 p 元素，而不能向前找。下面就是运行的结果：

  
从例子中可以看到，只有紧挨着 h1 元素的 p 标签有了 10px 的上边距，而且字体变红了。

Tips：  
相邻兄弟选择器通常有两类用处：  
1. 用于自动调整占位，比如后面在布局的时候，有 header 和没 header 情况下内容区的高度会不同，就可以使用相邻兄弟选择器来控制内容区的高度。  
2. 相邻兄弟选择器的第二种用法是用来控制相同元素中间的间隔，比如在 List 组件开发时，每个 li 元素之间要加上分割线的需求就会通过相邻兄弟选择器来实现。

****二、通用兄弟选择器****

通用兄弟选择器和相邻兄弟选择器很相似，它的语法是 “选择器 A ~ 选择器 B”，也是用选择器 A 做限制，选择器 B 是最终匹配的目标。不同的是通用兄弟选择器会匹配选择器 A 指定元素后面的所有符合选择器 B 规则的元素。例如：

<!-- HTML -->

<div class="article">

<p>黑化肥与灰化肥</p>

<h1>我是标题部分</h1>

<p>黑化肥发灰，灰化肥发黑</p>

<p>黑化肥发灰会挥发；灰化肥挥发会发黑</p>

<p>黑化肥挥发发灰会花飞；灰化肥挥发发黑会飞花</p>

</div>

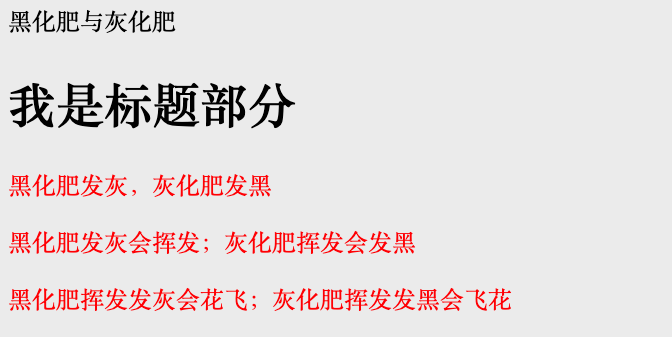
// CSS

h1 ~ p{

color: red;

}

上面的选择器会选中和 h1 元素同级且在 h1 元素后面出现的 p 元素，并给它们加上红色的字体样式。这个例子的运行结果就是：

  
这个例子中使用了通用兄弟选择器，和相邻兄弟选择器不同的是，h1 标签后面所有的 p 标签字体颜色都变红了，但 h1 前面的 p 标签字体颜色并没有变化。

Tips：  
兄弟选择器（包括相邻兄弟选择器和通用兄弟选择器）中都是只能向后选择，如果需要向前选择，就只能给前面的元素指定上 class，再用类选择器来实现了。这里为什么不供向前寻找的方式，我们留个悬念，后面在讲渲染原理的时候再来分析。

## **交集选择器**

交集选择器是为了找两个或多个选择器的交集，用法就是把两个选择器放在一起，形式如 “选择器 A 选择器 B”，中间不需要加空格或者其他符号。交集选择器最主要的作用是在限定范围内标识特殊的样式。比如：

<!-- HTML -->

<div class="menu">

<a class="menu-item">菜单1</a>

<a class="menu-item active">菜单2</a>

<a class="menu-item">菜单3</a>

<a class="menu-item">菜单4</a>

</div>

// CSS

.menu-item{

background: #ccc;

color: #000;

}

.menu-item.active{

background: #aaa;

color: #fff;

}

上面的例子是一个简单的菜单，通过 .menu-item 给所有菜单元素一个基础样式，然后通过交集选择器 .menu-item.active 给当前活跃的菜单选项一个特殊的标记。在这个例子里，交集选择器有两个意思，一方面是把选择器限定在 .menu-item 范围内，另一方面是选中范围内比较特殊的 .active 元素。

## **并集选择器**

并集选择器是为了合并类似的样式，可以把选择器不同但样式相同的 CSS 语法块做合并。并集选择器就是用逗号分割多个选择器，形式如 “选择器 A, 选择器 B”，表示该样式对选择器 A 和选择器 B 所选择的元素都生效。例如：

// CSS

h1{

margin: 0;

padding: 0;

}

h2{

margin: 0;

padding: 0;

}

h3{

margin: 0;

padding: 0;

}

上面的样式是希望把 h1，h2 和 h3 的内外边距都归零，来解决不同浏览器中元素默认的内外边距不一致的问题。这种写法会占用很多 CSS 代码量，但是可以通过并集选择器来进行简化：

h1, h2, h3{

margin: 0;

padding: 0;

}

这样通过一条 CSS 规则就替代了上面分开写的一大坨代码了，它们的效果是完全一样的。

其实这一节最重要的就是CSS组合中的~和+，这两个尤其常见。比如：对一个页面来说，我们都知道一般有header，可能在web-app中还有search，现在我们想要实现CSS的适配：如果用户写了header，那么search部分距离顶部xxxpx，但如果用户这个页面没有header，那么search距离顶部就是0：

用~就完全可以轻易实现 —— 因为header和search很容易发现他们都是“body”（或者说“HTML”）的子元素，他们同级且search在header后面：

/\* 搜索状态中，内容覆盖区域 \*/

.tt-search.on-focus{

position: fixed;

left: 0;

right: 0;

top: 0;

bottom: 0;

overflow-y: auto;

}

/\* 处理有标题栏的情况 \*/

.tt-header ~ .tt-search.on-focus{

top: 2.3rem;

}

CSS中还有一个神奇的地方在于，选择器组合起来使用有时会有“别样的功能”：

简单的如：

/\* 控制展开按钮的显示 \*/

.tt-article.fold .tt-article-unfold-btn{

display: block;

}

当文本元素（class为tt-article和flod）具有fold类名时，其下的（这里为什么是“其下的”？因为这里组合样式为“空格”——父子选择器组合）tt-article-unfold-btn元素会显示出来。反之，如果用JS控制fold去掉，则此样式不会生效了

复杂的如：

/\* 取消按钮的class \*/

.tt-search > .tt-search-cancel{

flex: 0 0 2.2rem;

padding-right: .5rem;

text-align: center;

font-size: .7rem;

display: none;

}

/\* 对取消按钮的控制 \*/

.tt-search.on-focus > .tt-search-cancel{

display: block;

}

上面这段代码就很形象：在tt-search类名标签下有代表“取消”功能的标签，为tt-search-cancel，开始时它是处于未显示状态（display为none），**当手指（或鼠标）聚焦到search组件上时（表现为通过JS为tt-search class中添加类名on-focus）**，就让tt-search-cancel显示出来（这里是display为block）。

这就是CSS组合选择器对于“**状态改变**”的支持！