目录

[CSS 1](#_Toc10242766)

[CSS规则集 1](#_Toc10242767)

[同时修改多个属性： 1](#_Toc10242768)

[多个元素修改同个属性 2](#_Toc10242769)

[选择器类型 2](#_Toc10242770)

[元素选择器 2](#_Toc10242771)

[类选择器 2](#_Toc10242772)

[ID选择器 2](#_Toc10242773)

[通用选择器 2](#_Toc10242774)

[属性选择器 3](#_Toc10242775)

[子串值属性选择器 3](#_Toc10242776)

[伪（pseudo）类选择器 3](#_Toc10242777)

[伪元素 4](#_Toc10242778)

[组合器 5](#_Toc10242779)

[颜色透明度 5](#_Toc10242780)

[CSS语句 6](#_Toc10242781)

[@-规则(At-rules) 6](#_Toc10242782)

[简写属性 7](#_Toc10242783)

[CSS背景图片 7](#_Toc10242784)

[背景裁剪 7](#_Toc10242785)

[层叠与继承 8](#_Toc10242786)

[继承 8](#_Toc10242787)

[区块模型 9](#_Toc10242788)

[溢流 10](#_Toc10242789)

[CSS框类型 10](#_Toc10242790)

[图片居中 11](#_Toc10242791)

# CSS

为HTML元素添加样式

1. **外部样式表**：

需要在HTML文档中连接CSS文件，在<head>标签之间加入**<link>**元素代码：

<link href="styles/style.css" rel="stylesheet" type="text/css">

**href**后面的属性值为.css文件路径（这里为styles文件下的style.css文件）

1. **内部样式表**：在<head>中用**<style>**元素包裹内容
2. **内联样式**：

<h1 style="color: blue;background-color: yellow;border: 1px solid black;">Hello World!</h1>

<p style="color:red;">This is my first CSS example</p>

CSS规则集

选择器+声明块=规则集（规则）



**选择器**（selector）：选择元素 给不同元素添加样式只需要更改选择器

**声明**（declaration）：**属性（value）+属性值（property）**制定添加样式元素的属性

同时修改多个属性：

p {

color: red;

width: 500px;

border: 1px solid black;

}

多个元素修改同个属性

将不同的选择器用逗号分开，如：

p, li, h1 {

color: red;

}

# 选择器类型

元素选择器**（标签或类型选择器）**

p 选择<p>，多个元素相同用逗号隔开

p ,div{

color: red;

}

类选择器**,**

点“.”以及类后面的类名组成(**.my-class**)，类名是**class**属性中没有空格的值

.done {

text-decoration: line-through;

}

多个元素可以具有相同类名，单个元素可以有多个类名

<li class="first done">Create an HTML document</li>

（空格将类名隔开，上例类名是first和done）

ID选择器

“#”符号和元素特定id组成(**#my-id**)

任何元素都可以用id属性设置唯一的id名称一个ID名称必须在文档中是唯一的

#polite {

font-family: cursive;

}

通用选择器

**\*** 选择一个页面中的所有元素

属性选择器

* [attr]：该选择器选择包含 attr 属性的所有元素，不论 attr 的值为何。
* [attr=val]：该选择器仅选择 attr 属性被赋值为 val 的所有元素。
* [attr~=val]：该选择器仅选择具有 attr 属性的元素，而且要求 val 值是 attr 值包含的被空格分隔的取值列表里中的一个。

如

[data-vegetable~="spicy"] {

color: red;

}

/\* 具有"data-vegetable"属性且属性值包含"spicy"的元素，

即使元素的属性中还包含其他属性值\*/

本例中的**data-\***属性被称为 [数据属性](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Global_attributes/data-*)。它们提供了一种在HTML属性中存储自定义数据的方法，由此，这些数据可以轻松地被提取和使用。

**img[src]** 选择<img src=”myimage.png”>而不是<img>

子串值属性选择器

* [attr|=val] : 选择attr属性的值是 val 或值以 val- 开头的元素（注意，这里的 “-” 不是一个错误，这是用来处理语言编码的）。
* [attr^=val] : 选择attr属性的值以 val 开头（包括 val）的元素。
* [attr$=val] : 选择attr属性的值以 val 结尾（包括 val）的元素。
* [attr\*=val] : 选择attr属性的值中包含子字符串 val 的元素（一个子字符串就是一个字符串的一部分而已，例如，”cat“ 是 字符串 ”caterpillar“ 的子字符串）。

伪（pseudo）类选择器

选择元素的某部分或者特定状态下的特定元素（比如鼠标指针悬停）， **a:hover** 仅在鼠标指针悬停在链接上时选择<a>

**伪类**是一个以冒号(:)作为前缀，被添加到一个选择器末尾的关键字如上面的hover

/\* 我们想让被访问过的链接呈蓝色\*/

a:visited {

color: blue;

}

/\* 当光标悬停于链接，键盘激活或锁定链接时，链接呈现高亮 \*/

a:hover,

a:active,

a:focus {

color: darkred;

text-decoration: none;

}

**:nth-of-type(<nth>)**设置条纹效果，为不同行添加不同样式

p:nth-of-type(2n+1) { /\*奇数段\*/

color: red;

}

p:nth-of-type(2n) { /\*偶数段\*/

color: blue;

}

p:nth-of-type(1) { /\*第一段\*/

font-weight: bold;

}

p:nth-of-type(4n) { /\*每组兄弟元素中选择第四个<p>元素\*/

color: lime;

}

**:nth-child(n)** 第n行

伪元素

前缀为两个冒号 (::)，同样是选择元素的某个部分

* [::after](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::after)
* [::before](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::before)
* [::first-letter](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-letter)
* [::first-line](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::first-line)
* [::selection](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::selection)
* [::backdrop](https://developer.mozilla.org/zh-CN/docs/Web/CSS/::backdrop)

/\*结合伪类所有含有"href"属性并且值以"http"开始的元素，

将会在其内容后增加一个箭头\*/

[href^=http]::after {

content: '⤴';

}

# 组合器

| **名称** | **组合器** | **选择** |
| --- | --- | --- |
| 选择器组 | A,B | 匹配满足A（和/或）B的任意元素. |
| 后代选择器 | A B | 匹配B元素，满足条件：B是A的后代结点（B是A的子节点，或者A  的子节点的子节点） |
| 子选择器 | A > B | 匹配B元素，满足条件：B是A的直接子节点 |
| 相邻兄弟选择器 | A + B | 匹配B元素，满足条件：B是A的下一个兄弟节点（AB有相同的父结点，  并且B紧跟在A的后面） |
| 通用兄弟选择器 | A ~ B | 匹配B元素，满足条件：B是A之后的任意一个兄弟节点（AB有相同的父节点，  B在A之后，但不一定是紧挨着A） |

/\* 所有table里的thead里的所有th \*/

table thead th {

color: white;

background: black;

}

/\* 所有table里的tbody里的所有td（第一个除外），每个td都是由它上边的td选择 \*/

table tbody td + td {

text-align: center;

}

/\* 定位在“with-currency”类中拥有属性lang并且这个属性值为en-US的元素中的，最后td(:last-child)节点的前面（::before）\*/

.with-currency[lang="en-US"] td:last-child::before {

content: '$';

}

# 颜色透明度

**RGBA和HSLA**

background-color: rgba(255,0,0,0.5);

background-color: hsla(240,100%,50%,0.5);

前三位表示颜色，第四个值是0~1之间的值，表示透明度（0完全透明，1完全不透明）

**不透明度opacity**

p:nth-child(2) {

background-color: rgb(255,0,0);

opacity: 0.5;

}

**注：**rgba和hsla只设置背景透明，opacity文本和背景都透明

# CSS语句

****@-规则(At-rules)****

* [@charset](https://developer.mozilla.org/zh-CN/docs/Web/CSS/@charset) 和 [@import](https://developer.mozilla.org/zh-CN/docs/Web/CSS/@import) （元数据）

@import 'custom.css';

向当前 CSS 导入其它 CSS 文件

* [@media](https://developer.mozilla.org/zh-CN/docs/Web/CSS/@media) 或 [@document](https://developer.mozilla.org/zh-CN/docs/Web/CSS/@document) （条件信息，又被称为**嵌套语句**)

@media (min-width: 801px) {

body {

margin: 0 auto;

width: 800px;

}

}

只有在页面宽度超过801像素时才会应用

* [@font-face](https://developer.mozilla.org/zh-CN/docs/Web/CSS/@font-face) （描述性信息）

**旋转动画**

@keyframes rotate {

0% {

transform: rotate(0deg);

}

100% {

transform: rotate(360deg);

}

}

p {

color: red;

width: 100px;

font-size: 40px;

transform-origin: center;

}

p:hover {

animation-name: rotate;

animation-duration: 0.6s;

animation-timing-function: linear;

animation-iteration-count: 5; /\*执行5次\*/

}

简写属性

如font,background,padding,border,margin等是简写属性，如

padding: 10px 15px 15px 5px;

/\* 在padding和margin这样的简写属性中，值赋值的顺序是top、right、bottom、left。

它们还有其他简写方式，例如给padding两个值，则第一个值表示top/bottom，第二个值表示left/right \*/

background: red url(bg-graphic.png) 10px 10px repeat-x fixed;

# CSS背景图片

**background-image**和另其他**background-\***属性是用来放置背景图片的

p {

background-image: url("images/dinosaur.jpg");

}

只作为装饰，没有语义上的意义，不能有任何备选文本。**url（“图片地址”）**

背景裁剪

**background-clip**属性

.default { background-clip: border-box; }

.padding-box { background-clip: padding-box; }

.content-box { background-clip: content-box; }

**轮廓outline**

outline: 1px solid white;



# 层叠与继承

**重要性**

使规则宗师优先于其他规则：**!important** 把它加在属性值后面

**专用性**

衡量选择器的具体程度。元素选择器<类选择器<id选择器<!important

1. 千位：如果声明是在[style](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Global_attributes#attr-style) 属性中该列加1分（这样的声明没有选择器，所以它们的专用性总是1000。）否则为0。
2. 百位：在整个选择器中每包含一个ID选择器就在该列中加1分。
3. 十位：在整个选择器中每包含一个类选择器、属性选择器、或者伪类就在该列中加1分。
4. 个位：在整个选择器中每包含一个元素选择器或伪元素就在该列中加1分。

/\* specificity: 0033 \*/

div div .nav:nth-child(2) a:hover {

border: 10px double black;

}

通用选择器 (\*), 复合选择器 (+, >, ~, ' ') 和否定伪类 (:not) 在专用性中无影响。

**源代码次序**

如果多个相竞争的选择器具有相同的重要性和专用性，后面的规则将战胜前面的。

继承

应用于某元素的一些属性值由钙元素的子元素继承，有些则不会

* [inherit](https://developer.mozilla.org/zh-CN/docs/Web/CSS/inherit)：属性值设置为与其父元素一样。
* [initial](https://developer.mozilla.org/zh-CN/docs/Web/CSS/initial) ：与浏览器默认样式表中设置的值一样。如果浏览器默认样式表中没有设置值，并且该属性是自然继承的，那么该属性值就被设置为 inherit。
* [unset](https://developer.mozilla.org/zh-CN/docs/Web/CSS/unset) ：该值将属性重置为其自然值，即如果属性是自然继承的，那么它就表现得像 inherit，否则就是表现得像 initial。
* [revert](https://developer.mozilla.org/zh-CN/docs/Web/CSS/revert) ：属性值被设置成自定义样式所定义的属性（如果被设置）， 否则属性值被设置成用户代理的默认样式。

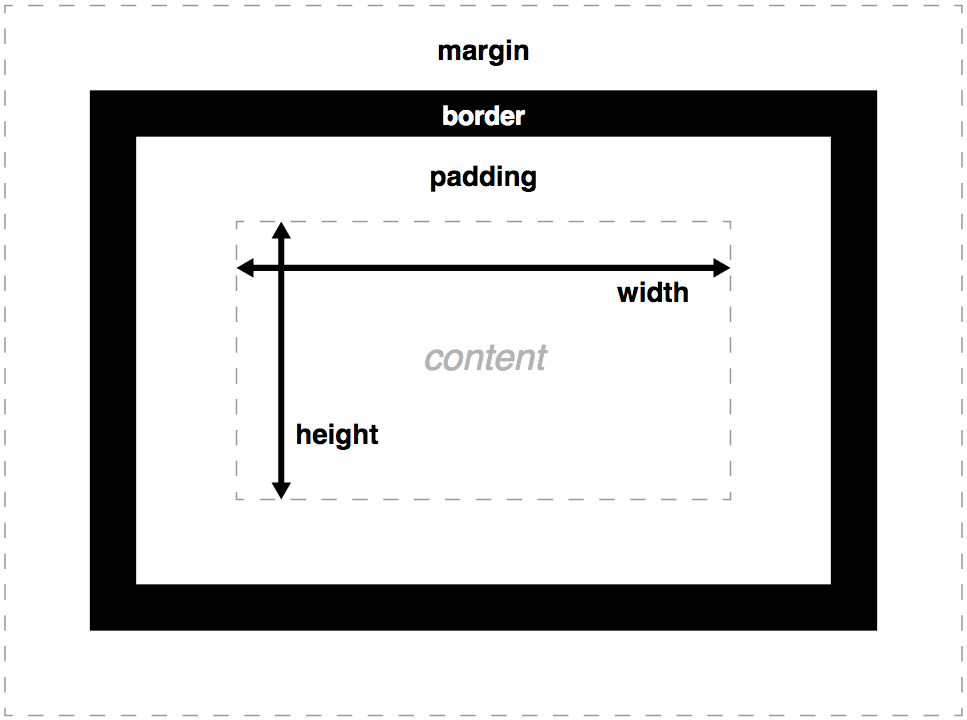
# 区块模型

尺寸，颜色，位置等

**padding**：内边距，围绕着内容（如段落）的空间，

**border**：边框，紧接着内边距的线，默认大小为0

**margin**：外边距，围绕元素外部的空间 ，以上三个属性值依次上右下左四个边



**width** ：元素的宽度

**background-color** ：元素内容和内边距底下的颜色

**color** ：元素内容（通常为文本）的颜色

**text-shadow** ：为元素内的文本设置阴影

**display** ：设置元素的显示模式

body {

width: 600px; /\*强制页面永远保持 600 像素宽\*/

margin: 0 auto;

background-color: #FF9500;

padding: 0 20px 20px 20px;

border: 5px solid black; /\* 设置 5 像素的黑色实线边框\*/

}

为 **margin 或 padding** 等属性设置两个值时，第一个值代表元素的上方**和**下方（在这个例子中设置为 0），而第二个值代表左边**和**右边（在这里，auto是一个特殊的值，意思是水平方向上左右对称）

* padding: 0 20px 20px 20px; —— 我们给内边距设置了四个值来让内容四周产生一点空间。这一次我们不设置上方的内边距，设置右边，下方，左边的内边距为20像素。值以上、右、下、左的顺序排列。

浏览器会没有任何 CSS 的情况下给 [<h1>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/h1)等元素设置一些**默认样式，**为了去掉那个间隙，我们通过设置 margin: 0; 覆盖掉默认样式。

溢流

用绝对值设置框的大小时，可能会不适合放置内容，使用**overflow**属性控制发生：

* auto:溢流的内容被隐藏，然后出现滚动条来滚动查看所有内容。
* hidden:溢流的内容被隐藏。
* visible:溢流的内容被显示在盒子的外边（默认行为）
* .autoscroll { overflow: auto; }
* .clipped { overflow: hidden; }
* .default { overflow: visible; }

CSS框类型

**dispaly**属性：

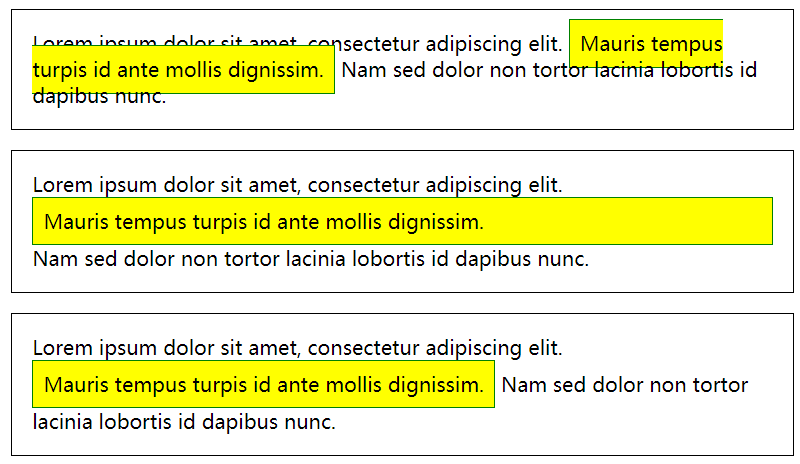
* 块框（ block box）是定义为堆放在其他框上的框（例如：其内容会独占一行），而且可以设置它的宽高，之前所有对于框模型的应用适用于块框 （ block box）
* 行内框（ inline box）与块框是相反的，它随着文档的文字流动（例如：它将会和周围的文字和其他行内元素出现在同一行，而且它的内容会像一段中的文字一样随着文字部分的流动而打乱），对行内盒设置宽高无效，设置padding, margin 和 border都会更新周围文字的位置，但是对于周围的的块框（ block box）不会有影响。
* 行内块状框（inline-block box） 像是上述两种的集合：它不会重新另起一行但会像行内框（ inline box）一样随着周围文字而流动，而且他能够设置宽高，并且像块框一样保持了其块特性的完整性，它不会在段落行中断开。（在下面的示例中，行内块状框会放在第二行文本上，因为第一行没有足够的空间，并且不会突破两行。然而，如果没有足够的空间，行内框会在多条线上断裂，而它会失去一个框的形状。）

默认状态下display属性值，块级元素display: block ，行内元素display: inline

.inline { display: inline; }

.block { display: block; }

.inline-block { display: inline-block; }



图片居中

img {

display: block;

margin: 0 auto;

}

使图像居中来使页面更美观，可以复用body的margin：0 auto，但是需要一点调整，<body>元素是**块级元素**，意味着它占据了页面的空间并且能够赋予外边距和其他改变间距的值，图片是**内联元素**，不具备块级元素的一些功能。所以为了使图像有外边距，我们必须**使用display：block给予其块级行为**。

当图片溢出 body 并占据页面的其他位置时，可以：  
1）使用 [图片编辑器](https://en.wikipedia.org/wiki/Raster_graphics_editor) 来减小图片宽度；  
2）用 CSS 限制图片大小，即减小<img> 元素 [width](https://developer.mozilla.org/zh-CN/docs/Web/CSS/width) 属性的值（比如400px）。