CSS 选择器

CSS 选择器是用来将 HTML 的元素拣选出来，然后套用对应的 CSS 样式。例如你想改变网页内超链接文字的颜色，调整不同段落的文字大小，又或者为图片加上圆角效果等等，可以透过内嵌 CSS 的写法将样式直接套用到相关的 HTML 元素里面。
栗子：

```
<img src=”某个图片” style=”border-radius:10px; width：20px;
height:20px”>
```

用 img 加一张图片，然后设定 style 属性，里面直接编写 CSS 样式,加个 border-radius 为图片加上圆角。

但这样过于麻烦，所以我们一般会将 CSS 与 html 分开来写，将 CSS 统一写在一个档案当中，然后再透过一种途径将这个 CSS 样式套用到对应的 HTML 元素上，这个途径就是今天要讲的 CSS 选择器。

添加 CSS 的方法：

1. 是在 HTML 中<head>里面添加<style>标签。
2. 在<head>中添加<link>标签，新建一个.css 文件：

<link rel=”stylesheet” type=”text/css” href=”文件名.css” />

CSS 语法：

```
选择器{
    属性名:属性值
}
```

举十种 CSS 选择器栗子：

1.ID 选取器

用#号开头然后自定一个名称，命名规则是：第一个字元一定要是英文字母（大小写皆可），第二个字元开始就可以是英文、数字、横线、底线等。
栗子:

```
#the-id-selector{
    font-size:24px;
    background-color:yellow;
    padding:8px;
}
```

即是选取拥有 id 这个属性。并且它的值是 the-id-selector 的 HTML 元素,ID 选取器有一个特点，就是它在整个 HTML 页面内必须是唯一的，一个 ID 只能用在一个 HTML 元素上，而一个 HTML 元素也只能有一个 id。

同时我们在 HTML 里加一个 ，将它的 id 设定为 the-id-selector,这样他们就关联在一起了！

```
<div id=”the-id-selector”>
    Coding Start
</div>
```

2. Class 选取器

这是最常用的选取器，使用.号开头，然后自定义一个名称，命名规则与 ID 选取器相同，第一个字元一定要是英文字母（大小写皆可），第二个字元开始就可以是英文、数字、横线、底线等。

给个栗子:

```
.my-class{
    font-family: Helvetica;
    font-size: 24px;
    color: yellow;
}
```

一个 class 可以套用到多个 HTML 元素上，一个 HTML 也可以有很多个 class。

栗子：

加一个叫 bg-blue 的 class

```
.bg-blue{
    background-color:blue;
}
```

此时在 HTML 中我们只需要将两个样式空格隔开即可同时运用

```
<div class=”my-class bg-blue”>
    Coding Startup
</div>
```

3. Tag 选取器

在 CSS 中直接使用 HTML 标签（Tag）的名称，例如<h1><p> 等等，会直接对应到 HTML 所有对应的标签中。

如果在 Tag 名称后加上 ID 选取器或 Class 选取器

栗子：

```
div.container{
    background-color:yellow;
}
```

即表示 container 这个 class 只能够套用到 div 上，如果我将它套用到其他 HTML 标签中则不会生效。

4. 空格

直接上栗子：

```
.container div{
```

```
    background-color: yellow;
}
```

既是选取拥有 container 这个 class 的 HTML 元素里所有 div，注意并不包括 container 本身。

5. 大于符号

栗子：

```
.container > div {
    background-color: yellow;
}
```

即是选取拥有 container 这个 class 的 HTML 元素里面的第一层的所有 div，我们可以透过画面中颜色的变化去理解这个层级关系。

6. 加号

栗子：

```
.container + div {
    background-color: yellow
}
```

即是选取拥有 coontainer 这个 class 的 HTML 元素与它身处同一层紧接着的 div，如果紧接着的不是 div，那此选择器就不会选取到任何东西。

7. 波浪符号

栗子：

```
.container ~ div {
    background-color: yellow
}
```

即是选取拥有 container 这个 class 的 HTML 元素与它身处同一层之后的所有 div，尽管其中有其它 HTML 元素隔开，只要是与 container 处于同一层都会套用得到。

8. 星号

*是任何 HTML 标签的意思，如果只用一个星号的话即是选取所有 HTML 元素，我们一般会配合刚才提到的空格、加号或者波浪符号去做进一步的筛选，缩小选区范围。

栗子：

会将 background-color 红色套用到 container 之后的 div 与 p 标签

```
.container ~ *{
    background-color: red!important;
}
```

9. 属性选择器

是透过 HTML 元素的属性去选取。

栗子：

HTML 中有三两个链接，分别连去 Apple、Dlmu 网站。

```
<a href=”https://www.apple.com” title=”Apple”>Apple</a>
<a href=”https://www.dlmu.edu.cn” title=”Dlmu”>Dlmu</a>
```

透过 a[title]即可选取带有 title 属性的<a>Tag:

```
a[title]{
    color:red
};
```

除此之外，我们还可以根据属性里面的值去做配对

栗子：

```
a[href=”https://www.apple.com”]
{
    color:red;
}
```

即可选取到 Apple 的超链接，它还支援类似搜寻的功能；

- 运用^=符号可以搜寻到以什么开头的值 a[href^=”https”]
- 运用$=符号可以搜寻到以什么结尾的值 a[href$=”edu.cn”]
- *=符号就当作关键字去搜寻 a[href*=”apple”]

10. Pseudo Classes（伪类）

用于选取普通选取器不能表达的讯息。比如当鼠标移到一个 HTML 元素上的时候更改一个浏览过的超链接的颜色，又比如在十个 中选取第 13579 个或选取第 6 个。这些都需要在普通的选取器上加以修饰才能表达得到的情况。

写法是在选取器后面加一个冒号，要设定超链接在浏览过后的颜色时可以用 a:visited 去定义。

```
a:visited{
    color:red;
}
```

设定当鼠标移到 HTML 元素上的时候的样式可以用：hover 去定义；

```
a:hover{
    color:yellow;
}
```

而在一组多个 HTML 元素中选取特定的某几个 可以使用 nth-child()

栗子：

```
container div:nth-child(2){
    color: red;
}
```

即是选取第二个

若括号里是 even，即是选取双数；

若括号里是 3n+0.即是选取 3 的倍数，以此类推。

附件一：

CSS 选择器

选择器	例子	例子描述
.class	.intro	选择 class="intro" 的所有元素。
.class1.class2	.name1.name2	选择 class 属性中同时有 name1 和 name2 的所有元素。
.class1 .class2	.name1 .name2	选择作为类名 name1 元素后代的所有类名 name2 元素。
#id	#firstname	选择 id="firstname" 的元素。
*	*	选择所有元素。
element	p	选择所有 <p> 元素。
element.class	p.intro	选择 class="intro" 的所有 <p> 元素。
element,element	div, p	选择所有 元素和所有 <p> 元素。
element element	div p	选择 元素内的所有 <p> 元素。
element>element	div > p	选择父元素是 的所有 <p> 元素。
element+element	div + p	选择紧跟 元素的首个 <p> 元素。
element1~element2	p ~ ul	选择前面有 <p> 元素的每个 元素。
[*attribute*]	[target]	选择带有 target 属性的所有元素。
[*attribute=value*]	[target=_blank]	选择带有 target="_blank" 属性的所有元素。
[*attribute~=value*]	[title~=flower]	选择 title 属性包含单词 "flower" 的所有元素。
[*attribute|=value*]	[lang|=en]	选择 lang 属性值以 "en" 开头的所有元素。
[*attribute^=value*]	a[href^="https"]	选择其 src 属性值以 "https" 开头的每个 <a> 元素。
[*attribute$=value*]	a[href$=".pdf"]	选择其 src 属性以 ".pdf" 结尾的所有 <a> 元素。
[*attribute*=value*]	a[href*="w3schools"]	选择其 href 属性值中包含 "abc" 子串的每个 <a> 元素。
:active	a:active	选择活动链接。
::after	p::after	在每个 <p> 的内容之后插入内容。
::before	p::before	在每个 <p> 的内容之前插入内容。
:checked	input:checked	选择每个被选中的 <input> 元素。
:default	input:default	选择默认的 <input> 元素。
:disabled	input:disabled	选择每个被禁用的 <input> 元素。

:empty	p:empty	选择没有子元素的每个 <p> 元素（包括文本节点）。
:enabled	input:enabled	选择每个启用的 <input> 元素。
:first-child	p:first-child	选择属于父元素的第一个子元素的每个 <p> 元素。
::first-letter	p::first-letter	选择每个 <p> 元素的首字母。
::first-line	p::first-line	选择每个 <p> 元素的首行。
:first-of-type	p:first-of-type	选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
:focus	input:focus	选择获得焦点的 input 元素。
:fullscreen	:fullscreen	选择处于全屏模式的元素。
:hover	a:hover	选择鼠标指针位于其上的链接。
:in-range	input:in-range	选择其值在指定范围内的 input 元素。
:indeterminate	input:indeterminate	选择处于不确定状态的 input 元素。
:invalid	input:invalid	选择具有无效值的所有 input 元素。
:lang(*language*)	p:lang(it)	选择 lang 属性等于 "it"（意大利）的每个 <p> 元素。
:last-child	p:last-child	选择属于其父元素最后一个子元素每个 <p> 元素。
:last-of-type	p:last-of-type	选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
:link	a:link	选择所有未访问过的链接。
:not(*selector*)	:not(p)	选择非 <p> 元素的每个元素。
:nth-child(*n*)	p:nth-child(2)	选择属于其父元素的第二个子元素的每个 <p> 元素。
:nth-last-child(*n*)	p:nth-last-child(2)	同上，从最后一个子元素开始计数。
:nth-of-type(*n*)	p:nth-of-type(2)	选择属于其父元素第二个 <p> 元素的每个 <p> 元素。
:nth-last-of-type(*n*)	p:nth-last-of-type(2)	同上，但是从最后一个子元素开始计数。
:only-of-type	p:only-of-type	选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
:only-child	p:only-child	选择属于其父元素的唯一子元素的每个 <p> 元素。
:optional	input:optional	选择不带 "required" 属性的 input 元素。
:out-of-range	input:out-of-range	选择值超出指定范围的 input 元素。
::placeholder	input::placeholder	选择已规定 "placeholder" 属性的 input 元素。
:read-only	input:read-only	选择已规定 "readonly" 属性的 input 元素。

:read-write	input:read-write	选择未规定 "readonly" 属性的 input 元素。
:required	input:required	选择已规定 "required" 属性的 input 元素。
:root	:root	选择文档的根元素。
::selection	::selection	选择用户已选取的元素部分。
:target	#news:target	选择当前活动的 #news 元素。
:valid	input:valid	选择带有有效值的所有 input 元素。
:visited	a:visited	

附件二：

CSS Selector

#	id=""	
.class	class=""	
.class1.class2	class="class1 class2"	选择 class1 和 class2 同时具备的元素（无空格）
.class1 .class2	class="class1" class="class2"	class1 下的 class2
tag{}	<tag></tag>	
空格	后代选择器：标签下的子标签 例如 .a b 指 a 下的 b	html 写法 <tag1 class="container" id="a"><tag id="b"><tag id="c"></tag></tag><tag id="d"></tag></tag1> css 写法 .container tag{}
p.marked{}	为所有 class="marked" 的 p 指定样式	
,	和	div , span 选中 和
>	A > B 选中所有父级是 A 的 B	html 写法 <tag class="container"id="a"> <tag id="b"> <tag id="c"></tag> </tag> <tag id="d"></tag> </tag> css 写法 .container > tag
+	A + B 紧跟着 A 的 B	html 写法 <tag class="class-name" id="a"> <tag id="b"></tag> <tag id="c"></tag> </tag> css 写法 tag + .b
~	某一层同一层之后的所有同种元素	html 写法 <tag1 class="container" id="a"> <tag id="b"></tag> </tag> <tag id="c"></tag> <tag id="d"></tag>

		css 写法 .container ~ tag{} 作用：修饰带有 class 属性值为 container 的容器之后的同一层的 tag 标签元素
*	所有元素 （一般会配合其他符号组合使用）	html 写法 <tag1 class="container" id="a"><tag id="b"></tag></tag><tag id="c"></tag><tag id="d"></tag> css 写法 .container *{} 作用：修饰 class 值为 container 的所有子元素，不包括元素本身。

属性选择器

元素[属性]	选择具有指定属性的指定元素，如 a[title]{} 选取具有 title 属性的<a>
正则表达式	
^="a"	选择以 a 开头的内容
$="b"	选择以 b 结尾的内容
*="c"	选择含有 c 的内容

伪类(Pesudo Class)

:visited	访问过的超链接
:hover	鼠标悬停
:nth-child(<表达式>){}	括号内表达式: 选择第几个/even 双数/odd 单数..?/3n+0