文本外观属性

html的效果不理想，css3提供了一些文本外观属性

color文本颜色：

用于定义文本颜色：有以下三种方式：

- 预定义颜色：比如red blue yellow
- 十六进制：由#开头，六位十六进制数组成，比如#FF0000，每两位十六进制对应一个RGB值
- RGB代码，比如红色rgb(256,0,0)或者rgb(100%,0%,0%)

> 使用%方法定义rgb的时候，所有的百分号都不能省略

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        p {
            color: rgb(100%, 20%, 0%)
        }

        h2 {
            color: blue
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <p>这是一个文本</p>
</body>

</html>
```

间距定义属性

这三个是间距定义符：

letter-spacing 字间距

可以为正数也可以为负数

word-spacing（英文）单词间距

可以为正数也可以为负数

line-height行间距

单位为三种：像素px、相对值em、百分比%

下面用一个列子表示：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        p {
            color: rgb(100%, 20%, 0%);
            font-size: 17px;
            line-height: 70px;
            letter-spacing: -3px;
            word-spacing: 40px;
            /* 行高为20px，字体大小为17px，颜色为红色,字母之间的间距缩短6px，单词之间的间距
为20px */
        }

        h2 {
            color: blue;
            line-height: 1.5em;
            /* 行高为1.5倍 */
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <h2>这是一个测试样例</h2>
    <p>这是一个p标签文本</p>
    <p>This is a lable of p</p>
</body>

</html>
```

Document

这是一个测试样例

这是一个p标签文本

This is a lable of p

可以看到，输出的p标签更加地紧凑，并且行间距非常大，英文单词之间的间距也有所增加

文本格式属性

text-transform文本转换

用于转换英文字母的属性：

- none 不转换（默认值）
- capitalize：首字母大写
- uppercase：全部字符转为大写
- lowercase：全部字符转为小写

text-decoration文本装饰

- none 没有装饰（默认值）
- underline：下划线
- overline：上划线
- line-through：删除线

这个属性可以赋以多个值，来增加文本的装饰

text-align水平对齐

相当于html5中的align对其属性，有以下几个取值：

- left：左对齐（默认值）
- right：右对齐
- center：居中

text-indent首行缩进

属性值可以是不同单位的数值、em字符宽度的倍数、相对于浏览器窗口的百分比。

允许使用负值

white-space空白符处理

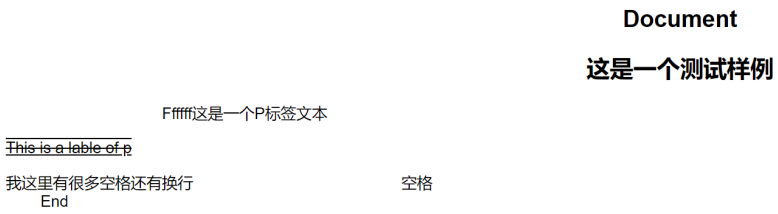
由于html在网页中不会显示空格或者换行，使用这个属性可以达到保留多个空格的效果

- normal 常规，默认值，最多保留一个空格，不支持换行，到达浏览器边界时自动换行
- pre 保留所有空格，换行
- nowarp：空格换行无效

下面是一个综合示例

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        .f {
            text-transform: capitalize;
            /* 首字母大写 */
```

```
            text-indent: 10em;
            /* 首行缩进十倍 */
        }

        .one {
            text-decoration: line-through overline underline;
            /* 下划线 上划线 删除线 */
        }

        .two {
            white-space: pre;
        }

        h2 {
            text-align: center;
            /* 标题居中 */
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <h2>这是一个测试样例</h2>
    <p class="f">ffffff这是一个p标签文本</p>
    <p class="one">This is a lable of p</p>
    <p class="two">我这里有很多空格还有换行
     空格
        End
    </p>
</body>

</html>
```



文本特殊效果

text-shadow阴影效果

有几个属性值来完成这个阴影效果：

- h-shadow：水平阴影的距离
- v-shadow：垂直阴影的距离
- blur：设置模糊半径
- color：设置阴影颜色

此外，属性值可以有多组，用逗号隔开，例如：

```
text-shadow: 2px 2px 3px red, 3px 2px 3px yellow;
```

示例

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        h2 {
            text-align: center;
            /* 标题居中 */
            text-shadow: 2px 2px 3px red;
        }

        p {
            text-shadow: 2px 2px 3px red, 3px 2px 3px yellow;
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <h2>这是一个测试样例</h2>
    <p>ffffff这是一个p标签文本</p>
    <p>This is a lable of p</p>
</body>

</html>
```

Document

这是一个测试样例

ffffff这是一个p标签文本

This is a lable of p

text-overflow标示对象内溢出文本

该属性标示了对象溢出某个范围的文本，有两个属性：

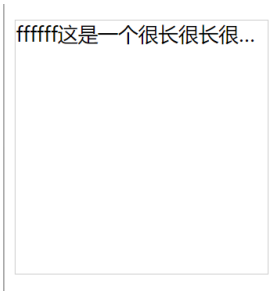
- clip：修剪溢出文本，不显示标记"..."
- ellipsis：用省略标记"..."标示被修剪的文本，省略标记插入的是最后一个字符

下面的例子：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        h2 {
            text-align: center;
            /* 标题居中 */
            text-shadow: 2px 2px 3px red;
```

```
        }

        p {
            width: 200px;
            height: 200px;
            border: 1px solid #ccc;
            /* 圈定文本范围 */
            white-space: nowrap;
            /* 强制文本不能换行 */
            overflow: hidden;
            text-overflow: ellipsis;
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <h2>这是一个测试样例</h2>
    <p>ffffff这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本这是一个很长很长很长的p标签文本</p>
    <!-- <p>This is a lable of p</p> -->
</body>

</html>
```

效果：



> 在使用text-overflow的时候，需要先使用overflow属性进行隐藏文本

CSS高级特性

CSS层叠性和继承性

层叠性

层叠性就是CSS最基本的 特性，CSS可以支持多种样式的叠加

比如：

```
<p class="special" id="one">css层叠性</p>
```

继承性

父标签的CSS代码可以被子标签继承

比如CSS中定义标签选择器body，那么例如p标签、div、h1、h2等等都具有body的样式

下面的属性不具有继承性：

- 边框属性
- 外边距属性
- 内边距属性
- 背景属性
- 定位属性
- 布局属性
- 元素宽高属性

> 此外，标题文本不采用父类型的字体大小属性，本身h系就有默认字号

CSS优先级

CSS优先级用权重来表示，即权重越大优先级越高

继承而来的表示权重为0，即最低的权重，可以被任何的CSS代码覆盖，即内嵌的代码永远具有最高的权重

就近原则

还有一种方法就是进行权重的叠加

```
div div div  div div div div div div div div div  div div div{

}
```

此时这个选择器的权重就是15个div，就是15

CSS3选择器

属性选择器

E[att^=value]选择器

E是标记，att是属性名，value是属性值，^表示的是开头的元素

表示的是以value符号开头的元素需要使用的渲染

属性选择器可以直接选择某一个标签的属性值来进行渲染

E[att$=value]选择器

表示的是以value符号结尾的元素需要使用的渲染

E[att*=value]选择器

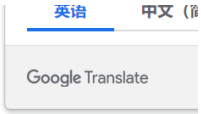
表示的是属性值包含value的元素需要使用 的渲染

综合例子

下面是一个例子:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        p[id^="one"] {
            color: red;
        }

        p[id$="567"] {
            color: green;
        }

        p[id*=go] {
            color: blue;
            font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS',
sans-serif;
        }
    </style>
</head>

<body>
    <h2>Document</h2>
    <h2>这是一个测试样例</h2>
    <p id="one123">这是一个p标签文本,使用的id选择器是one123</p>
    <p id="two567">这是一个p标签文本,使用的id选择器是two567</p>
    <p id="45678go456789">这是一个p标签文本,使用的id选择器是45678go456789</p>

    <!-- <p>This is a lable of p</p> -->
</body>

</html>
```

显示效果:



Document

这是一个测试样例

这是一个p标签文本,使用的id选择器是one123

这是一个p标签文本,使用的id选择器是two567

这是一个p标签文本,使用的id选择器是45678go456789

关系选择器

子代选择器（>）

子代选择器可以作用于只适用某一个标签的子标签使用

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        h2>strong {
            color: red;
        }
    </style>
</head>
<body>
    <h2>Document</h2>
    <h2>这是一个<strong>测试</strong>样例</h2>
    <p>这是一个<strong>P标签</strong>文本</p>
</body>

</html>
```

效果：

Document

这是一个**测试**样例

这是一个**P标签**文本

这个例子中，红色只作用于h2中的strong

兄弟选择器（+，~）

临近兄弟选择器（+）

两个选择器用+连接，在html中，这两个标签都有一个父标签

此时，紧跟前面一个标签的后一个标签会起作用：

假如h2+p，那么起作用的就是p

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        h2+p {
            color: red;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
```

```
    <p>这是一个P标签文本</p>
    <h2>这是一个h2标签文本</h2>
    <p>这是一个P标签文本</p>
    <p>这是一个P标签文本</p>
</body>

</html>
```

这是一个测试样例

这是一个P标签文本

这是一个h2标签文本

这是一个P标签文本

这是一个P标签文本

普通兄弟选择器：

普通兄弟选择器用~连接表示，但是不用相邻即可渲染后一个元素

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        h2~p {
            color: red;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    <p>这是一个P标签文本</p>
    <h2>这是一个h2标签文本</h2>
    <p>这是一个P标签文本</p>
    <p>这是一个P标签文本</p>
</body>

</html>
```

效果：

这是一个测试样例

这是一个P标签文本

这是一个h2标签文本

这是一个P标签文本

这是一个P标签文本

结构化伪类选择器

:root选择器

:root选择器用于匹配文档的根元素，即<html>元素

在此所有的元素都可以生效

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        /* * {
            color: blue;
        }
         */

         :root {
            color: red;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    <p>这是一个P标签文本</p>
    <h2>这是一个h2标签文本</h2>
    <p>这是一个P标签文本</p>
    <p>这是一个P标签文本</p>
</body>

</html>
```

效果：

这是一个测试样例

这是一个P标签文本

这是一个h2标签文本

这是一个P标签文本

这是一个P标签文本

> 注意：当:root和*两个选择器一起使用的时候，通配符选择器的优先级更高

:not选择器

想排除某个元素的下面的子元素的使用：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        body *:not(h2) {
            color: red;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    <p>这是一个P标签文本</p>
    <h2>这是一个h2标签文本</h2>
    <p>这是一个P标签文本</p>
    <p>这是一个P标签文本</p>
</body>

</html>
```

使用方法

```
父标签 *:not(子标签)
```

效果：

这是一个测试样例

这是一个P标签文本

这是一个h2标签文本

这是一个P标签文本

这是一个P标签文本

:noly-child选择器

叫做唯一子元素选择器，当某个父标签只有那一个子元素的时候会对其进行渲染

例如：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        li:only-child {
            color: red;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1：
    <ul>
        <li>子元素1</li>
        <li>子元素2</li>
    </ul>
    父标签2：
    <ul>
        <li>子元素1</li>
    </ul>
</body>

</html>
```

li作为子元素只有一个是父标签2的子元素1

这是一个测试样例

父标签1：

- 子元素1
- 子元素2

父标签2：

- 子元素1

:first-child 和:last-child选择器

为父元素选择第一个和最后一个子元素

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        li:first-child {
            color: red;
```

```
            font-weight: 300;
        }

        li:last-child {
            color: green;
            font-family: "宋体";
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1:
    <ul>
        <li>子元素1</li>
        <li>子元素2</li>
        <li>子元素3</li>
        <li>子元素4</li>
        <li>子元素5</li>
    </ul>
</body>

</html>
```

冒号的前面输入的是选择的子标签

效果:

这是一个测试样例

父标签1:

- 子元素1
- 子元素2
- 子元素3
- 子元素4
- 子元素5

:nth-child(n) 和nth-last-child(n) 选择器

这是:first-child 和:last-child选择器的拓展，里面的n就是第几个子元素，从1开始计数

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        li:nth-last-child(5) {
            color: red;
            font-weight: 300;
        }

        li:nth-child(3) {
            color: green;
            font-family: "宋体";
        }
    </style>
```

```
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1:
    <ul>
        <li>子元素1</li>
        <li>子元素2</li>
        <li>子元素3</li>
        <li>子元素4</li>
        <li>子元素5</li>
    </ul>
</body>

</html>
```

效果:

这是一个测试样例

父标签1:

- 子元素1
- 子元素2
- 子元素3
- 子元素4
- 子元素5

:nth-of-type(n)和:nth-last-of-type(n)选择器

用于匹配属于父元素特定类型的标签:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        li:nth-of-type(odd) {
            color: red;
        }

        li:nth-of-type(even) {
            color: brown;
        }

        h2:nth-last-of-type(1) {
            color: yellowgreen;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1:
    <ul>
        <li>子元素1</li>
```

```
        <li>子元素2</li>
        <li>子元素3</li>
        <li>子元素4</li>
        <li>子元素5</li>
    </ul>
</body>

</html>
```

效果：

这是一个测试样例

父标签1：

- 子元素1
- 子元素2
- 子元素3
- 子元素4
- 子元素5

:empty选择器

选择没有使用过的空标签进行渲染

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        li {
            color: blue;
        }

         :empty {
            background-color: greenyellow;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1:
    <ul>
        <li>子元素1</li>
        <li>子元素2</li>
        <li></li>
        <li>子元素3</li>
        <li>子元素4</li>
        <li>子元素5</li>
    </ul>

</body>
```

```
</html>
```

效果：

这是一个测试样例

父标签1：

- 子元素1
- 子元素2
-
- 子元素3
- 子元素4
- 子元素5

:target选择器

:target选择器会对html瞄点进行渲染

例如：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
         :target {
            background-color: yellow;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1：
    <ul>
        <li><a href="#1">子元素1</a></li>
        <li><a href="#2">子元素2</a></li>
    </ul>
    <p id="1">aaa</p>
    <p id="2">bbb</p>

</body>

</html>
```

单机子元素1和子元素2都会让那一条背景色变黄，如图：

这是一个测试样例

父标签1:

- 子元素1
- 子元素2

aaa

bbb

伪元素选择器

:before选择器

被选元素插入前面的内容，需要配合content属性进行指定插入的内容

:after选择器

插入到被选择元素的后面

综合样例

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        p:before {
            content: ":before标签的使用，插入到p标签的前面";
            color: royalblue;
            font-size: 20px;
        }

        li:after {
            content: ":after标签选择器的使用，插入到li标签的后面";
            color: aqua;
        }
    </style>
</head>

<body>
    <h2>这是一个测试样例</h2>
    父标签1:
    <ul>
        <li><a href="#1">子元素1</a></li>
        <li><a href="#2">子元素2</a></li>
    </ul>
    <p id="1">aaa</p>
    <p id="2">bbb</p>

</body>

</html>
```

效果：

这是一个测试样例

父标签1：

- 子元素1:after标签选择器的使用，插入到li标签的后面
- 子元素2:after标签选择器的使用，插入到li标签的后面

:before标签的使用，插入到p标签的前面aaa

:before标签的使用，插入到p标签的前面bbb

链接伪类

用于a标签的渲染：

- a:link{} 未访问的时候超链接的渲染
- a:visited 访问后超链接的渲染
- a:hover 鼠标悬停的时候超链接的渲染
- a:active 鼠标单机不动时超链接的渲染

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style type="text/css">
        a:link {
            color: palegoldenrod;
        }

        a:visited {
            color: aquamarine;
        }

        a:hover {
            color: red;
        }

        a:active {
            color: blue;
        }
    </style>
</head>

<body>
    <a href="https://www.baidu.com">Document</a>
    <a href="#">aaa</a>
    <a href="#">bbbb</a>
    <a href="#">Document</a>

</body>

</html>
```

> Document aaa bbbb Document