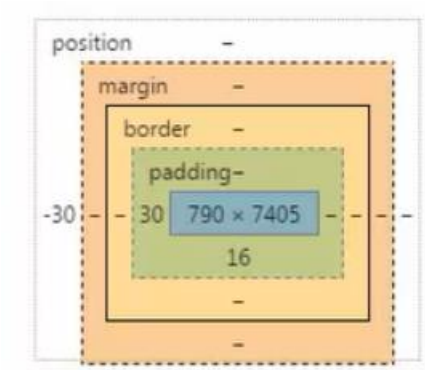
1、CSS3 介绍：

在网页制作时采用 CSS 层叠样式表技术，可以有效的对页面的布局、字体、颜色、背景和其他效果实现更加精确的控制。

CSS3 是 CSS 技术的升级版本，朝着**模块化**发展。以前只有一个整体模块，现在把整体模块分解成一些小模块，也就是加入了很多新模块，比如：

（1）盒子模型：



（2）文字特效：

（3）边框圆角：

（4）盒阴影：

文字旋转：

颜色渐变：

2、CSS3 新增的超级选择器

CSS 选择器能够对页面中的元素实现一对一、一对多、多对一的控制，可以大幅度提高开发人员编写或者修改页面样式的工作效率。CSS 定义语句：选择器{样式}，其中“选择器”指明了“样式”的作用对象，即作用于页面中的哪些元素

（1）属性选择器：（包括三个）

①**[att*=val]**:如果某元素的 att 属性的属性值**包含** val 指定的字符串，则该元素使用该样式，例如：[id*=section1]{ background: #f60;}

②**[att^=val]**:如果某元素的 att 属性的属性值**以** val 指定的字符串**开头**，则该元素使用该样式例如：[id^=sec]{ background: blue; }

③**[att$=val]**:如果某元素的 att 属性的属性值**以** val 指定的字符串**结尾**，则该元素使用该样式

```
[id$=a]{ background: green; }
[id$=_a]{ background: #f60; }
/*如果属性值最末尾字符前面带连接符（-），则必须在连接符（-）前面加反斜线（\）*/
[id$=\-1]{ background: red; }
```

具体 demo 如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>css3 中的属性选择器小案例</title>
    <style type="text/css">
        /*after 表示在符合筛选条件的元素后面插入新的内容及样式*/
        a[href$=html]:after{
            content:'网页文件';
            color:green;
        }
        a[href$=png]:after{
            content: '图片文件';
```

```
            color:green;
        }
    </style>
</head>
<body >
<a href="index.html">网站首页</a>
<a href="index.html">网站首页</a>
<a href="HTML5.png">网站图片</a>
<a href="HTML5.png">网站图片</a>
</body>
</html>
```

（2）（结构性）伪类选择器及伪元素选择器（冒号后面的单词都是筛选条件）

1）类选择器的语法：.class 属性值{样式}，其中选择器的名称可以随意定义，例如：

```
<style type="text/css">
        p.left{
            text-align: left;
            color:red;
        }
        p.right{
            text-align: right;
            color: green;
        }
</style>
<p class='left'>伪类选择器及伪元素</p>
<p class='right'>伪类选择器及伪元素</p>
```

2）伪类选择器与类选择器的区别在于：伪类选择器是 CSS 中已经定义好的选择器，不可以随意起名，最常见的几种伪类选择器有：（都是用来**定义链接样式**）

```
a:link{ color:#f60; }/*  未被访问的链接颜色*/
a:visited{ color:#87b291; }/*  已被访问的链接颜色*/
a:hover{ color:#6535b2; }/*  鼠标移动到链接上的颜色*/
a:active{ color:#f60; }/*  鼠标单击链接时的颜色*/
```

3）伪元素选择器：针对 CSS 中已经定义好的伪元素使用的选择器，语法是：

选择器:伪元素{样式} or 选择器.类名:伪元素{样式}，CSS 中主要有**四个**已经定义好的**伪元素**：

①:first-line 伪元素:用于向某个元素中的**第一行文字**使用样式，例如：

```
p:first-line{ color:#f60; }
<p>
    CSS 中主要有四个已经定义好的伪元素：<br/>
    first-line:用于向某个元素中的第一行文字使用样式
</p>
```

②:first-letter 伪元素:用于向某个元素中的**第一个字母或汉字**使用样式，例如：

```
p:first-letter{
    color:green;
    font-size:24px;
}
```

③**:before 伪元素**:用于在某个元素之前插入新的内容

```
li{
    list-style:none;/*去掉列表项默认圆点样式*/
}
li:before{
    content:'前面添加内容_';
    color:green;
}
<ul>
    <li><a href="index.html">伪类选择器</a></li>
    <li><a href="index.html">伪类选择器</a></li>
    <li><a href="index.html">伪类选择器</a></li>
</ul>
```

④**:after 伪元素**:用于在某个元素之后插入新的内容

```
li:after{
    content:'_后面追加内容';
    color:#f60;
}
```

4）结构性伪类选择器：（包括四个）

①:root 选择器：将样式应用到页面的根元素（HTML 元素）上，例如：

```
:root{ background: blue; } /*  整个页面的背景颜色都是蓝色*/
对比  body{ background: #fff; }   /*  只有 body 部分的背景颜色是白色*/
```

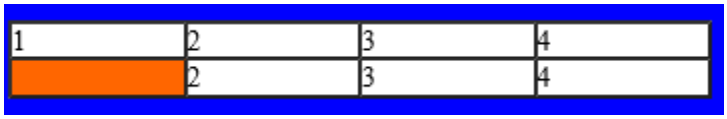
②:not 选择器：排除某些元素，使这些元素不应用该样式，例如：

```
body :not(h1){ background: #fff; } /* body 所有子元素中排除 h1 元素*/
```

③:empty 选择器：将样式应用到内容为空的元素上，例如：

```
:empty{ background: #f60; }
<table border="1" cellpadding="0" cellspacing="0" width='50%'>
    <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
    </tr>
    <tr>
        <td></td>
        <td>2</td>
        <td>3</td>
        <td>4</td>
    </tr>
</table>
```

效果图为：



④:target 选择器：将样式应用到页面中某个 target 元素上，该样式只有在用户点击了页面中的超链接，并且跳转到该 target 元素后才起作用，例如：

```
<style type="text/css">
    :target{
        background: #000;
        color:#fff;
    }
</style>
<body >
<!--单击 A 以后会链接到 id="A"的 div 元素，则该 div 元素即为要应用样式的 target 元素-->
<a href="#A">A</a>
<a href="#B">B</a>
<a href="#C">C</a>
<div id="A">
    <h2>标题</h2>
    <p>内容.....</p>
</div>
<div id="B">
    <h2>标题</h2>
    <p>内容.....</p>
</div>
<div id="C">
    <h2>标题</h2>
    <p>内容.....</p>
</div>
</body>
```

5）子元素过滤选择器：（有四个）

① **:first-child：**单独指定第一个子元素的样式

② **:last-child：**单独指定最后一个子元素的样式

③ **:nth-child(n)：**匹配正数第 n 个子元素

:nth-child(odd)：匹配正数第奇数个子元素

:nth-child(even)：匹配正数第偶数个子元素

④ **:nth-last-child：**匹配倒数第 n 个子元素

:nth-last-child：匹配倒数第奇数个子元素

:nth-last-child：匹配倒数第偶数个子元素

```
<style type="text/css">
    /*为第一个 li 子元素设置样式*/
    li:first-child{
        background: #f60;
    }
    li:last-child{
        background: green;
    }
    li:nth-child(3){
```

```
            background: red;
        }
        li:nth-last-child(3){
            background: blue;
        }
</style>
<ul>
        <li>第一个项目</li>
        <li>第二个项目</li>
        <li>第三个项目</li>
        <li>第四个项目</li>
</ul>
```

⑤:only-child:若某子元素是其父元素的唯一子元素，则将该子元素匹配出来

```
<style type="text/css">
        li:only-child{
            background: red;
        }
</style>
<ul>
        <li>唯一子标题</li>
</ul>
```

⑥循环使用样式：nth-child(An+B)，其中 A、B 代表任意常数，n 从 0 开始带入

```
<style type="text/css">
        /*为第 2n+1 个 li 元素指定红色，为第 2n+2 个 li 元素指定绿色，每两次一个循环*/
        li:nth-child(2n+1){
            background: red;
        }
        li:nth-child(2n+2){
            background: green;
        }
</style>
```

6）同类型的子元素选择器：nth-of-type、nth-last-of-type

因为在使用 nth-child 和 nth-last-child 时会产生这样的问题：系统会自动将 h1-h6 以及 p 元素都当做标题元素，造成样式的混乱，所以使用 nth-of-type 和 nth-last-of-type 会避免这类问题，因为它们只针对**同类型的子元素**

:nth-of-type(n)：匹配正数第 n 个同类型的子元素

:nth-last-of-type(n)：匹配倒数第 n 个同类型的子元素

具体 demo 如下：

```
<style type="text/css">
        h2:nth-of-type(odd){
            background: red;
        }
        h2:nth-last-of-type(even){
            background: green;
```

```
    }
</style>
<h1>同类型的子元素选择器</h1>
<h2>标题 1</h2>
<p>内容 1</p>
<h2>标题 2</h2>
<p>内容 2</p>
<h2>标题 3</h2>
<p>内容 3</p>
```

（3）UI 元素状态伪类选择器

UI 状态伪类选择器的特征是只有当元素处于某种状态时，指定的样式才起作用，在默认状态下不起作用，一共有 17 种，分别是：

1）E:hover 指定当鼠标指针**移动到元素上**时，元素所使用的样式，基本语法是：

元素:hover{CSS 样式}，也可以在元素中添加 type 属性，例如：input[type='text']:hover{样式}

2）E:active 指定元素**被激活（点击）时**使用的样式

3）E:focus 指定元素**获得光标聚焦点时**使用的样式，主要是在文本框控件获得聚焦点并进行文字输入时使用

以上三种选择器的 demo 如下：

注意：同一个元素如果定义了多个样式并且有冲突，则执行最后定义的样式，即越靠后定义的样式优先级越高

```
<style type="text/css">
    /*当鼠标移动到文本框上时背景变为绿色；当单击文本框时，文本框既处于 active 状态又处于 focus 状态，根据优先级使用 active 样式，背景变为蓝色；当松开鼠标时背景变为橙色*/
    input[type='text']:hover{
        background: green;
    }
    input[type='text']:focus{
        background: #f60;
    }
    input[type='text']:active{
        background: blue;
    }
</style>
<form>
    姓名：<input type="text" placeholder="请输入姓名">
</form>
```

4）E:enabled 指定元素处于**可用状态**时的样式

5）E:disabled 指定元素处于**不可用状态**时的样式

以上两种选择器的 demo 如下：

```
<style type="text/css">
        /*当文本框可用时,背景颜色变为绿色；当文本框不可用时,背景颜色变为灰色*/
        input[type='text']:enabled{
            background: green;
```

```
            color:#fff;
        }
        input[type='text']:disabled{
            background: #ccc;
        }
</style>
<form>
    姓名：<input type="text" placeholder="请输入姓名">
    <br/><br/>
    学校：<input type="text"  placeholder="请输入学校" disabled>
</form>
```

6）E:read-only 指定元素处于**只读状态时**的样式

7）E:read-write 指定元素处于**非只读状态时**的样式

以上两种选择器的 demo 如下：

```
<style type="text/css">
        /*当文本框既能读又能写时,字体颜色变为红色；当文本框只能读时,背景颜色变为黑色，字体变为白色*/
        input[type='text']:read-write{
            color:red;
        }
        input[type='text']:read-only{
            background:#000;
            color:#fff;
        }
</style>
<form>
    姓名：<input type="text" placeholder="请输入姓名">
    <br/><br/>
    地址：<input type="text"  placeholder="请输入地址" value="大连理工大学" readonly>
</form>
```

8）E:checked 指定表单中的 radio 单选按钮或者 checkBox 复选框中的**选项被选中后**的样式

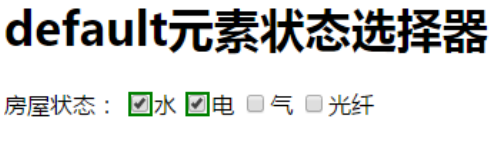
```
/*模拟一个选择房屋状况的控件，当选中某几个选项以后，会给它们再加上一个 2px 实线绿色的边框*/
input[type='checkbox']:checked{
    /*设置复选框的边框样式*/
    outline: 2px solid green;
}
<form>
    房屋状态：
    <input type="checkbox">水
    <input type="checkbox">电
    <input type="checkbox">气
    <input type="checkbox">光纤
</form>
```

最终效果图：

9）E:default 指定打开页面时**默认处于选中状态**的单选按钮或复选框选项的样式

```
/*模拟一个默认选中房屋状况的控件，会给默认选项加上 2px 实线 绿色的边框*/
input[type='checkbox']:default{
    /*设置复选框的边框样式*/
    outline: 2px solid green;
}
<form>
    房屋状态：
    <input type="checkbox" checked>水
    <input type="checkbox" checked>电
    <input type="checkbox">气
    <input type="checkbox">光纤
</form>
```

首次打开页面就会呈现下图的效果：



10）E:indeterminate 指定当页面刚打开时，一组单选框中没有一个单选框被选中，**整组单选框**都设定的样式

```
/*实现一个页面中的单选按钮都没有选中时，会给它们加上一个 2px 实线 绿色的边框*/
input[type='radio']:indeterminate{
    /*设置单选按钮的边框样式*/
    outline: 2px solid green;
}
<form>
    性别：
    <input type="radio" name="sex">男
    <input type="radio" name="sex">女
</form>
```

11）E::selection（中间必须是双冒号）指定元素处于**选中状态时**的样式

```
<style type="text/css">
        /*将网页中所有选中的文字的背景颜色改为绿色，字体颜色改为白色*/
        ::selection{
            background: green;
            color:#fff;
        }
        /*单独将文本框中选中文字的背景颜色改为橙色，字体颜色改为白色*/
        input[type='text']::selection{
            background: #f60;
```

```
            color:#fff;
        }
</style>
<h1>selection 元素状态选择器</h1>
<p>十三届全国人大一次会议 17 日上午选举习近平为中华人民共和国主席、中华人民共和国中央军事委员会主席</p>
<input type="text" placeholder="请输入文本">
```

12）E:invalid 指定当**元素内容不符合** HTML5 元素属性**规定的数据格式时**的样式，一般用来判断输入的数据类型是否正确

13）E:valid 指定当**元素内容符合** HTML5 元素**规定的数据格式时**的样式

以上两种选择器的 demo 如下：

```
<style type="text/css">
        /*通过具有 email 属性的表单判断输入是否正确，输入正确的内容为绿色，输入错误为红色*/
        input[type='email']:invalid{
            color:red;
        }
        input[type='email']:valid{
            color:green;
        }
</style>
<form>
    <input type="email" placeholder="请输入邮箱">
</form>
```

14）E:required 指定 允许使用并且**已经指定 required 属性**的 input 元素、select 元素、textarea 元素的样式，一般用来判断表单是否必须输入

15）E:optional 指定 允许使用但**未指定 required 属性**的 input 元素、select 元素、textarea 元素的样式

以上两种选择器的 demo 如下：

```
<style type="text/css">
        /*设置必填姓名的背景颜色为红色，可选填年龄的背景颜色为绿色*/
        input[type='text']:required{
            background:red;
            color:#fff;
        }
        input[type='text']:optional{
            background:green;
            color:#fff;
        }
</style>
<form>
    姓名:<input type="text" placeholder="必须输入姓名" required>
    <br/><br/>
    年龄:<input type="text" placeholder="非必须输入年龄">
```

```
</form>
```
16）E:in-range 指定元素实际的输入值**在限定的有效值范围之内时**的样式

17）E:out-of-range 指定元素实际的输入值**超过限定的有效值范围时**的样式

以上两种选择器的 demo 如下：

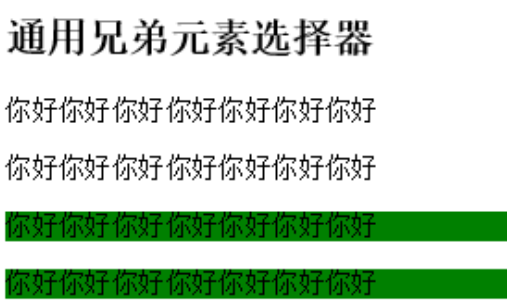
```
<style type="text/css">
        /*输入 0-100 以内的数字，若在范围内字体变成绿色，否则变成红色*/
        input[type='number']:in-range{
            color:green;
        }
        input[type='number']:out-of-range{
            color:red;
        }
</style>
<input type="number" min='0' max="100">
```

（4）通用兄弟元素选择器

指定某个元素**之后的所有同级兄弟元素**的样式，使用方法是：

某元素~同级的兄弟元素{CSS 样式}

```
<style type="text/css">
    /*选择 div 元素之后的所有同级兄弟元素 p，将其背景设置为绿色*/
    div~p{
        background: green;
    }
</style>
<div>
    <p>你好你好你好你好你好你好你好你好</p>
    <p>你好你好你好你好你好你好你好你好</p>
</div>
<p>你好你好你好你好你好你好你好你好</p>
<p>你好你好你好你好你好你好你好你好</p>
```



效果图为：

3、使用选择器在页面中插入内容

统一利用 **before 伪元素和 after 伪元素**在页面中插入文字、图像、项目编号等内容，插入的内容在选择器的 **content 属性**中定义

（1）插入文字：利用 content:”要插入的文字内容”，并且可以对插入的内容定义样式，例如：

```
<style type="text/css">
        h2:before{
```

```
            content: 'Title';
            color:#fff;
            background: green;
            padding: 1px 5px;
            margin-right:10px;
        }
</style>
<h2>使用选择器插入文字</h2>
```

注意：排除一些不需要插入内容的元素：在元素中添加 content:none 样式，例如：

```
<style type="text/css">
        h3:before{
            content: 'Title';
            color:#fff;
            background: green;
            padding: 1px 5px;
            margin-right:10px;
        }
        /*排除了具有 oncontent 类的 h3 元素，其他 h3 元素前面都插入上面的内容*/
        h3.oncontent:before{
            content:none;
        }
</style>
<h3>使用选择器插入文字</h3>
<h3 class="oncontent">使用选择器插入文字</h3>
<h3>使用选择器插入文字</h3>
```

Title 使用选择器插入文字

使用选择器插入文字

Title 使用选择器插入文字

效果图为：

（2）插入图片：利用 content:url(图片路径及名称)，例如：

```
<style type="text/css">
        h3:after{
            content: url(hot.gif);
        }
</style>
<h3>使用选择器插入图片</h3>
```

注意：利用选择器插入图片的好处是可以节省开发人员的工作量，例如可以为具有不同 class 值的标题插入不同的图片（**模仿论坛里的帖子分类**）：

```
<style type="text/css">
        h3.hot:after{
            content: url(hot.gif);
        }
```

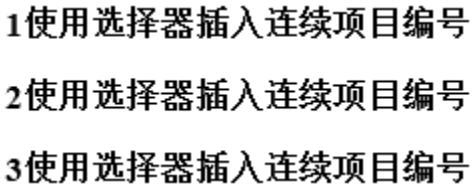
```
        h3.digest:after{
            content: url(digest.gif);
        }
        h3.xinrentie:after{
            content: url(xinrentie.gif);
        }
</style>
<h3 class="hot">使用选择器插入图片</h3>
<h3 class="digest">使用选择器插入图片</h3>0
<h3 class="hot">使用选择器插入图片</h3>
<h3 class="xinrentie">使用选择器插入图片</h3>
```

效果图为：

（3）插入连续项目编号：必须利用计数器样式 元素:before{content:counter(计数器名称)}，同时还要在元素中添加增值样式 元素{counter-increment: content 属性值指定的计数器名称}，注意计数器**默认使用数字编号**

```
<style type="text/css">
        h3:before{
            content:counter(h3jishuqi);
        }
        h3{
            counter-increment:h3jishuqi;
        }
</style>
<h3>使用选择器插入连续项目编号</h3>
<h3>使用选择器插入连续项目编号</h3>
<h3>使用选择器插入连续项目编号</h3>
```

效果图为：



注意：

1）给项目编号添加文字：可以在项目编号 counter(计数器名称)前后加入任意文字或符号，例如元素:before{content:”第”counter(计数器名称)”章”}，最后项目编号变成：第一章、第二章……

2）设定编号的样式：与设定文字样式的方法一样，利用 color、font-size、font-family 等属性

3）设定编号的类型：不仅可以插入数字编号，还可以插入字母编号或罗马编号，利用 content:counter(计数器名称，编号类型)，其中**编号类型就是 list-style-type 的属性值**，例如

“upper-alpha”表示大写字母编号，“upper-roman”表示罗马字母编号

4）编号嵌套功能配合编号重置功能使用：可以大编号嵌套中编号，中编号嵌套小编号，子标题实现编号重置，是在父标题的样式中设置{ counter-reset:子标题的计数器名称}，例如：

```
<style type="text/css">
        h3:before{
            content:counter(h3jishuqi)'.';
            color:green;
            font-size: 24px;
        }
        h3{
            counter-increment:h3jishuqi;
            counter-reset: h4jishuqi;
        }
        h4:before{
            content:counter(h4jishuqi)'.';
            color:#f60;
            font-size: 18px;
        }
        h4{
            counter-increment:h4jishuqi;
        }
</style>
<h3>使用选择器插入连续项目编号</h3>
<h4>编号嵌套、重置编号</h4>
<h4>编号嵌套、重置编号</h4>
<h3>使用选择器插入连续项目编号</h3>
<h4>编号嵌套、重置编号</h4>
<h4>编号嵌套、重置编号</h4>
<h3>使用选择器插入连续项目编号</h3>
```

效果图为：

1.使用选择器插入连续项目编号

1.编号嵌套、重置编号

2.编号嵌套、重置编号

2.使用选择器插入连续项目编号

1.编号嵌套、重置编号

2.编号嵌套、重置编号

3.使用选择器插入连续项目编号

5）大编号中嵌入中编号（模拟论文目录编号）：利用 content:counter(大编号计数器名称)’-’

counter (小编号计数器名称)，例如:

```
<style type="text/css">
        h3:before{
            content:counter(h3jishuqi)'.';
            color:green;
            font-size: 24px;
        }
        h3{
            counter-increment:h3jishuqi;
            counter-reset: h4jishuqi;
        }
        h4:before{
            content:counter(h3jishuqi)'-' counter(h4jishuqi);
            color:#f60;
            font-size: 18px;
        }
        h4{
            counter-increment:h4jishuqi;
        }
</style>
<h3>使用选择器插入连续项目编号</h3>
<h4>编号嵌套、重置编号</h4>
<h4>编号嵌套、重置编号</h4>
<h3>使用选择器插入连续项目编号</h3>
<h4>编号嵌套、重置编号</h4>
<h4>编号嵌套、重置编号</h4>
```

效果图为：

6）在字符串两边嵌套符号：使用 content:open-quote 及 content:close-quote 在字符串两边添加诸如括号、单引号、双引号等符号，然后在元素中添加样式{ quotes: 字符类型}，例如：

```
<style type="text/css">
        h3:before{
            content:open-quote;/*插入开头符号*/
        }
```

```
        h3:after{
            content:close-quote; /*插入结尾符号*/
        }
        h3{
            quotes: '（''）'; /*设置开头符号和结尾符号*/
        }
</style>
<h3>在字符串两边嵌套符号</h3>
```

效果图为：（在字符串两边嵌套符号）

4、文字阴影与自动换行

CSS3 中新增与文字相关的一些属性：比如 text-shadow 属性、word-break 属性、word-wrap 属性等

（1）text-shadow 属性：给页面的**文字加上阴影效果**

使用方法是 **text-shadow：length（必需）length（必需）length（可选）color（可选）**，包括四个参数：第一个参数是阴影离开文字的**横向位移距离**，第二个参数是阴影离开文字的**纵向位移距离**，第三个参数是**阴影模糊半径**（数值越大越模糊），第四个参数是**阴影颜色**（如果没有指定，阴影会默认使用文字的颜色），例如：

```
<style type="text/css">
        h3{
            text-shadow: 15px 15px 2px #f60;
        }
</style>
<h3>文字阴影 text-shadow</h3>
```

效果图为：文字阴影text-shadow

注意：可以给文字指定多个阴影，使用逗号“,”将多个阴影进行分割，并且可以针对每个阴影使用不同的颜色，例如：

```
<style type="text/css">
        h3{
            text-shadow: 15px 15px 3px #f60,30px 30px 3px red,45px 45px 3px green;
        }
</style>
```

效果图为：文字阴影text-shadow

（2）word-break 属性：实现页面**文字的自动换行**

浏览器默认的换行方式是中文不允许标点符号显示在开头，英文不会对单词拆分而是将整个单词换行，CSS3 中使用 word-break 属性处理换行有三个属性值，分别是 normal（使用浏览器的默认方式）、keep-all（只能在半角空格或连字符处换行）、break-all（允许在单词内换行），例如：p{ word-break:break-all; }

（3）word-wrap 属性：实现**长单词或长 URL 地址的自动换行**

该属性有两个属性值，分别是 normal（使用浏览器的默认处理方式）、break-word（允许在长单词或长 URL 内部换行），例如：div{ word-wrap:break-word; }

（4）服务器端字体和@font-face 属性：

使用@font-face 属性获取服务器端字体的语法如下，可以使用的字体格式有：otf、ttf、eot

```
<style type="text/css">
        @font-face{
            font-family: webFont;
            src:url('字体所在路径/字体名称.字体格式');
        }
        h3{
            font-family: webFont;      /*使用服务器端字体*/
        }
</style>
```

使用@font-face 属性获取客户端本地字体的语法如下

```
<style type="text/css">
        @font-face{
            font-family: '字体名称';
            src:local('与上面一样的字体名称');
        }
        h3{
            font-family:与上面一样的字体名称;      /*使用客户端字体*/
        }
</style>
```

注意：字体可以设置的属性有：

font-family（字体类型）、font-size（字体大小）、font-style（字体样式）、font-variant（字体大小写）、font-weight（字体粗细）、font-stretch（字体横向的拉伸变形）、src（设置自定义字体的路径，只能在@font-face 规则中使用）