# 第一章： css和文档

1：CSS分替换元素和非替换元素；

2：CSS2.1还使用另外两种基本元素类型：块级元素和行内元素；

# 第二章： 选择器

1：属性选择器

H1[class]{color:silver};/\*所有拥有类的h1元素\*/

input[type=button]{color:red}/\*所有type=button的input元素\*/

2 : 子串匹配属性选择器““

[foo^=“red”]　／\*选择foo属性以red开头的元素\*／

[foo$=“red”]　／\*选择foo属性以red结尾的元素\*／ [foo\*=“red”]　／\*选择foo属性包含red的元素\*／

3 : 特定属性选择类型

Img[ src |=“figure”] /\*以figure开头或者等于figure的属性

4：区分后代元素以及筛选相同匹配的元素

/\*匹配类名为sidebar的div元素，匹配的标签肯定是div元素\*/

div.sidebar{color:red}

/\*中间有空格，匹配div元素中类名为sidebar的后代元素，匹配的标签不一定是div元素\*/

div .sidebar{color:red}

5：兄弟元素，个人理解为只能选择身后的兄弟元素

H1 + p 匹配紧接着h1元素的P元素；

6：伪类和伪元素

伪类选择器：

静态：

a:link{color:red;} //匹配超链接

a:visited{color:red;} //匹配已经访问过的超链接

动态：

:focus //获取焦点

:hover //鼠标停留

:active //激活

出场顺序最好不要变，link,visited,focus,hover,active;

选择第一个子元素:匹配p元素(作为第一个子元素)

p:first-child{font-weigth:bold}

伪元素选择器:

CSS2.1中总设置了4个伪元素：

设置首字母样式；设置第一行样式；设置之前和之后元素的样式

p:first-letter{color:red}; //设置块级元素首字母的样式

p:first-line {color:red}; //设置块级元素第一行文字的样式

p:before {content:“}}”；color:red}; //前面加一对红色中括号

p:after{content: ”}}”;color:red}; //后面加一对红色中括号

# 第三章： 结构和层叠

1：选择器的特殊性(样式冲突时如果选择的问题)

Id属性值 加 100

类、属性、伪类 加 10

标签选择、伪元素 加 1

结合符和通配选择器 加 0

h1{color:red} //1

body h1{color:red} //2

h2.grape {color:red} //11

h2{color:red} //1

html>body table tr[id=”totals”] td ul > li {color:maroon;}

0017

Li#answer{color:navy};

101

2 : 声明重要性最为匹配，次者为内联样式；

p.dark{color:#333 !important} //秒杀其他属性，唯我独尊

0特殊性比没有特殊强，通配符就比继承值强；

避免继承样式出问题，最好就是一开始声明通配符；

3：层叠的排序；

万一特殊性出现相同则采取以下的排序方式：

1：读者的重要声明；

2：创作人员的重要声明；

3：创作人员的正常声明；

4：读者的正常声明；

5：用户代理的声明；

# 第四章： 值和单位

1：WEB安全色: 16进制 00 33 66 99 CC的三元组都是WEB安全色；

2：相对长度单位：em\ex\px;

ex方便的假设等于em的一半；

像素理论:

像素是组成图象的最基本单元要素：点。

分辨率是指在长和宽的两个方向上各拥有的像素个数。

一个像素有多大呢？主要取决于显示器的分辨率，相同面积不同分辨率的显示屏，其像素点大小就不相同。

PPI : 图像分辨率的单位，表示的是每英寸所拥有的像素数目；

CSS2建议将90ppi作为参考像素；

CSS2.1建议将96ppi作为参考像素；

# 第五章： 字体

1：归类为5种通用的字体系列，每个系列又有一些区分；

**Serif字体:**

Serif是有衬线字体，意思是在字的笔画开始、结束的地方有额外的装饰，而且笔画的粗细会有所不同。

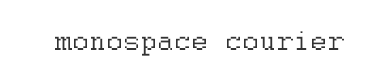
**Sans-serif字体：**

Sans-serif是无衬线字体。与衬线字体相反，这种字体通常是机械的和统一线条的，它们往往拥有相同的曲率，笔直的线条，锐利的转角。



**Monospace字体：**

Monospace字体是不成比例的。它通常用于模拟打字机打出的文本。老式的点阵机的输出，甚至是更老式的视频显示终端。



**Cursive字体**

这些字体试图模仿人的手写体。通常有曲线和sedf字体中没有的笔划装饰组成。



**Fantasy字体**

这些字体无法用任何特征来定义，只有一点是确定的，那就是我们无法很容易地将其归到任何一种其他字体。



2：指定字体系列

p{font-family:times,timesNR,Georgia,serif;} //按照顺序匹配

有特殊符号的字体名要加引号：

h2{font-family:Wedgie,‘Karrank%’}

3 : font-weight字体加粗是如何起作用的

字体加粗度包括100-900 共9个值；

然而这些数字并没有指定的加粗度，100，200，300可能与400都映射到同样的粗细；只要一个关键字对应的变形不会比前一个关键字所对应的更细，CSS规范都是允许的；

另外，还可以选填normal,bold的值，normal对应400，bold对应700；

各种字体的加粗度由字体系列本身指定；

让字体更粗，继承父元素的粗细，会比父元素的计数值大(至于大多少就不好说，至于计数值大了，可是看起来是否更大也不好说)

font-weight:bolder;

让字体更细，font-weight:lighter;

4 : font-size字体大小

可选值：\*px、xx-small、x-small、small、medium、large、x-large、

xx-large；

缩放因子的大小取决于用户的浏览器，往往是1.2；相对于其父元素；

字体的继承可以嵌套，慎用；

5：字体风格 font-style 斜体/直立/倾斜/继承

6：行高，font-size总是在line-height之前的；

# 第六章： 文本属性

1：文本缩进主要用于段落的开头，text-indent;

能继承，百分数慎用；

主要应用于块级元素，行内元素不起作用；也能设置负值；

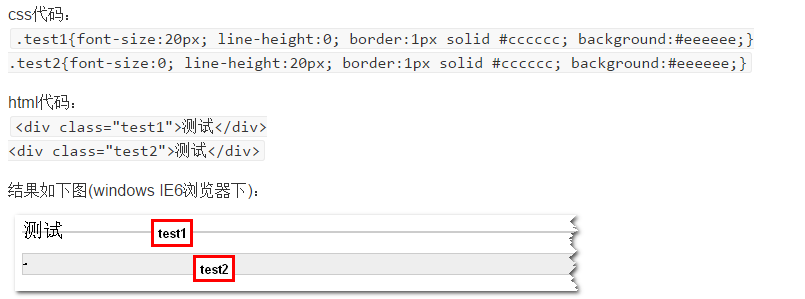
2：文本对齐，text-align;除了left center right 之外，还有justify属性值，根据单元行的宽度拉伸字体间的距离，填满左右边界之间的空间；

3：行高 line-height,主要用于垂直对齐；下面说说行高是如何起作用的；

行高: 具体来说是指两行文字间基线之间的距离；四线三格的第三条线

为什么div里面有文字就会有高度呢？

根本不是文字撑开了div的高度，而是line-height！要证明很简单(如下测试代码)：



为什么line-height行高怎么就产生高度了呢？

在inline box模型中，有个line boxes，这玩意是看不见的，这个玩意的工作就是包裹每行文字。一行文字一个line boxes。例如“艾佛森退役”这5个字，如果它们在一行显示，你艾佛森再牛逼，对不起，只有一个line boxes罩着你；但“春哥纯爷们”这5个字，要是竖着写，一行一个，那真是够爷们，一个字罩着一个line boxes，于是总计五个line boxes。line boxes什么特性也没有，就高度。所以一个没有设置height属性的div的高度就是由一个一个line boxes的高度堆积而成的。

其实line boxes不是直接的生产者，属于中层干部，真正的活儿都是它的手下 – inline boxes干的，这些手下就是文字啦，图片啊，span之类的inline属性的标签啦。line boxes只是个考察汇报人员，考察它的手下谁的实际line-height值最高，谁最高，它就要谁的值，然后向上汇报，形成高度。

行高的垂直居中性：

无论line boxes占据的高度是多少，其占据的空间都是与文字内容共用水平中垂线的；可利用文字与图片水平居中对齐。

图片垂直居中 :



如何设置更好地设置行高的值：

150%虽然和1.5在值上是一样的，但是它们也是有差别的，差别在于继承性，使用百分比会计算line-height的值，然后以px像素为单位继承下去，而1.5则是先继承1.5这个值，遍历到了该标签再计算去line-height的像素值。

最好指定缩放因子；

4：行高vertical-align,基线对齐：

vertical-align：sub: 会使本身元素的基线相对于父元素的基线降低，降低多少取决于用户代理；

vertical-align：super: 会使本身元素的基线相对于父元素的基线上升，上升多少取决于用户代理；

vertical-align： bottom: 会使本身元素的行内框的底端与行框的底端对齐；

vertical-align：top: 会使本身元素的行内框的顶端与行框的顶端对齐；

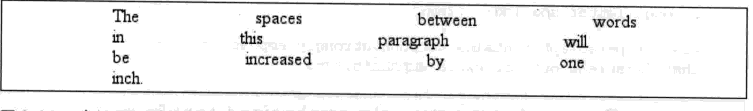
vertical-align： text-bottom: 会使本身元素的行内框的底端与文本框的底端对齐；

vertical-align：text -top: 会使本身元素的行内框的顶端与文本框的顶端对齐；

vertical-align：middle: 父元素的中部；

5：字母隔和字母间隔

word-spacing: \*px; 可能会受text-align的影响；



Letter-spacing: \*px;

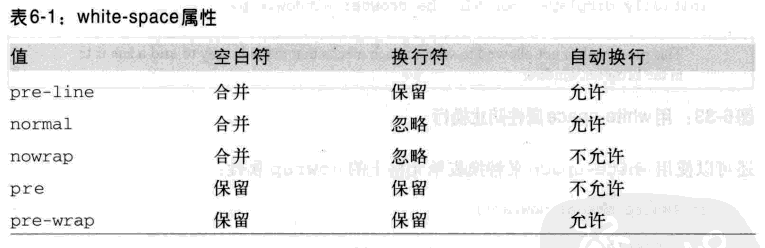
6：文本转换 text-transform

Uppercase lowercase capitalize(首字母) none

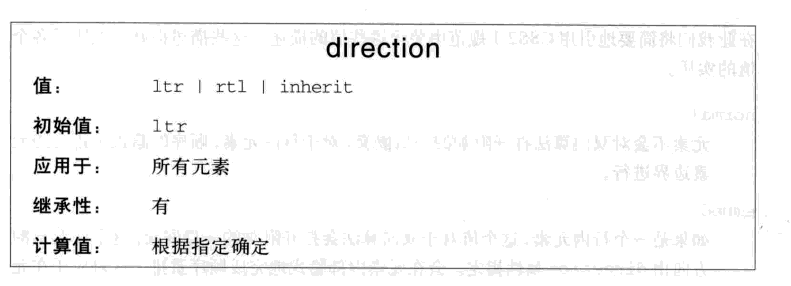
7：文本装饰 text-decoration

Underline overline line-through blink none

8 : white-space属性

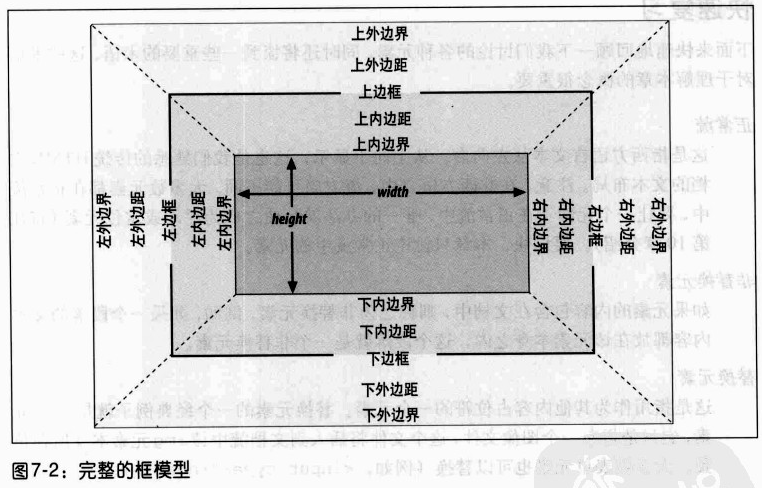


9：direction 文本方向



# 第七章： 基本视觉格式化

1、块级元素的边界于区域



一个元素的width:

IE7,8:

左内边界-右内边界；（内容区）

Chrome,ie11:

Padding-left +(左内边界-右内边界) + padding-right；（内容区）

一个元素的height:

IE7,8:

上内边界-下内边界；（内容区）

Chrome,ie11:

Padding-top +(上内边界-下内边界) + padding-bottom；（内容区）

使用auto：

width,margin-left,margin-right这三个值必须要指定其中的两个值才可以使用auto;优先考虑margin-left,再到width，再到margin-right;

三个元素都设置auto等于没有设置；

严格地说，需求应该是“元素的7项水平属性的总和“不能比其包含块更宽；

替换元素的auto:

替换元素的长宽如果指定auto属性的话，会随着另一个属性的改变而等比例地改变；

垂直属性：

如果一个正常流中的一个快元素的margin-top或margin-bottom设置为auto,她会自动计算为0； 所以不能设置垂直居中；

height=auto:

1:如果父元素设置height=auto，而没有边框或者内边距。那么父元素的高度只是子元素高度的 内容区+内边距+边框的高度；

2:如果父元素设置height=auto，而有边框或者内边距。那么父元素的高度就是子元素高度的 外边距+内容区+内边距+边框的高度；

合并垂直外边距：

只有普通文档流中的块框垂直外边距才会发生合并。

行内框、浮动框或绝对定位之间的外边距不会合并。

1. 一个元素出现在另一个元素上面时

2. 一个元素包含另一个元素时（没有内边距或边框把它们的外边距分开）

3. 一个没有内边距和边框的空元素，它的上、下外边距相遇也会合并

4. 上述3中空元素的外边距遇到另一个元素的外边距，3个外边距发生合并

至于内容区、行高或者行内框之类的概念就要自己查资料了；

# 第八章： 内边距、边框和外边距

# 第九章： 背景和颜色

1：背景图像 background-image

Body {background-image:url(bg23.gif)};

2 : 有方向的重复 background-repeat

repeat repeat-x repeat-y no-repeat inherit

3：背景定位 background-position

接收两个值，可以是数值，也可以是方向值的英文；

P{background-position:0% 0%;}

P{background-position:top bottom;}

4：背景颜色：background-color: red;

5：关联属性：Background-attachment: scroll fixed inherit

让背景图像类似于position：fixed的效果；

不会随文档滚动，而是随着视窗的大小而确定；

汇总：

上述的背景属性可以汇总到一个简写属性：background。这个属性可以从各个其他背景属性取一个值，而且可以采用任何顺序；

例如：

body{background:white url(yinyang.gif) top left repeat scroll};

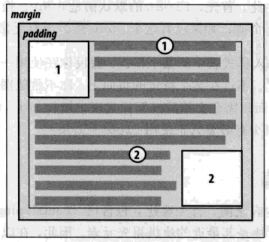
# 第十章： 浮动和定位

1：浮动的详细内幕

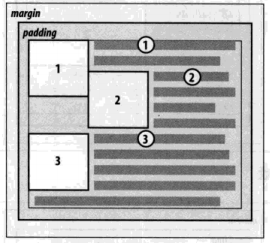
包含块：浮动元素最近的祖先元素；

块级框：不论这个浮动的元素本身是什么，都会生成一个块级框

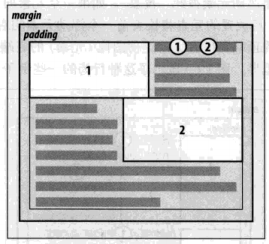
1. 浮动元素的左（右）边界不能超出其包含块的左（右）内边界



1. 浮动元素的左边界必须是源文档中之前出现的左浮动元素的右外边界之外，除非后出现浮动元素的顶端在先出现浮动元素的底端下面；



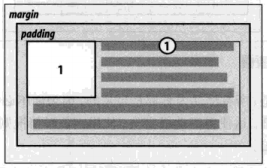
1. 左浮动元素的右边界不会在其右边右浮动元素的左外边界的右边。一个右浮动元素的左外边界不会在其左边任何左浮动元素的右外边界的左边；



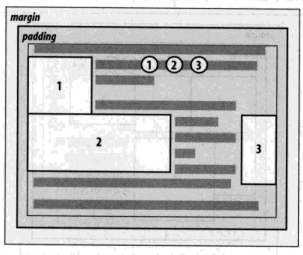
1. 一个浮动元素的顶端不能比其父元素的内顶端更高；

如果一个浮动元素在两个合并外边距之间，放置这个浮动元素时就好像在两个元素之间有一个块级父元素。

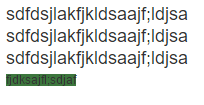
第二点：如果有三个段落，其中中间的段落浮动。在这种情况下，浮动段落就会像有一个块级父元素一样浮动。这就防止浮动段落一直向上移动到三个段落共同的父元素的顶端。



1. 浮动元素的顶端不能比之前所有浮动元素或块级元素的顶端更高；

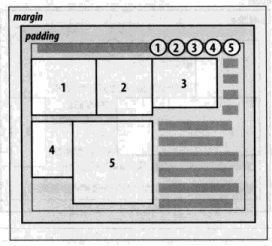


1. 如果源文档中一个浮动元素之前出现另一个元素，浮动元素的顶端不能比包含该元素所生成框的任何行框的顶端更高；

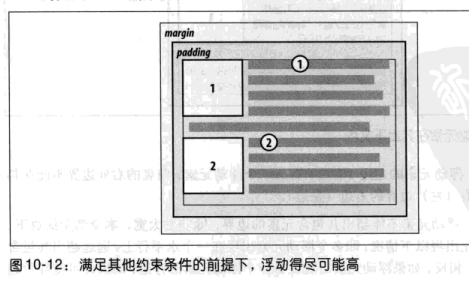


1. 左浮动元素的左边有另一个浮动元素，前者的右外边界不能在其包含块的右边界的右边；

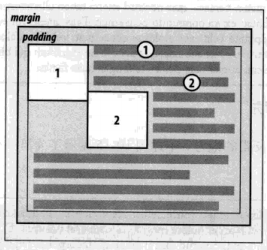
就是说，包含块的宽度必须比浮动元素的所有宽度之和大，否则只能另起一行；



1. 浮动元素必须尽可能高地放置；



1. 左浮动元素必须向左尽可能远，右浮动元素则必须向右尽可能远。位置越高，就会向右或向左浮动得越远；



实用行为：

浮动图像超出了其父元素的底端。因为浮动元素会延伸，所以会可能超出父元素的边界；

通过设置父元素为浮动元素就可以把浮动元素包含在其父元素内。（当然父元素不能把height的值写死）

清除浮动：

<br clear=”both”/>

大多数浏览器的默认行为都是为br元素生成行内框，所以clear不能应用于br，除非改变其display值。

所以最好使用<div style=” clear：”both”；height:0px”></div>

定位：position

static:元素框正常，矩形框；

relative:元素框偏移某个距离。原本所占的空间仍保留。

absolute:元素框从文档流完全删除；

fixed:与absolute相似，不过包含块是视窗本身。

偏移属性：top,right,bottom,left

偏移属性配合着定位属性使用；

限制宽度和高度:推荐使用，因为使用这个属性的话万一子元素扩大，那么父元素就能自适应高度了；

min-width，min-height

max-width, max-height

内容溢出和剪裁:

overflow:内容溢出时的处理方式；

clip:浏览器支持并不好；

绝对定位：

包含块和绝对定位元素：(absolute)

绝对定位元素的包含块是最近的position值不为static的祖先元素。如果找不到则包含块为body元素；

# 第十一章： 表布局

表显示值：

table 此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。

inline-table 此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。

table-row-group 此元素会作为一个或多个行的分组来显示（类似 <tbody>）。

table-header-group 此元素会作为一个或多个行的分组来显示（类似 <thead>）。

table-footer-group 此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。

table-row 此元素会作为一个表格行显示（类似 <tr>）。

table-column-group 此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。

table-column 此元素会作为一个单元格列显示（类似 <col>）

table-cell 此元素会作为一个表格单元格显示（类似 <td> 和 <th>）

table-caption 此元素会作为一个表格标题显示（类似 <caption>）

对象插入规则:

如果一个table-cell元素的父元素不是table-row元素，则会在该table-cell元素及其父元素之间插入一个匿名table-row对象。所插入的这个对象将包含该table-cell元素的所有连续兄弟。

以此类推:

表标题：caption-side

定义表标题的摆放位置，到底是top还是bottom;

表单元格边框：border-collapse属性

collapse:表格中的单元格相互之间是合并的；

separate:表格中的单元格相互之间是分隔的；

边框间隔：border-spacing属性

如果border-collapse属性设置为分隔状态，那么border-spacing属性就是设置分隔的间隔到底有多大。可以分别设置宽度和高度；

表大小：table-layout:auto,fixed,inherit

# 第十二章： 列表与生成内容

1 : 列表类型： list-style-type

disc:使用一个实心圆作为列表项标志；

circle:使用一个空心圆作为列表项标志；

square:使用一个方块作为列表项标志；

decimal:1,2,3,4,5

decimal-leading-zero:01,02,03,04,05

none:没有标志

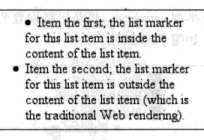
2：列表项图像: list-style-image:

ul {list-style-image:url(ohio.gif);}

列表类型就会变成一张图片；

3：列表标志位置： list-style-position：inside,outside

确定标志出现在列表内容之外还是在内容内部；



4 : 简写列表样式list-style：\*-type,\*-image,\*-position

li{list-style:url(ohio.gif) square inside;}

可以按任意顺序列出，这些值都可以忽略；

5：列表布局

margin,padding的使用；

生成内容：

1：插入生成内容：

a[href]:before{content:”(link)”;}

a[href]:before{content:url(pdf-doc-icon.gif);}

2 : 指定内容 :

h2:before{content:”<em>&para;</em>”;color:gray;}

3 : 插入属性值 :

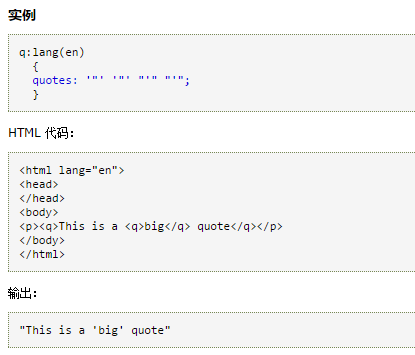
a[href]:after{content:attr(href);}

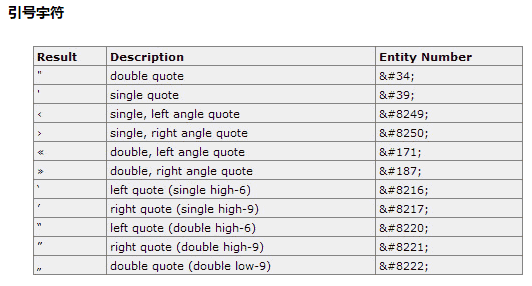
4 : 生成引号：quotes

none

string string string string

前两个值规定第一级引用嵌套，后两个值规定下一级引用嵌套；





5 : 计数器 counter-reset;

可以将小节计数器命名为subsection subsec ss甚至bob

h1{counter-reset:chapter;}

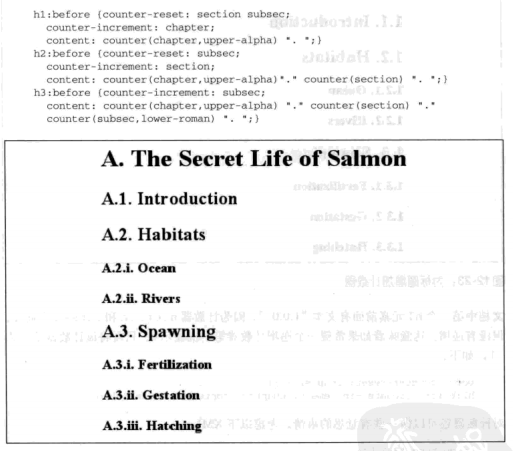
h1#ch4{counter-reset:chapter 4;}

设置计数器递增；

counter-increment:

ol li{counter-increment:ordered -1;}

改变计数器的样式：



# 第十三章： 用户界面样式

1 : 系统字体和颜色

caption:标题控件

icon:硬盘驱动器，文件夹和文件图标

menu:下拉菜单和菜单列表中文本使用的字体样式

message-box:对话框中文本使用的字体样式

small-caption:标签使用的字体样式

status-bar:窗口状态条中文本使用的字体样式

2 : 系统颜色

3 : 改变光标 cursor

url 需使用的自定义光标的 URL。

注释：请在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标。

default 默认光标（通常是一个箭头）

auto 默认。浏览器设置的光标。

crosshair 光标呈现为十字线。

pointer 光标呈现为指示链接的指针（一只手）

move 此光标指示某对象可被移动。

e-resize 此光标指示矩形框的边缘可被向右（东）移动。

ne-resize 此光标指示矩形框的边缘可被向上及向右移动（北/东）。

nw-resize 此光标指示矩形框的边缘可被向上及向左移动（北/西）。

n-resize 此光标指示矩形框的边缘可被向上（北）移动。

se-resize 此光标指示矩形框的边缘可被向下及向右移动（南/东）。

sw-resize 此光标指示矩形框的边缘可被向下及向左移动（南/西）。

s-resize 此光标指示矩形框的边缘可被向下移动（南）。

w-resize 此光标指示矩形框的边缘可被向左移动（西）。

text 此光标指示文本。

wait 此光标指示程序正忙（通常是一只表或沙漏）。

help 此光标指示可用的帮助（通常是一个问号或一个气球）。

4 : 轮廓

轮廓样式：outline-style

轮廓宽度：outline-width

轮廓颜色：outline-color

