

20 | CSS 选择器：如何选中svg里的a元素？

winter 2019-03-05



00:00

15:15

讲述：winter 大小：13.98M

你好，我是 winter。

我们在之前 CSS 语法课程中，已经介绍了关于选择器的一部分基础知识。在今天的这一课里，我们来系统学习一下 CSS 选择器。

在 CSS 语法课程中，我们已经见过一些选择器了，但在进入到具体的选择器介绍之前，我们首先要对选择器有一个整体的认识。

我先来讲讲选择器是什么，选择器是由 CSS 最先引入的一个机制（但随着 document.querySelector 等 API 的加入，选择器已经不仅仅是 CSS 的一部分了）。我们今天这一课，就重点讲讲 CSS 选择器的一些机制。

选择器的基本意义是：根据一些特征，选中元素树上的一批元素。

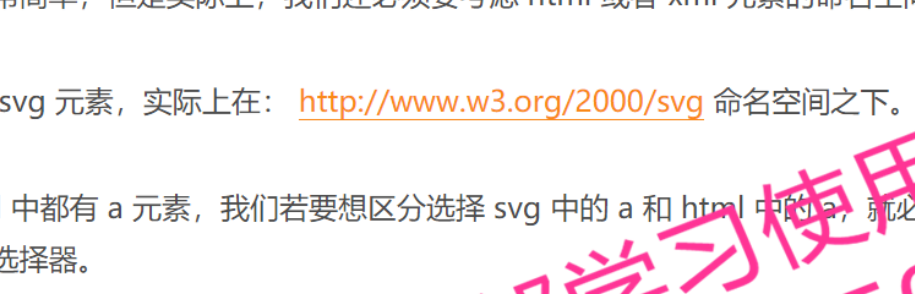
我们把选择器的结构分一下类，那么由简单到复杂可以分成以下几种。

- 简单选择器：针对某一特征判断是否选中元素。
- 复合选择器：连续写在一起的简单选择器，针对元素自身特征选择单个元素。
- 复杂选择器：由 “ ”（空格）“ > ” “ ~ ” “ + ” “ || ” 等符号连接的复合选择器，根据父元素或者前序元素检查单个元素。
- 选择器列表：由逗号分隔的复杂选择器，表示 “或” 的关系。

我们可以看到，选择器是由简单选择器逐级组合而成的结构，那么我们就来首先看一下简单选择器。

简单选择器

我们在前面说过，简单选择器是针对某一特征判断是否为选中元素。今天我会为你介绍一系列常见的简单选择器，我们把相似的简单选择器放在一起，这样更易于你去记忆。



类型选择器和全体选择器

我们要介绍的第一个简单选择器就是类型选择器，它根据一个元素的标签名来选中元素。

比如：

```
1  div {
2
3  }
4
```

这看上去非常简单，但是实际上，我们还必须要考虑 html 或者 xml 元素的命名空间问题。

比如我们的 svg 元素，实际上在：<http://www.w3.org/2000/svg> 命名空间之下。

svg 和 html 中都有 a 元素，我们若要想区分选择 svg 中的 a 和 html 中的 a，就必须用带命名空间的类型选择器。

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <meta charset="utf-8">
5    <title>JS Bin</title>
6  </head>
7  <body>
8    <svg width="100" height="28" viewBox="0 0 100 28" version="1.1"
9      xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
10     <desc>Example link01 - a link on an ellipse
11   </desc>
12   <a xlink:href="http://www.w3.org">
13     <text y="100%">name</text>
14   </a>
15 </svg>
16 <br/>
17 <a href="javascript:void 0;">name</a>
18 </body>
19 </html>
20
21 @namespace svg url(http://www.w3.org/2000/svg);
22 @namespace html url(http://www.w3.org/1999/xhtml);
23 svg|a {
24   stroke:blue;
25   stroke-width:1;
26 }
27
28 html|a {
29   font-size:40px
30 }
31
```

这里有一个特殊的选择器，就是 “ * ”，它称为全体选择器，可以选中任意元素。它的用法跟类型选择器是完全一致的，这里就把它们放到一起介绍了。

id 选择器与 class 选择器

id 选择器和 class 选择器都是针对特定属性的选择器。id 选择器是 “#” 号后面跟随 id 名，class 选择器是 “.” 后面跟随 class 名。我们来看看基本用法：

```
1  #myid {
2    stroke:blue;
3    stroke-width:1;
4  }
5
6  .mycls {
7    font-size:40px
8  }
9
```

这两个选择器都是在属性选择器之前就设计出来的选择器，属性选择器出来了以后，理论上可以一定程度上替代它们。但是要注意，class 选择器识别的是：用空格分隔的 class 语法。

```
1  <a class="a b c">xxx</a>
2  .a {
3    color:red;
4  }
5
```

在这个例子中，我们使用了用空格分隔的 class 属性，使用 “.a” “.b” 或者 “.c” 都能够选中元素，也可以使用多个 class 选择器来要求元素具有多个类。

属性选择器

属性选择器根据 HTML 元素的属性来选中元素。属性选择器有四种形态。

- 第一种，[att]

直接在方括号中放入属性名，是检查元素是否具有这个属性，只要元素有这个属性，不论属性是什么值，都可以被选中。

- 第二种，[att=val]

精确匹配，检查一个元素属性的值是否是 val。

- 第三种，[att~=val]

多种匹配，检查一个元素的值是否是若干值之一，这里的 val 不是一个单一的值了，可以用空格分隔的一个序列。

- 第四种，[att|=val]

开头匹配，检查一个元素的值是否是以 val 开头，它跟精确匹配的区别是属性只要以 val 开头即可，后面内容不管。

有些 HTML 属性含有特殊字符，这个时候，可以把 val 用引号括起来，形成一个 CSS 字符串。CSS 字符串允许使用单双引号来规避特殊字符，也可以用反斜杠转义，这样，就可以表示出任意属性值啦。

伪类选择器

接下来我们开始介绍伪类选择器，伪类选择器是一系列由 CSS 规定好的选择器，它们以冒号开头。伪类选择器有普通型和函数型两种。

我们首先来介绍一下伪类中最常用的部分：树结构关系伪类。

树结构关系伪类选择器

:root 伪类表示树的根元素，在选择器是针对完整的 html 文档情况，我们一般用 html 标签即可选中根元素。但是随着 scoped css 和 shadow root 等场景出现，选择器可以针对某一子树来选择，这时候就很可能需要 root 伪类了。

- :empty 伪类表示没有子节点的元素，这里有个例外就是子节点为空白文本节点的情况。

- :nth-child 和 :nth-last-child 这是两个函数型的伪类，CSS 的 An+B 语法设计的是比较复杂的，我们这里仅仅介绍基本用法。我们还是看几个例子：

选择器	效果
:nth-child(even)	选中偶数节点
:nth-child(4n-1)	选中第3个、第7个、第11个这样符合4的倍数减一的数字
:nth-child(3n+1 of li,important)	选中第4个、第7个、第10个...li,important 在这里只有li,important会被计数

- :nth-last-child 的区别仅仅是从后往前数。

- :first-child :last-child 分别表示第一个和最后一个元素。

- :only-child 按字面意思理解即可，选中唯一一个孩子元素。

of-type 系列，是一个变形的语法糖，S:nth-of-type(An+B) 是:nth-child([An+B] of S) 的另一种写法。

以此类推，还有 nth-last-of-type、first-of-type、last-of-type、only-of-type。

链接与行为伪类选择器

链接与行为是第一批设计出来的伪类，也是最常用的一批。

- :any-link 表示任意的链接，包括 a、area 和 link 标签都可能匹配到这个伪类。

- :link 表示未访问过的链接，:visited 表示已经访问过的链接。

- :hover 表示鼠标悬停在上的元素，

- :active 表示用户正在激活这个元素，如用户按下按钮，鼠标还未抬起时，这个按钮就处于激活状态

- :focus 表示焦点落在这个元素之上。

- :target 用于选中浏览器 URL 的 hash 部分所指示的元素。

在 Selector Level 4 草案中，还引入了 target-within、focus-within 等伪类，用于表示 target 或者 focus 的父容器。

逻辑伪类选择器

我们这里介绍一个逻辑伪类 —— :not 伪类。

这个伪类是个函数型伪类，它的作用时选中内部的简单选择器命中的元素。

```
1  *|*:not(:hover)
2
```

选择器 3 级标准中，not 只支持简单选择器，在选择器 4 级标准，则允许 not 接受一个选择器列表，这意味着选择器支持嵌套，仅靠 not 即可完成选择器的一阶真值逻辑完备，但目前还没有看到浏览器实现它。

在 Selector Level 4 草案中，还引入了 :is :where :has 等逻辑伪类，但是它们有一些违背了选择器匹配 DOM 树不回溯的原则，所以这部分设计最终的命运如何还不太确定。

其它伪类选择器

还有一些草案中或者不常用的选择器，你仅做大概了解即可。

- 国际化：用于处理国际化和多语言问题。

- dir

- lang

- 音频 / 视频：用于区分音视频播放状态。

- play

- pause

- 时序：用于配合读屏软件等时序性客户端的伪类。

- current

- past

- future

- 表格：用于处理 table 的列的伪类。

- nth-col

- nth-last-col

伪类是很大的一类简单选择器，它是选择器能力的一种补充。在实际使用中，我还是建议你尽量通过合适的 id 和 class 来标识元素，约束伪类的使用。最好只在不得不使用伪类的场景使用伪类，这对于 CSS 代码的性能和可读性都有好处。

结语

这一节课程中，我们介绍了 CSS 选择器的整体结构，并且介绍了一系列简单选择器。它们包括了下面这些内容。

- 类型选择器：根据一个元素的标签名来选中元素。

- 全体选择器：与类型选择器类似，选择任意元素。

- id 选择器：# 后面跟随 id 名。

- class 选择器：. 后面跟随 class 名。

- 伪类选择器：一系列由 CSS 规定好的选择器，它们以冒号开头。伪类有普通型和函数型。

在下一节课，我们开始进入到更复杂的情况，我们将会介绍选择器的组合使用方式和选择器的一些机制。

今天留给你的思考题是，用 JavaScript 实现一个能够处理所有简单选择器的 querySelector（行为伪类除外），你可以把你的答案分享出来，我们一起来探讨吧。