SVG 是一种基于 XML 语法的图像格式，全称是可缩放矢量图（Scalable Vector Graphics）。其他图像格式都是基于像素处理的，SVG 则是属于对图像的形状描述，所以它本质上是文本文件，体积较小，且不管放大多少倍都不会失真。

Svg使用

SVG 文件可以直接插入网页，成为 DOM 的一部分

SVG 代码也可以写在一个独立文件中，然后用<img>、<object>、<embed>、<iframe>等标签插入网页。

<img src="circle.svg">

<object id="object" data="circle.svg" type="image/svg+xml"></object>

<embed id="embed" src="icon.svg" type="image/svg+xml">

<iframe id="iframe" src="icon.svg"></iframe>

CSS 也可以使用 SVG 文件。

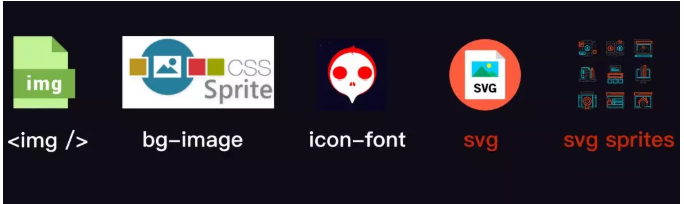
.logo {

background: url(icon.svg);

}

* SVG Sprite
* 如何实现跨浏览器的SVG Sprites

Web自1989年到2019年也算是走过了30年的历程了，那么Web上面有关于图标的使用也经历了相应的发展：

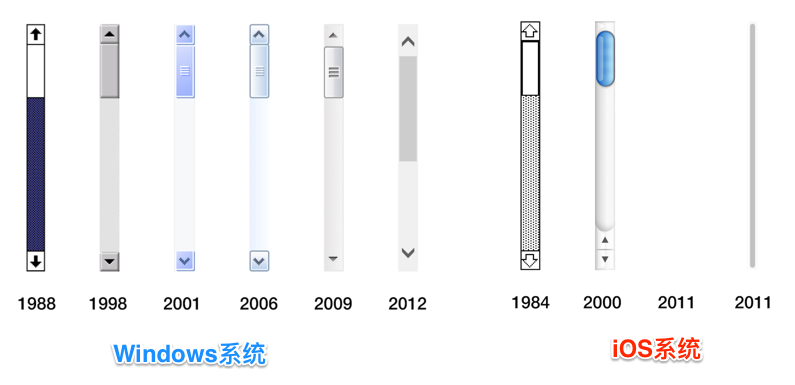


**墨守成规的滚动条**

一直以来，如果仅使用CSS来控制滚动条，我们只能借用overflow属性，比如：

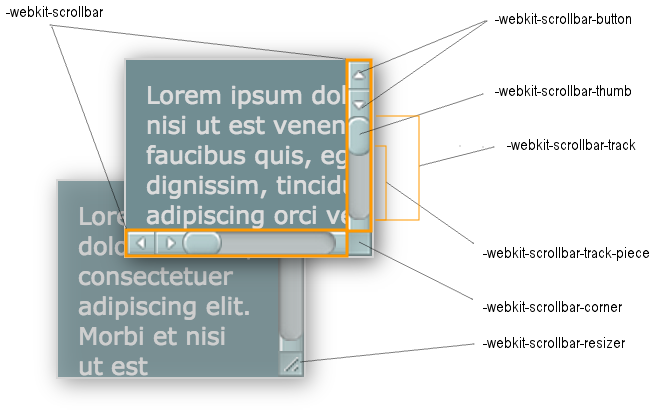
* overflow: auto | scroll;
* // 或者
* overflow-x: auto | scroll;
* // 或者
* overflow-y: auto | scroll;

除了滚动体验之外，视觉体验相对而言更为糟糕，不同系统的浏览器渲染的滚动条风格也不是一致：

[[](https://static.qdskill.com/2018/08/1534753646-scrolling-3.png?_upt=93205f001534754249)](https://static.qdskill.com/2018/08/1534753646-scrolling-3.png?_upt=93205f001534754249)

webkit-scrollbar提供了七个伪元素，通过这些伪元素，我们可以来定制滚动条外观效果。这七个伪元素分别是：

* **::-webkit-scrollbar**：整个滚动条
* **::-webkit-scrollbar-button**：滚动条上的按钮（下下箭头）
* **::-webkit-scrollbar-thumb**：滚动条上的滚动滑块
* **::-webkit-scrollbar-track**：滚动条轨道
* **::-webkit-scrollbar-track-piece**：滚动条没有滑块的轨道部分
* **::-webkit-scrollbar-corner**：当同时有垂直和水平滚动条时交汇的部分
* **::-webkit-resizer**：某些元素的交汇部分的部分样式（类似textarea的可拖动按钮）

[[](https://static.qdskill.com/2018/08/1534753758-scrolling-8.png?_upt=249bf4e31534754362)](https://static.qdskill.com/2018/08/1534753758-scrolling-8.png?_upt=249bf4e31534754362)

Scroll-behavior

3.overflow与锚点定位

  什么是锚点定位？通俗点的解释就是让页面定位到某个位置的点。

CSS规定触发锚点定位行为发生的条件有两种：

  （1）URL地址中的锚链与锚点元素对应并有交互行为

  （2）可focus的锚点元素处于focus状态

  URL的触发条件比较容易，最常用的a标签即可帮助我们实现锚点定位功能。如下代码

< a href=" ">我要定位到和1绑定的元素</ a>

<div id="1">我是要被定位的元素</div>

    此时我们点击a标签，便会触发URL的哈希值改变，然后页面会根据实际情况让id为1的div元素定位在浏览器窗体的上边缘（如果需要的话，div元素本身就在页面的第一行就没必要定位了）

  可focus的锚点元素处于focus状态:在PC端，我们使用Tab键可以快速定位可focus的元素，如果下一个focus元素位于屏幕外，那么浏览器就会自动重新定位，将这个屏幕外的元素定位到屏幕之中。

  虽然两者都是锚点定位，但是这两种方式的表现行为还是有差异的，"URL锚点定位"是让元素定位在浏览器窗体的上边缘，而"focus锚点定位"是让元素在浏览器窗体范围显示即可，不一定是在上边缘

  锚点定位行为发生的本质是通过改变容器滚动高度/宽度来实现的

锚点定位功能和有没有滚动条时也能实现，后面我们会借助这个条件用CSS实现一个选项卡功能。干巴巴的说了许多，还没确认过功能，下面我们就来简单验证一下CSS自带的锚点定位功能。

# letter-spacing与字符动效实例页面

CSS 的 letter-spacing 属性明确了文字的间距行为。

默认值：normal 等同于间距 0

当font-size值与letter-spacing负值绝对值相等时，字体重叠

##### HTML：

<div class="title">我是标题文字内容</div>

.title {

width: 8em;

font-size: 18px;

margin: auto;

white-space: nowrap;

animation: textIn 1s both;

}

@keyframes textIn {

0% {

opacity: 0;

letter-spacing: -200px;

}

60% {

letter-spacing: 5px;

}

100% {

opacity: 1;

letter-spacing: 0;

}

}

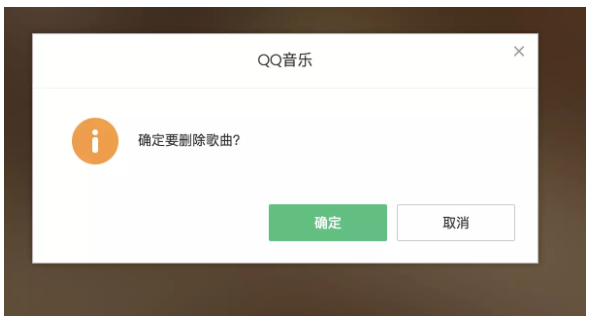
### Direction

 direction: ltr; 默认值

* ltr -> left-to-right(从左到右)
* **东亚、欧美文字书写**都是属于ltr模式的

 direction: rtl;

* rtl -> right-to-left(从右到左)
* 阿拉伯语、希伯来语的书写属于rtl模式
* 再举个栗子：
* 比如制作弹窗组件的时候，**确认**和**取消按钮**有的时候会根据用户的使用行为会**显示在不同的位置**
* 下面来看看这种特性的表现在实际开发中的作用
* windows用户看到的样子：



* mac用户看到的样子：  
  

#### writing-mode作用及真正需要关注的属性值

writing-mode原本设计的是控制内联元素的显示的（即所谓的文本布局-Text Layout）。

**writing-mode语法**

两套不同的语法。一个是IE私有属性，第二个是CSS3规范属性。

先看下未来所需的CSS3语法：

/\* 关键字值 \*/

writing-mode: horizontal-tb; /\* 默认值 \*/

writing-mode: vertical-rl;

writing-mode: vertical-lr;

**用text-indent实现文字下沉效果**

为什么有如此的实现呢？这要归功于中文，在垂直流排版的时候，中文是不会旋转的，还是直立的，也就是说，虽然我们肉眼看上去文字没什么变化，但是，布局流已经发生了变化，以前类似text-indent/letter-spacing等水平控制属性都作用在垂直方向了。