CSS选择器

@M了个J

https://github.com/CoderMJLee

https://space.bilibili.com/325538782

小码哥教育 SEEMYGO

实力IT教育 www.520it.com

CSS选择器（selector）

- 开发中经常需要找到特定的网页元素进行设置样式
- 思考：如何找到特定的那个元素？

- 什么是CSS选择器
  - 按照一定的规则选出符合条件的元素，为之添加CSS样式
- 选择器的种类繁多，大概可以这么归类
  - 通用选择器（universal selector）
  - 元素选择器（type selectors）
  - 类选择器（class selectors）
  - id选择器（id selectors）
  - 属性选择器（attribute selectors）
  - 组合（combinators）
  - 伪类（pseudo-classes）
  - 伪元素（pseudo-elements）

元素选择器（type selectors）

- 所有的div元素

```
div {
    color: red;
}
```

```
<div>文字内容1</div>
<p>文字内容2</p>
<span>文字内容3</span>
<div>文字内容4</div>
```

文字内容1

文字内容2

文字内容3

文字内容4

- 或者叫做“标签选择器”

通用选择器（universal selector）

- 所有的元素

```
* {
    color: red;
}
```

```
<div>文字内容1</div>
<p>文字内容2</p>
<span>文字内容3</span>
```

文字内容1

文字内容2

文字内容3

- 一般用来给所有元素作一些通用性的设置
  - 比如内边距、外边距
  - 参考：http://www.jd.com

- 效率比较低，尽量不要使用

id选择器（id selectors）

- id值为one的元素

```
#one {
    color: red;
}
```

```
<div id="two">文字内容1</div>
<p id="one">文字内容2</p>
<span id="three">文字内容3</span>
<div id="four">文字内容4</div>
```

文字内容1

文字内容2

文字内容3
文字内容4

id注意点

- 一个HTML文档里面的id值是唯一的，不能重复
- id值如果由多个单词组成，单词之间可以用中划线-、下划线_连接，也可以使用驼峰标识
- 最好不要用标签名作为id值

```
<!-- 中划线隔开 -->
<div id="the-first-box"></div>

<!-- 下划线隔开 -->
<div id="the_first_box"></div>

<!-- 驼峰标识：第一个单词首字母小写，其他单词首字母大写-->
<div id="theFirstBox"></div>
```

- 中划线又叫连字符（hyphen）

类选择器（class selectors）

- class值有one的元素

```
.one {
    color: red;
}
```

```
<div class="two">文字内容1</div>
<p class="one">文字内容2</p>
<span class="two one">文字内容3</span>
<div class="one">文字内容4</div>
```

文字内容1

文字内容2

文字内容3
文字内容4

class注意点

- 一个元素可以有多个class值，每个class之间用空格隔开
- class值如果由多个单词组成，单词之间可以用中划线-、下划线_连接，也可以使用驼峰标识
- 最好不要用标签名作为class值

```
<!-- 中划线隔开 -->
<div class="the-first-box"></div>

<!-- 下划线隔开 -->
<div class="the_first_box"></div>

<!-- 驼峰标识：第一个单词首字母小写，其他单词首字母大写-->
<div class="theFirstBox"></div>
```

小码哥教育 SEEMYGO

练习

```css
/* box style */
.box {
    margin: 0 0 30px 0;
    padding: 15px;
    color: red;
    background-color: #007799;
    box-shadow: none;
}
```

练习

确定 删除 重置 取消

- 在开发过程中，类选择器可以认为是最常用最灵活的选择器
- 一般会将一些公共样式抽取出来，写到某个类选择器中。谁想要使用这些样式，加上类名即可

属性选择器（attribute selectors） - [att]

- 拥有title属性的元素

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

```
<div title="one">文字内容1</div>
<div>文字内容2</div>
<div>
  <p title="two">文字内容3</p>
</div>
<span title="">文字内容4</span>
```

文字内容1
文字内容2

文字内容3

文字内容4

属性选择器 - [att=val]

- title属性值恰好等于one的元素

```
[title="one"] {
    color: red;
}
```

```
[title='one'] {
  color: red;
}
```

```
[title=one] {
    color: red;
}
```

```
<div title="one">文字内容1</div>
<div>文字内容2</div>
<div>
    <p title="one">文字内容3</p>
</div>
<span title="two">文字内容4</span>
```

文字内容1
文字内容2

文字内容3

文字内容4

- #test和[id="test"]看起来好像一样，其实本质不一样

属性选择器 - [attr~=val]

- title属性值包含单词one的元素（单词one与其他单词之间必须用空格隔开）

```
[title~="one"] {
    color: red;
}
```

```
<div title="testonetwo">文字内容1</div>
<div title="testone two">文字内容2</div>
<div title="test one two">文字内容3</div>
<div title="test one">文字内容4</div>
<div title="one two">文字内容5</div>
<div title="one-two">文字内容6</div>
<div title="one_two">文字内容7</div>
<div title="one">文字内容8</div>
<span title="two">文字内容9</span>
```

文字内容1
文字内容2
文字内容3
文字内容4
文字内容5
文字内容6
文字内容7
文字内容8
文字内容9

- 以下2个选择器效果一致

```
[class~="one"] {
    color: red;
}
```

```
.one {
    color: red;
}
```

属性选择器 - [attr|=val]

- title属性值恰好等于one 或者 以单词one开头且后面紧跟着连字符-的元素

```
[title|="one"] {
    color: red;
}
```

```
<div title="test one two">文字内容1</div>
<div title="test-one">文字内容2</div>
<div title="one two">文字内容3</div>
<div title="one-two">文字内容4</div>
<div title="one_two">文字内容5</div>
<div title="one">文字内容6</div>
<span title="two">文字内容7</span>
```

文字内容1
文字内容2
文字内容3
文字内容4
文字内容5
文字内容6
文字内容7

- 一般是用在lang属性上面

```
[lang|="en"] {
  color: red;
}
```

```
<div>
  <span lang="en-US">666</span>
  <span lang="zh">777</span>
  <span lang="zh-CN">888</span>
  <span lang="en">999</span>
  <span lang="en-BZ">000</span>
</div>
```

属性选择器 - [attr^=val]

- title属性值以单词one开头的元素

```
[title^="one"] {
    color: red;
}
```

```
<div title="one">文字内容1</div>
<p title="one two">文字内容2</p>
<span title="onetwo">文字内容3</span>
<div title="one-two">文字内容4</div>
<p title="one_two">文字内容5</p>
```

文字内容1

文字内容2

文字内容3
文字内容4

文字内容5

属性选择器 - [attr$=val]

- title属性值以单词one结尾的元素

```
[title$="one"] {
    color: red;
}
```

```
<div title="one">文字内容1</div>
<p title="twoone">文字内容2</p>
<span title="two one">文字内容3</span>
<div title="two-one">文字内容4</div>
<p title="two_one">文字内容5</p>
```

文字内容1

文字内容2

文字内容3

文字内容4

文字内容5

属性选择器 - [attr*=val]

- title属性值包含单词one的元素

```
[title*="one"] {
    color: red;
}
```

```
<div title="one">文字内容1</div>
<p title="twoone">文字内容2</p>
<span title="two one">文字内容3</span>
<div title="two-one">文字内容4</div>
<p title="two_one">文字内容5</p>
<p title="two哈哈one">文字内容6</p>
```

文字内容1

文字内容2

文字内容3

文字内容4

文字内容5

文字内容6

后代选择器（descendant combinator）

- div元素里面的span元素（包括直接、间接子元素）

```css
div span {
    color: red;
}
```

```html
<span>文字内容1</span>
<div>
    <span>文字内容2</span>
    <p>
        <span>文字内容3</span>
    </p>
</div>
<div>
    <span>文字内容4</span>
</div>
<span>文字内容5</span>
```

文字内容1
文字内容2

文字内容3

文字内容4
文字内容5

后代选择器

```css
div p span {
    color: red;
}
```

```html
<span>文字内容1</span>
<div>
    <span>文字内容2</span>
</div>
<div>
    <p>
        <span>文字内容3</span>
        <strong>
            <span>文字内容4</span>
        </strong>
    </p>
</div>
<p>
    <span>文字内容5</span>
</p>
```

文字内容1
文字内容2

文字内容3 **文字内容4**

文字内容5

子选择器（child combinators）

- div元素里面的直接span子元素（不包括间接子元素）

```
div>span {
    color: red;
}
```

```
div > span {
    color: red;
}
```

```
<span>文字内容1</span>
<div>
    <span>文字内容2</span>
    <p>
        <span>文字内容3</span>
    </p>
</div>
<div>
    <span>文字内容4</span>
</div>
<span>文字内容5</span>
```

文字内容1
文字内容2

文字内容3

文字内容4
文字内容5

小码哥教育 SEEMYGO

子选择器

```
div>p>span {
    color: red;
}
```

```
<div>
    <span>span1</span>
    <p>
        <span>span2</span>
        <a href="#"><span>span3</span></a>
    </p>
    <ul>
        <li>
            <p>
                <span>span4</span>
            </p>
        </li>
    </ul>
</div>
```

span1

span2 span3

- span4

相邻兄弟选择器（adjacent sibling combinator）

- div元素后面紧挨着的p元素（且div、p元素必须是兄弟关系）

```
div+p {
  color: red;
}
```

```
<p>文字内容1</p>
<div>
    <p>文字内容2</p>
</div>
<p>文字内容3</p>
<p>文字内容4</p>
```

文字内容1

文字内容2

文字内容3

文字内容4

全体兄弟选择器（general sibling combinator）

- div元素后面的p元素（且div、p元素必须是兄弟关系）

```css
div~p {
  color: red;
}
```

```html
<span>文字内容1</span>
<p>文字内容2</p>
<div>文字内容3</div>
<div>
    <p>文字内容4</p>
</div>
<p>文字内容5</p>
<p>文字内容6</p>
```

文字内容1

文字内容2

文字内容3

文字内容4

文字内容5

文字内容6

选择器组 - 交集选择器

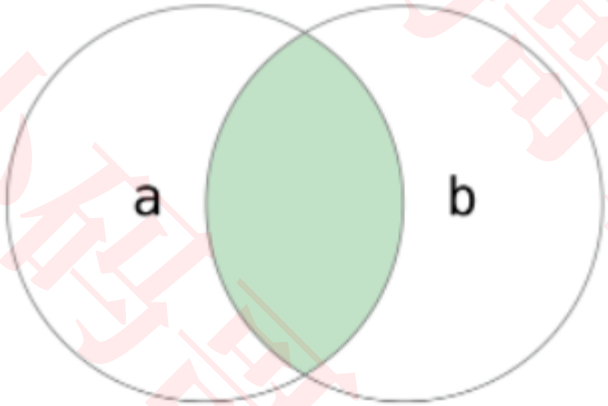
- 同时符合2个条件的元素：div元素、class值有one

```
div.one {
    color: red;
}
```

```
<div class="one">文字内容1</div>
<div class="two">文字内容2</div>
<p class="one">文字内容3</p>
```

文字内容1
文字内容2

文字内容3



- 所有同时符合3个条件的元素：div元素、class值有one、title属性值等于test

```
div.one[title="test"] {
    color: red;
}
```

```
<div class="one">文字内容1</div>
<div class="one" title="test">文字内容2</div>
<div class="one" title="other">文字内容3</div>
```

文字内容1
文字内容2
文字内容3

选择器组 - 并集选择器

- 所有的div元素 + 所有class值有one的元素 + 所有title属性值等于test的元素

```
div, .one, [title="test"] {
    color: red;
}
```

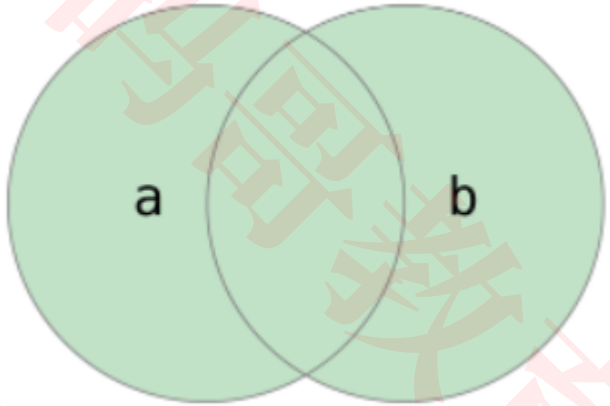
```
<div>文字内容1</div>

<span title="test">文字内容2</span>

<p class="one">文字内容3</p>
```

文字内容1
文字内容2

文字内容3




```
div {
  color: red;
}

.one {
  color: red;
}

[title="test"] {
  color: red;
}
```

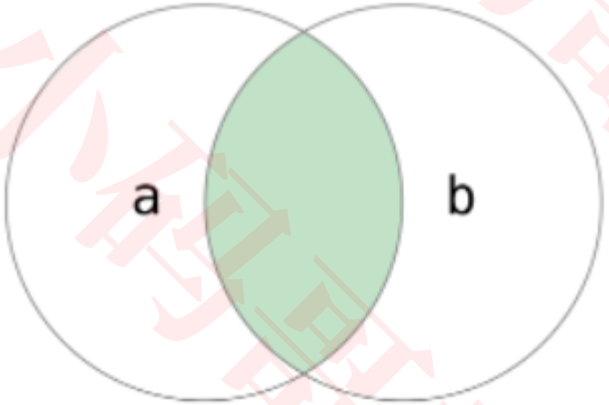
```
h1 { color: red; }
h2 { color: red; }
h3 { color: red; }
h4 { color: red; }
h5 { color: red; }
h6 { color: red; }

h1, h2, h3, h4, h5, h6 {
  color: red;
}
```

选择器组 - 交集、并集选择器对比

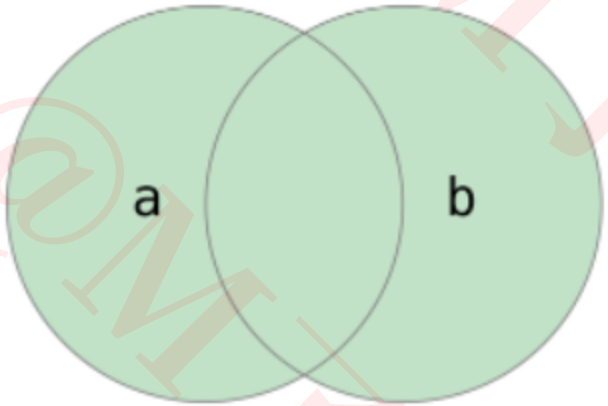
- 交集选择器

```
div.one {
    color: red;
}
```



- 并集选择器

```
div, .one {
  color: red;
}
```



练习

- 让所有文本输入框的文字颜色都为红色

```
input[type="text"], textarea {
  color: red;
}
```

- 思考以下代码的含义

- `#container .box .badge { }`
- `div div span { }`和`div span { }`的区别

练习

- 请编写选择器找到内容为em2\em3\em4的em元素，设置文字颜色为red

```
<div><em>em0</em></div>
<div class="box">
    <div><p>paper</p></div>
    <p>
        <span>span</span>
        <strong>
            <span title="one">
                <strong id="dog">strong-dog</strong>
                <span>
                    <em>em1</em>
                    <strong id="cat">strong-cat</strong>
                    <em>em2</em>
                    <em>em3</em>
                    <em>em4</em>
                </span>
            </span>
        </strong>
    </p>
</div>
```

练习

- 选择器编写建议：在保证精准性的前提下，尽量简洁

```
div.box>p>strong>span[title="one"]>span>strong#cat~em {
    color: red;
}

div.box span[title="one"] strong#cat~em {
    color: red;
}

.box [title="one"] #cat~em {
    color: red;
}

#cat~em {
    color: red;
}
```

伪类（pseudo-classes）

- 常见的伪类有
- 动态伪类（dynamic pseudo-classes）
  - ✓ :link、:visited、:hover、:active、:focus
- 目标伪类（target pseudo-classes）
  - ✓ :target
- 语言伪类（language pseudo-classes）
  - ✓ :lang( )
- 元素状态伪类（UI element states pseudo-classes）
  - ✓ :enabled、:disabled、:checked
- 结构伪类（structural pseudo-classes）
  - ✓ :nth-child( )、:nth-last-child( )、:nth-of-type( )、:nth-last-of-type( )
  - ✓ :first-child、:last-child、:first-of-type、:last-of-type
  - ✓ :root、:only-child、:only-of-type、:empty
- 否定伪类（negation pseudo-classes）
  - ✓ :not()

动态伪类（dynamic pseudo-classes）

- 使用举例
  - a:link 未访问的链接
  - a:visited 已访问的链接
  - a:hover 鼠标挪动到链接上
  - a:active 激活的链接（鼠标在链接上长按住未松开）

- 使用注意
  - :hover必须放在:link和:visited后面才能完全生效
  - :active必须放在:hover后面才能完全生效
  - 所以建议的编写顺序是 :link、:visited、:hover、:active
  - 记忆：女朋友看到LV包包后，ha ha大笑

- 除了a元素，:hover、:active也能用在其他元素上

动态伪类 - :focus

- :focus指当前拥有输入焦点的元素（能接收键盘输入）
- 文本输入框一聚焦后，背景就会变红色

```
input:focus {
    background: red;
}
```

- 因为链接a元素可以被键盘的Tab键选中聚焦，所以:focus也适用于a元素

```
input:focus {
  color: red;
}

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

```
<input type="text">

<input type="text">

<a href="#">链接1</a>

<a href="#">链接2</a>
```

- 动态伪类编写顺序建议为
- :link、:visited、:focus、:hover、:active
- 记忆：女朋友看到LV包包后，（Feng）疯一样地ha ha大笑

去除a元素默认的:focus效果

链接1

```
a:focus {
    outline: none;
}
```

- 或者将tabindex属性设置为-1

tabindex属性

- 使用tabindex可以控制tab键选中元素的顺序，从1开始
- tabindex设置为-1，代表禁止使用tab键选中

细节

- 直接给a元素设置样式，相当于给a元素的所有动态伪类都设置了

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

- 相当于a:link、a:visited、a:hover、a:active、a:focus的color都是red

目标伪类（target pseudo-classes）

```css
p:target {
    color: red;
}
```

```html
<p id="ppp">ppp</p>

<a href="#ppp">aaa</a>
```

- 当元素被锚点链接当作目标跳转之后起作用

语言伪类（language pseudo-classes）

- 语言是en系列（英语）的所有div元素

```
<body lang="en">

    <div>文字内容1</div>
    <div lang="en">文字内容2</div>
    <div lang="zh">文字内容3</div>
    <div lang="en-us">文字内容4</div>

</body>
```

```
div:lang(en) {
    color: red;
}
```

文字内容1
文字内容2
文字内容3
文字内容4

```
div[lang|="en"] {
    color: red;
}
```

文字内容1
文字内容2
文字内容3
文字内容4

元素状态伪类（UI element states pseudo-classes）

- :enabled
  - 启用状态
- :disabled
  - 禁用状态
- :checked
  - 被选中状态

```html
<input type="text">
<input type="button" value="按钮1">
<br><br>

<input type="text" disabled>
<input type="button" value="按钮2" disabled>
<br><br>

<input type="checkbox" checked>
<input type="checkbox">
<input type="radio">
<input type="radio" checked>
```

```css
input:enabled {
    border: 2px solid red;
}

input:disabled {
    border: 2px solid black;
}

input:checked {
    outline: 2px solid blue;
}
```

结构伪类（structural pseudo-classes） - :nth-child( )

- :nth-child(1)
  - 是父元素中的第1个子元素

```
p:nth-child(1) {
    color: red;
}
```

```
<body>
    <p>文字内容1</p>
    <p><span>文字内容2</span></p>
    <div>
        <p>文字内容3</p>
        <p>文字内容4</p>
    </div>
</body>
```

文字内容1

文字内容2

文字内容3

文字内容4

结构伪类 - :nth-child( )

- :nth-child(2n)
  - n代表任意正整数和0
  - 是父元素中的第偶数个子元素（第2、4、6、8.....个）
  - 跟:nth-child(even)同义

```
p:nth-child(2n) {
  background: red;
}
```

```
<div>
  <p>第1行</p>
  <p>第2行</p>
  <p>第3行</p>
  <p>第4行</p>
  <p>第5行</p>
</div>
```

结构伪类 - :nth-child( )

- :nth-child(2n + 1)
  - n代表任意正整数和0
  - 是父元素中的第奇数个子元素（第1、3、5、7.....个）
  - 跟:nth-child(odd)同义

```
p:nth-child(2n + 1) {
  background: red;
}
```

```
<div>
  <p>第1行</p>
  <p>第2行</p>
  <p>第3行</p>
  <p>第4行</p>
  <p>第5行</p>
</div>
```

结构伪类 - :nth-child( )

```css
p:nth-child(4n + 1) { background: red; }
p:nth-child(4n + 2) { background: blue; }
p:nth-child(4n + 3) { background: green; }
p:nth-child(4n + 4) { background: orange; }
```

```html
<body>
    <div>div</div>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
    <span>span</span>
    <p>文字内容5</p>
    <p>文字内容6</p>
</body>
```

结构伪类 - :nth-child( )

- :nth-child()的完整使用格式是:nth-child(an + b)
  - 是父元素中的第an+b个子元素
  - n代表任意正整数和0
  - a、b需要给出具体值，可以是正整数、负整数、0

- 举例
  - 如果a为0，b为1，an + b就是1，也就是:nth-child(1)
  - 如果a为2，b为0，an + b就是2n，也就是:nth-child(2n)

- 思考：如何用:nth-child()表示最前面2个子元素?
  - :nth-child(-n + 2)

- 注意：不能写成b + an

结构伪类 - :nth-last-child( )

- :nth-last-child()的语法跟:nth-child()类似，不同点是:nth-last-child()从最后一个子元素开始往前计数
- :nth-last-child(1)，代表倒数第一个子元素
- :nth-last-child(-n + 2)，代表最后2个子元素

```
p:nth-last-child(-n + 3) {
    background: red;
}
```

```
<body>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
    <p>文字内容5</p>
</body>
```


```
p:nth-last-child(odd) {
    background: red;
}
```

```
<body>
    <p>文字内容1</p>
    <p>文字内容2</p>
    <p>文字内容3</p>
    <p>文字内容4</p>
</body>
```

结构伪类 - :nth-child( )、:nth-last-child( )

- 思考：如何表示第2个~倒数第2个元素（去除头和尾元素）

```
<body>
        <p>文字内容1</p>
        <p>文字内容2</p>
        <p>文字内容3</p>
        <p>文字内容4</p>
        <p>文字内容5</p>
        <p>文字内容6</p>
</body>
```

```
p:nth-child(n + 2):nth-last-child(n + 2) {
    background: red;
}
```

文字内容1

文字内容2

文字内容3

文字内容4

文字内容5

文字内容6

结构伪类 - :nth-of-type( )、:nth-last-of-type( )

- :nth-of-type()用法跟:nth-child()类似，不同点是:nth-of-type()计数时只计算同种类型的元素

```
p:nth-of-type(2) {
    background: red;
}
```

```
<body>
        <p>文字内容1</p>
        <div><p>文字内容2</p></div>
        <div>
            <div><p>文字内容3</p></div>
            <p>文字内容4</p>
            <p>文字内容5</p>
        </div>
        <p>文字内容6</p>
</body>
```

文字内容1

文字内容2

文字内容3

文字内容4

文字内容5

文字内容6

- :nth-last-of-type()用法跟:nth-of-type()类似
  - 不同点是:nth-last-of-type()从最后一个这种类型的子元素开始往前计数

小码哥教育 SEEMYGO

结构伪类

- **:first-child**，等同于**:nth-child(1)**
- **:last-child**，等同于**:nth-last-child(1)**
- **:first-of-type**，等同于**:nth-of-type(1)**
- **:last-of-type**，等同于**:nth-last-of-type(1)**
- **:only-child**，是父元素中唯一的子元素
  - 等同于**:first-child:last-child**或者**:nth-child(1):nth-last-child(1)**
- **:only-of-type**，是父元素中唯一的这种类型的子元素
  - 等同于**:first-of-type:last-of-type**或者**:nth-of-type(1):nth-last-of-type(1)**
- **:root**，根元素，就是HTML元素

结构伪类 - :empty

- :empty代表里面完全空白的元素

```
p:empty {
    width: 100px;
    height: 20px;
    background: red;
}
```

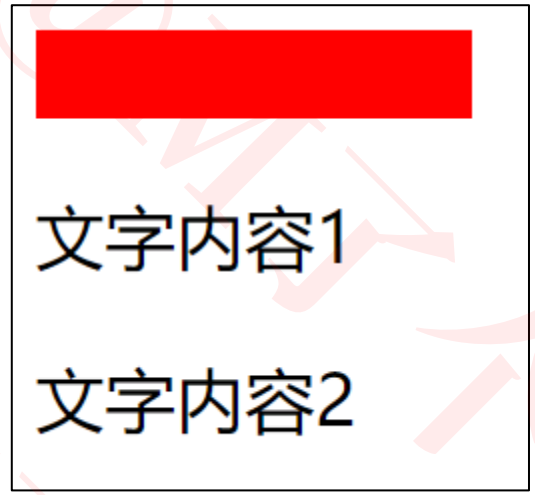
```
<p></p>

<p>文字内容1</p>

<p>    </p>

<p><span></span></p>

<p>文字内容2</p>
```



否定伪类（negation pseudo-class）

- :not()的格式是:not(x)
- x是一个简单选择器
- ✓ 元素选择器、通用选择器、属性选择器、类选择器、id选择器、伪类（除否定伪类）

- :not(x)表示除x以外的元素

```
:not(p):not(body):not(html) {
    color: red;
}
```

```
<p>文字内容1</p>
<span>文字内容2</span>
<div>文字内容3</div>
<p>文字内容4</p>
```

文字内容1

文字内容2
文字内容3

文字内容4

```
p:not(:first-of-type):not(:last-of-type) {
    color: red;
}
```

```
<p>文字内容1</p>
<p>文字内容2</p>
<p>文字内容3</p>
<p>文字内容4</p>
```

文字内容1

文字内容2

文字内容3

文字内容4

否定伪类

- :not()支持简单选择器，不支持组合。比如下面的写法是不支持的

```
:not(div.one) {
  color: red;
}
```

```
:not(div .one) {
  color: red;
}
```

伪元素（pseudo-elements）

- 常用的伪元素有
  - `:first-line`、`::first-line`
  - `:first-letter`、`::first-letter`
  - `:before`、`::before`
  - `:after`、`::after`

- 为了区分伪元素和伪类，建议伪元素使用2个冒号，比如`::first-line`

伪元素 - ::first-line

- ::first-line可以针对首行文本设置属性

```
div::first-line {
  color: blue;
  text-decoration: underline;
}
```

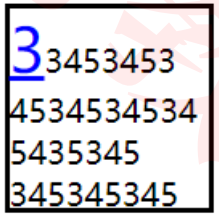
33453453
4534534534
5435345
345345345

- 只有下列属性可以应用在::first-line伪元素
  - 字体属性、颜色属性、背景属性
  - word-spacing、letter-spacing、text-decoration、text-transform、line-height

伪元素 - ::first-letter

- ::first-letter可以针对首字母设置属性

```
div::first-letter {
  color: blue;
  text-decoration: underline;
  font-size: 30px;
}
```



- 只有下列属性可以应用在::first-letter伪元素
  - 字体属性、margin属性、padding属性、border属性、颜色属性、背景属性
  - text-decoration、text-transform、letter-spacing、word-spacing（适当的时候）、line-height、float、vertical-align（只有当float是none时）

伪元素 - ::before和::after

- ::before和::after用来在一个元素的内容之前或之后插入其他内容（可以是文字、图片）

```
div {
  color: red;
}

div::before {
  content: "before";
  font-size: 40px;
}

div::after {
  content: url("bg001.png");
}

<div>原来的一段文字</div>
```

- 在CSS属性值中，使用url(图片的URL)来引用图片
  - url(dot.png);
  - url('dot.png');
  - url("dot.png");

attr()

- 在content中，还可以使用attr(属性名)来获得元素的属性值

```
a[href]::after {
  content: "[" attr(href) "]";
}
```

```
<a href="http://www.jd.com">京东</a><br>
<a href="http://www.baidu.com">百度</a><br>
<a href="http://www.taobao.com">淘宝</a><br>
<a href="http://www.520it.com">小码哥</a>
```

京东[http://www.jd.com]
百度[http://www.baidu.com]
淘宝[http://www.taobao.com]
小码哥[http://www.520it.com]