21-CSS选择器：伪元素是怎么回事儿？

你好，我是winter。

在上一篇文章中，我已经给你介绍了一些简单选择器，这一节课我会继续给你介绍选择器的几个机制：选择器的组合、选择器的优先级和伪元素。

选择器的组合

在CSS规则中，选择器部分是一个选择器列表。

选择器列表是用逗号分隔的复杂选择器序列；复杂选择器则是用空格、大于号、波浪线等符号连接的复合选择器；复合选择器则是连写的简单选择器组合。

根据选择器列表的语法，选择器的连接方式可以理解为像四则运算一样有优先级。

- 第一优先级
  - 无连接符号
- 第二优先级
  - “空格”
  - “~”
  - “+”
  - “>”
  - “||”
- 第三优先级
  - “,”

例如以下选择器：

```
.c,.a>.b.d {
    /*......*/
}
```

我们应该理解为这样的结构。

- .c,.a>.b.d
  - .c
  - .a>.b.d
    - .a
    - .b.d
      - .b

- .d

复合选择器表示简单选择器中“且”的关系，例如，例子中的“ .b.d ”，表示选中的元素必须同时具有b和d两个class。

复杂选择器是针对节点关系的选择，它规定了五种连接符号。

- **“空格”**：后代，表示选中所有符合条件的后代节点， 例如“ .a .b ”表示选中所有具有class为a的后代节点中class为b的节点。
- **“>”**：子代，表示选中符合条件的子节点，例如“ .a>.b ”表示：选中所有“具有class为a的子节点中，class为b的节点”。
- **“~”**：后继，表示选中所有符合条件的后继节点，后继节点即跟当前节点具有同一个父元素，并出现在它之后的节点，例如“ .a~.b ”表示选中所有具有class为a的后继中，class为b的节点。
- **“+”**：直接后继，表示选中符合条件的直接后继节点，直接后继节点即nextSlibling。例如 “.a+.b ”表示选中所有具有class为a的下一个class为b的节点。
- **“||”**：列选择器，表示选中对应列中符合条件的单元格。

我们在实际使用时，比较常用的连接方式是“空格”和“>”。

工程实践中一般会采用设置合理的class的方式，来避免过于复杂的选择器结构，这样更有利于维护和性能。

空格和子代选择器通常用于组件化场景，当组件是独立开发时，很难完全避免class重名的情况，如果为组件的最外层容器元素设置一个特别的class名，生成CSS规则时，则全部使用后代或者子代选择器，这样可以有效避免CSS规则的命名污染问题。

逗号表示“或”的关系，实际上，可以把它理解为“两条内容一样的CSS规则”的一种简写。如我们开头的例子，可以理解成与下面的代码等效：

```
.c {
    /*......*/
}
.a>.b.d {
    /*......*/
}
```

到这里，我们就讲完了如何用简单选择器组合成复合选择器和复杂选择器，形成选择器列表，这能够帮助我们应对各种复杂的需求。

CSS选择器是基于规则生效的，同一个元素命中多条规则是非常常见的事情。不同规则指定同一个属性为不同值时，就需要一个机制来解决冲突。这个机制，就是接下来我们要讲的选择器优先级。

选择器的优先级

CSS标准用一个三元组 (a, b, c) 来构成一个复杂选择器的优先级。

- id选择器的数目记为a；
- 伪类选择器和class选择器的数目记为b；
- 伪元素选择器和标签选择器数目记为c；
- “*” 不影响优先级。

CSS标准建议用一个足够大的进制，获取“ a-b-c ”来表示选择器优先级。

即：

```
specificity = base * base * a + base * b + c
```

其中，base是一个“足够大”的正整数。关于base，历史中有些趣闻，早年IE6采用256进制，于是就产生“256个class优先级等于一个id”这样的奇葩问题，后来扩大到65536，基本避免了类似的问题。

现代浏览器多采用了更大的数量，我们正常编写的CSS规则数量不太可能达到数万，因此我们可以认为这样的base就足够大了。

行内属性的优先级永远高于CSS规则，浏览器提供了一个“口子”，就是在选择器前加上“!import”。

这个用法非常危险，因为它相当于一个新的优先级，而且此优先级会高于行内属性。

同一优先级的选择器遵循“后面的覆盖前面的”原则，我们可以看一个例子：

```
<div id="my" class="x y">text<div>
```

```
.x {
    background-color:lightblue;
}
.y {
    background-color:lightgreen;
}
```

调换“.x”和“.y”我们可以得到不同的显示效果。选择器的优先级是针对单条规则的，多条规则的选择器同时命中元素，优先级不会发生叠加。

```
<div id="my" class="x y z">text<div>
```

```
.x {
    background-color:lightblue;
}
.z {
    background-color:lightblue;
}
.y {
    background-color:lightgreen;
}
```

在这个例子中，“.x ”和“.z ”都指定了背景色为浅蓝色，但是因为“.y ”规则在最后，所以最终显示结果为浅绿色。另外一个需要注意的是，选择器的优先级是针对复杂选择器的优先级，选择器列表不会合并计算优先级。

我们看一个例子：

```
<div id="my" class="x y z">text<div>
```

```
.x, .z {
    background-color:lightblue;
}
.y {
    background-color:lightgreen;
}
```

这里选择器列表“ .x, .z”命中了div，但是它的两项分别计算优先级，所以最终优先级仍跟“ .y” 规则相同。

以上就是选择器优先级的相关规则了，虽然我们这里介绍了详细的计算方式，但是我认为选择器的使用上，如果产生复杂的优先级计算，代码的可读性一定是有问题的。

所以实践中，建议你“根据 id 选单个元素”“class和class的组合选成组元素”“tag选择器确定页面风格”这样的简单原则来使用选择器，不要搞出过于复杂的选择器。

伪元素

在上一课，我们有意忽略了一种重要的简单选择器：伪元素。

我之所以没有把它放在简单选择器中，是因为伪元素本身不单单是一种选择规则，它还是一种机制。

所以本节课，我就来讲一讲伪元素机制。伪元素的语法跟伪类相似，但是实际产生的效果却是把不存在的元

素硬选出来。

目前兼容性达到可用的伪元素有以下几种。

- ::first-line
- ::first-letter
- ::before
- ::after

下面我们就来分别讲讲它们。

::first-line 和 ::first-letter 是比较类似的伪元素，其中一个表示元素的第一行，一个表示元素的第一个字母。

我们可以看一个示例：

```
<p>This is a somewhat long HTML
paragraph that will be broken into several
lines. The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

```
p::first-line {
    text-transform: uppercase
}
```

这一段代码把段落的第一行字母变为大写。注意这里的第一行指的是排版后显示的第一行，跟HTML代码中的换行无关。

::first-letter 则指第一个字母。首字母变大并向左浮动是一个非常常见的排版方式。

```
<p>This is a somewhat long HTML
paragraph that will be broken into several
lines. The first line will be identified
by a fictional tag sequence. The other lines
will be treated as ordinary lines in the
paragraph.</p>
```

```
p::first-letter {
    text-transform: uppercase;
```

```
    font-size:2em;
    float:left;
}
```

虽然听上去很简单，但是实际上，我们遇到的HTML结构要更为复杂，一旦元素中不是纯文本，规则就变得复杂了。

CSS标准规定了first-line必须出现在最内层的块级元素之内。因此，我们考虑以下代码。

```
<div>
  <p id=a>First paragraph</p>
  <p>Second paragraph</p>
</div>
```

```
div>p#a {
    color:green;
}

div::first-line {
    color:blue;
}
```

这段代码最终结果第一行是蓝色，因为p是块级元素，所以伪元素出现在块级元素之内，所以内层的color覆盖了外层的color属性。

如果我们把p换成span，结果就是相反的。

```
<div>
  <span id=a>First paragraph</span><br/>
  <span>Second paragraph</span>
</div>
```

```
div>span#a {
    color:green;
}

div::first-line {
    color:blue;
}
```

这段代码的最终结果是绿色，这说明伪元素在span之外。

::first-letter的行为又有所不同，它的位置在所有标签之内，我们把前面的代码换成::first-letter。

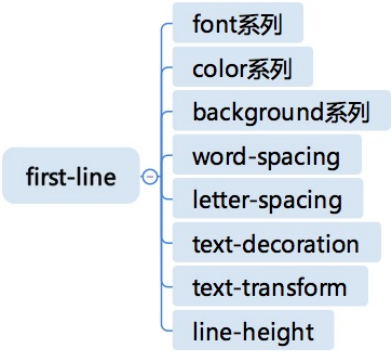
```
<div>
  <span id=a>First paragraph</span><br/>
  <span>Second paragraph</span>
</div>
```

```
div>span#a {
    color:green;
}

div::first-letter {
    color:blue;
}
```

执行这段代码，我们可以看到，首字母变成了蓝色，这说明伪元素出现在span之内。

CSS标准只要求 ::first-line 和 ::first-letter 实现有限的几个CSS属性，都是文本相关，这些属性是下面这些。



接下来我们说说 ::before 和 ::after 伪元素。

这两个伪元素跟前面两个不同的是，它不是把已有的内容套上一个元素，而是真正的无中生有，造出一个元素。

::before 表示在元素内容之前插入一个虚拟的元素，::after 则表示在元素内容之后插入。

这两个伪元素所在的CSS规则必须指定content属性才会生效，我们看下例子：

```
<p class="special">I'm real element</p>
```

```
p.special::before {
    display: block;
    content: "pseudo! ";
}
```

这里要注意一点，::before 和 ::after 还支持content为counter，如：

```
<p class="special">I'm real element</p>
p.special::before {
    display: block;
    content: counter(chapno, upper-roman) ". ";
}
```

这对于实现一些列表样式是非常有用的。

::before 和 ::after 中支持所有的CSS属性。实际开发中，这两个伪元素非常有用，有了这两个伪元素，一些修饰性元素，可以使用纯粹的CSS代码添加进去，这能够很好地保持HTML代码中的语义，既完成了显示效果，又不会让DOM中出现很多无语义的空元素。

总结

这一课，我们讲了CSS选择器的三种机制：选择器的组合、选择器优先级、以及伪元素。

在选择器组合这一部分，我们讲到了，选择器的连接方式像四则运算一样有优先级，

第一优先级是无连接符号；第二优先级是：“空格”“~”“+”“>”“||”；第三优先级是“,”。

然后我们又介绍了选择器优先级的计算方式。

最后我们为大家介绍了伪元素，我们逐次讲解了

- ::first-line
- ::first-letter
- ::before
- ::after

四种伪元素。伪元素的语法跟伪类相似，但是实际产生的效果是把不存在的元素硬选出来。这一点就与伪类不太一样了。

结合上一节课我们讲的简单选择器，对它们灵活运用，就能够满足大部分CSS的使用场景的需求了。

最后，留给你一个问题，你所在的团队，如何规定CSS选择器的编写规范？你觉得它好吗？

猜你喜欢

精选留言：

- Scorpio 2019-03-08 08:09:32
  我们团队没有规范。。。 [6赞]

- 阿成 2019-03-07 07:49:00
  有两个问题想请教一下winter老师：
  1. 您对styled-component类似的方案怎么看
  2. 您对使用属性选择器代替class怎么看 [5赞]

- Carson 2019-03-07 02:33:38
  如果是注重复用的开发，一般采用组件化的形式，给组件一套命名空间；

  如果是页面较少的网页开发，不太在意复用和扩展，一般采用 BEM 的规则。

  ”根据 id 选单个元素，class 和 class 的组合选择成组元素，tag 选择器确定页面风格。“ 从这个原则中收获很大。
  [4赞]

- 阿歡。 2019-03-13 22:47:18
  老师您好,下面例子中 把
去掉，会变成First paragraph为绿色，Second paragraph为蓝色，这是为何？

  First paragraph
  Second paragraph

  div>span#a {
  color:green;
  }
  div::first-line {
  color:blue;
  } [2赞]

- qqq 2019-03-22 11:38:17
  提醒下：伪元素那部分说的是子元素 color 覆盖父元素 color，而非 CSS 规则覆盖 [1赞]

- Lcina 2019-03-18 12:54:07

行内属性的优先级永远高于 CSS 规则，浏览器提供了一个“口子”，就是在选择器前加上“!import”。应该是 important 吧 [1赞]

- Geek_8c1d64 2019-03-07 01:14:39
  img、br等不能包含子元素的标签不能创建::before和::after。但一个例外是hr，不知道为什么。或许是我的理解有问题？ [1赞]

- 靠人品去赢 2019-04-03 15:49:25
  我放一个伪类和伪元素的链接吧，这两者属于见过但是没注意更没区分过，估计有人会需要https://developer.mozilla.org/zh-CN/docs/Learn/CSS/Introduction_to_CSS/Pseudo-classes_and_pseudo-elements

- Ranjay 2019-03-24 21:54:52
  BEM规范实际上就已经是很好的实践