



.NET架构

主讲教师 张智
计算机学院

第4章 CSS3技术

4.1 基本概念

4.2 样式定义形式

4.3 CSS选择器

4.4 CSS常用属性

4.5 CSS盒子模型

4.6 综合示例

【附录1】 display 显示属性

【附录2】 position 定位属性

【附录3】 float 属性

4.1 基本概念



- CSS: 层叠样式表 (Cascading Style Sheets)
- 样式定义如何显示 HTML 元素
- 样式是为了解决内容与表现分离的问题
- 多个样式定义可层叠为一
- 样式通常存储在 CSS 文件中
- CSS3 是最新的 CSS 标准

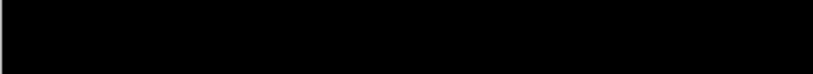








CSS单位（部分）

| 单位 | 描述 |
|----|--|
| % | 百分比 |
| px | 像素 (计算机屏幕上的一个点) |
| pt | 磅 (1 pt 等于 1/72 英寸) |
| em | 相对长度单位。相对于当前对象内文本的字体尺寸，例如，当前元素内字体大小是10px，那么在当前元素中2em=20px，0.5em=5px。浏览器一般默认字体大小为16px，因此，如果没有另外设置字体大小，那么1em=16px。em 是非常有用的单位，它可以自动适应用户所使用的字体大小。 |
| ex | 1ex=小写字母 x 的高度。(小写字母x的高度通常是字体尺寸的一半) |
| in | 英寸 (1in = 96px = 2.54cm) |
| cm | 厘米 |
| mm | 毫米 |

CSS颜色（部分）

| 单位 | 描述 |
|-----------------|---|
| 颜色名 | 颜色名称 (比如 red) |
| #rrggbb | 十六进制数 (比如红色: #ff0000, 简写为#f00) |
| rgb(x,x,x) | RGB 值 (比如红色: rgb(255,0,0)) |
| rgb(x%, x%, x%) | RGB 百分比值 (比如 rgb(100%,0%,0%), 100%相当于255) |
| rgba(r,g,b,a) | 其中a实现透明度(rgba(255,0,0,0.8), 0.8表示不透明度) |

颜色部分示例

| 颜色(Color) | 颜色十六进制(Color HEX) | 颜色RGB(Color RGB) | |
|---|-------------------|------------------|-------------------------|
|  | #000000 | rgb(0,0,0) | black |
|  | #FF0000 | rgb(255,0,0) | red |
|  | #00FF00 | rgb(0,255,0) | green |
|  | #0000FF | rgb(0,0,255) | blue |
|  | #FFFF00 | rgb(255,255,0) | yellow |
|  | #00FFFF | rgb(0,255,255) | cyan |
|  | #FF00FF | rgb(255,0,255) | fuchsia |
|  | #C0C0C0 | rgb(192,192,192) | silver |
|  | #FFFFFF | rgb(255,255,255) | white |

[【返回】](#)

4.2 样式定义形式

- 内联样式： 嵌在元素的`style`属性中定义 (局部)
- 内部样式： 在 `<style>` 标签中定义 (页面级)
- 外部样式： 在`css`文件中定义 (可被多个页面调用)

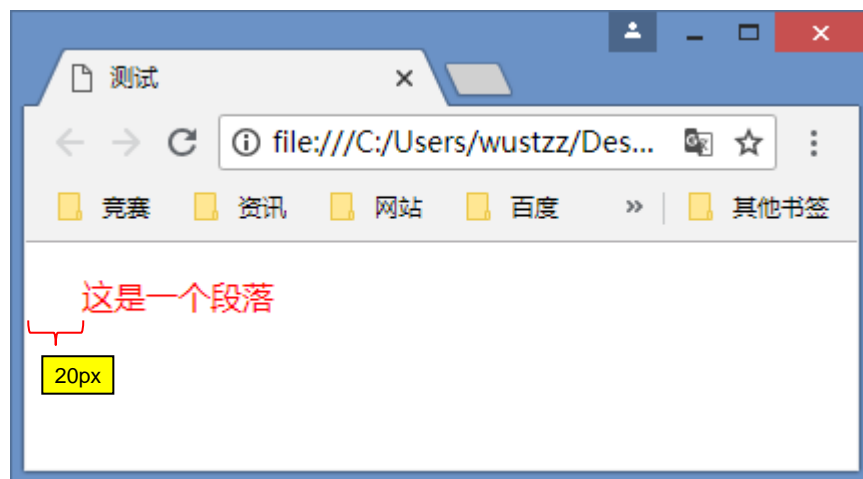
【[返回](#)】

1. 内联样式

元素的左外边距

```
<p style="color:red; margin-left:20px">这是一个段落</p>
```

注：不要在属性值与单位之间留有空格



[【返回】](#)

2. 内部样式

基本语法

<head>

<style>

<style>通常放在<head>标签中

h1 {

color: red; /*#ff0000, 简写#f00,或rgb(255,0,0)*/

font-size: 20px;

text-align: center;

}

body {

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

}

</style>

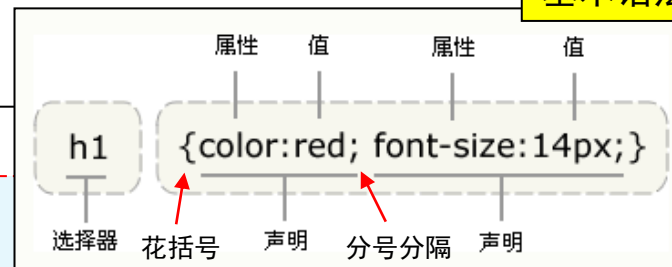
</head>

<body>

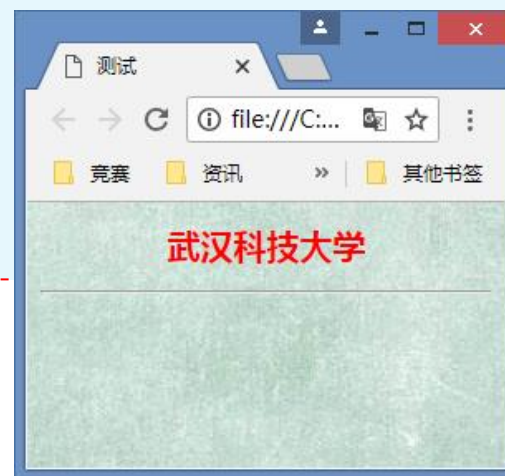
<h1>武汉科技大学</h1>

<hr>

</body>



/*CSS注释*/



[【返回】](#)

3. 外部样式

- 样式表定义在外部 .css 文件中
- 每个页面使用 <link> 标签链接到样式表

style.css文件

```
h1 {  
    color: blue;  
    font-size: 20px;  
    text-align: center;  
}  
body {  
    background-image: url("images/bg.jpg");  
}
```

css文件不能包含任何的 html 标签, 所以无需 <style> 标签

<head>

<link rel="stylesheet" href="style.css">



</head>

<body>

<h1>武汉科技大学</h1>

<hr>

</body>

关于多重样式优先级

■ 多重样式：

- 如果某些属性在不同的样式表中被同样的选择器定义，那么属性值将从根据优先级被继承过来。

■ 一般情况多重样式优先级：

- 内联样式 > 内部样式 > 外部样式 > 浏览器默认样式

多重样式示例

```
<head>  
  <link rel="stylesheet" href="style.css">
```

style.css文件

```
h3 {  
  color: red;  
  text-align: left;  
  font-size: 5px;  
}
```

```
<style>
```

提醒：一般将<link>放在<style>上面

```
  h3 {
```

```
    text-align: right;
```

```
    font-size: 20px;
```

```
  }
```

```
</style>
```

```
</head>
```

```
<body>
```

```
  <h3 style="font-size: 10px;">武汉科技大学</h3>
```

```
  <hr>
```

```
</body>
```

h3最终样式：
color: red;
text-align: right;
font-size: 10px;

注意：如果外部样式放在内部样式的后面(即<link>放在<style>后面)，则外部样式将覆盖内部样式。试一下

[【返回】](#)

4.3 CSS选择器

- 通配符选择器：选择器是星号 (*)
- 标签选择器：选择器是某个HTML元素标签名
- id选择器：选择器以 # 开头
- class选择器：选择器以 . 开头
- 属性选择器：[属性名]
- 后代选择器：用空格分隔或者用>连接
- 兄弟选择器：用+连接 用~连接
- 伪类和伪元素：用于向某些选择器添加特殊的效果

一般区分大小写

【[返回](#)】

1. 通配符选择器

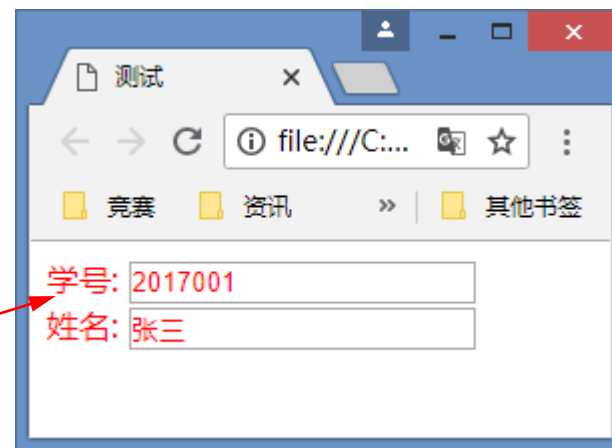
选择器是星号

该选择器可以与任何元素匹配

```
* {  
    color: red;  
}
```

```
<body>  
学号: <input type="text"> <br/>  
姓名: <input type="text"><br/>  
</body>
```

所有文字颜色都为red



[【返回】](#)

2. 标签选择器

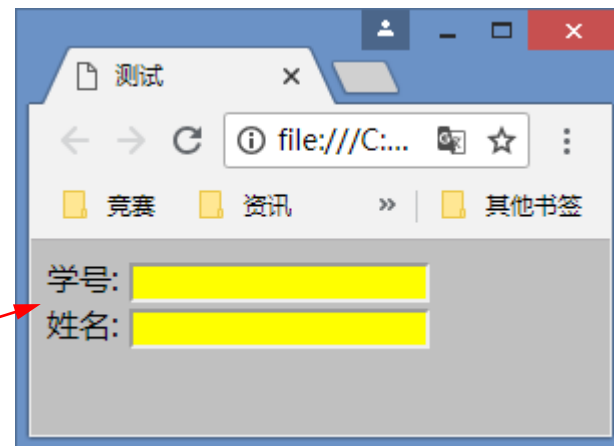
```
body { background-color:silver; }  
input { background-color:yellow; font-size: 12px; }  
p {color:gray;}
```

选择器是HTML标签

标签选择器将匹配该标签每一个实例

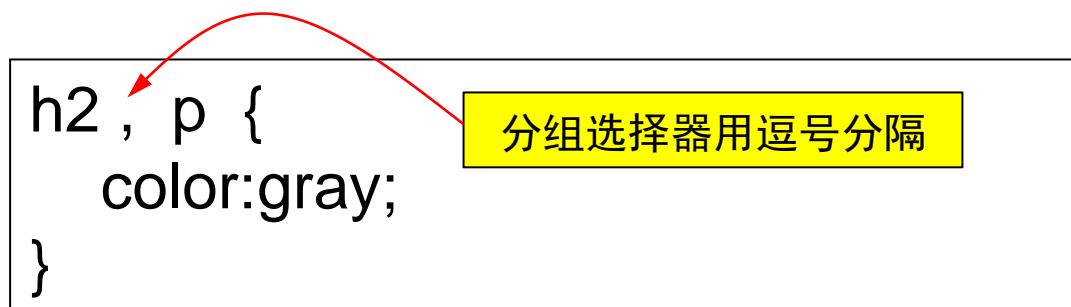
```
<body>  
学号: <input type="text"><br/>  
姓名: <input type="text"><br/>  
</body>
```

本例input都将使用同一样式



补充：选择器分组

- 假设希望 h2 和段落 p 文字都是灰色：



[【返回】](#)

3. id选择器

选择器以 # 开头

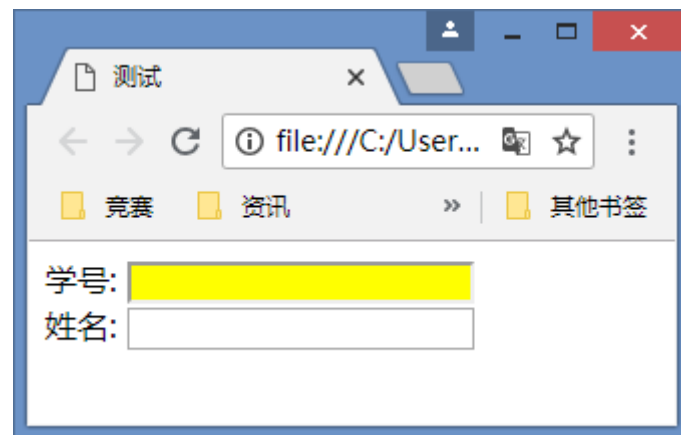
id 选择器为标有特定 id 的 HTML 元素指定样式

```
#xh {  
    background-color: yellow;  
}
```

id="xh"的元素背景色变成黄色

```
<body>  
    学号: <input type="text" id="xh"/> <br/>  
    姓名: <input type="text" id="name"/><br/>  
</body>
```

思考：如果"姓名"也用id="xh"结果如何？



[【返回】](#)

4. class选择器

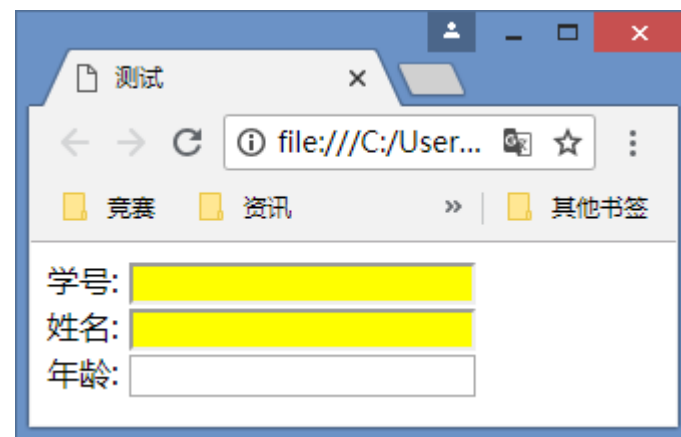
选择器以 . 开头

class选择器为标有特定class属性的元素指定样式

```
.bg {  
    background-color: yellow;  
}
```

class="bg"的元素背景色变成黄色

```
<body>  
学号: <input type="text" class="bg"> <br/>  
姓名: <input type="text" class="bg"><br/>  
年龄: <input type="text"><br/>  
</body>
```



补充1: id、class选择器前面可结合元素选择器

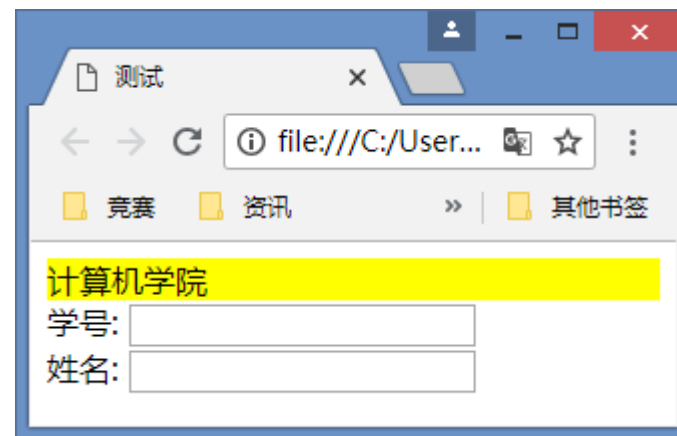
- 假设只想让div元素的背景色是黄色:

```
div.bg {  
    background-color: yellow;  
}
```

注意: 此处选择器之间没有空格

class="bg"的div元素背景色变成黄色

```
<body>  
<div class="bg">计算机学院</div>  
学号: <input type="text" class="bg"> <br/>  
姓名: <input type="text" class="bg"><br/>  
</body>
```



补充2：多class选择器（组合）

说明：如果属性不重复则样式叠加，
否则以最后面的class为准

■ 一个元素引用多个class样式

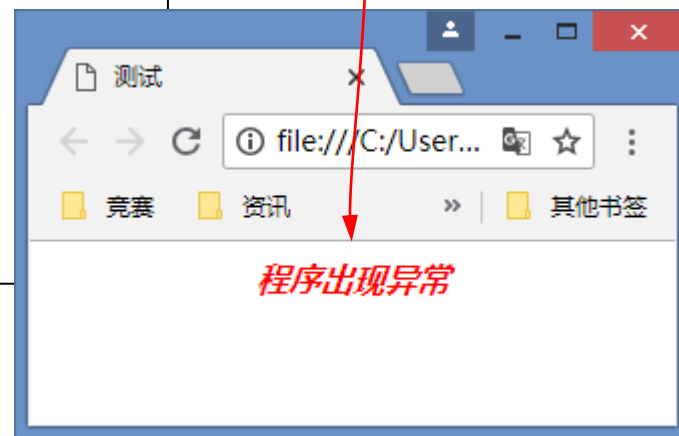
多个class用空格分隔,顺序无关紧要

```
<div class="center err">程序出现异常</div>
```

```
.center {  
    text-align: center;  
}
```

```
.err {  
    color: red;  
    font-weight: bold;  
    font-style: italic;  
}
```

本例将使用center和err两个class样式



补充2：多class选择器(续)

```
<div class="center err">程序出现异常</div>
```

```
.center {  
    text-align: center;  
}
```

```
.err {  
    color: red;  
    font-weight: bold;  
    font-style: italic;  
}
```

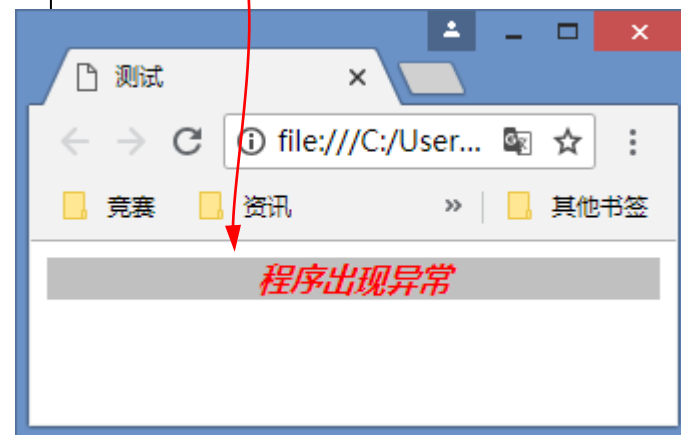
```
.center.err {  
    background-color: silver;  
}
```

```
.center.err.urgent {  
    font-size: 30px;  
}
```

本例将使用center、err以及.center.err这三个样式，第四个不会匹配

可以把多个类选择器连接在一起，中间没有空格，顺序不限

注意：多个类选择器链接在一起**仅**可以选择同时包含这些类的元素，如果类名列表中包含没有的类名，则匹配就会失败



5. 属性选择器

```
div[title] {
```

```
    background-color: yellow;
```

```
}
```

本例此处无空格

属性选择器可以根据元素的属性及属性值来选择元素

包含title属性的div元素背景色变为黄色

```
<body>
```

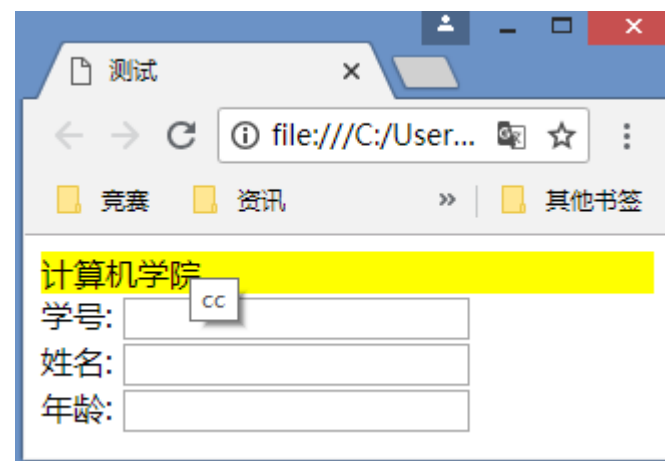
```
<div title="cc">计算机学院</div>
```

```
学号: <input type="text" title="学号"> <br/>
```

```
姓名: <input type="text" title="姓名"><br/>
```

```
年龄: <input type="text"><br/>
```

```
</body>
```



属性选择器（续）

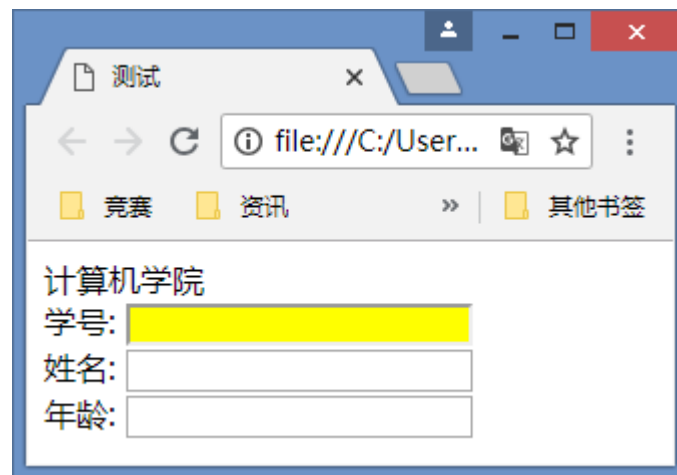
=属性值选择器：要求必须与属性值完全匹配(属性值引号可省略)

```
[title="学号"] {  
    background-color: yellow;  
}
```

title="学号"的元素背景色变为黄色

```
<body>  
<div title="cc">计算机学院</div>  
学号: <input type="text" title="学号"> <br/>  
姓名: <input type="text" title="姓名"><br/>  
年龄: <input type="text"><br/>  
</body>
```

备注：还可以根据多个属性进行选择，只需将属性选择器连接在一起即可，例如：[title][type="text"]



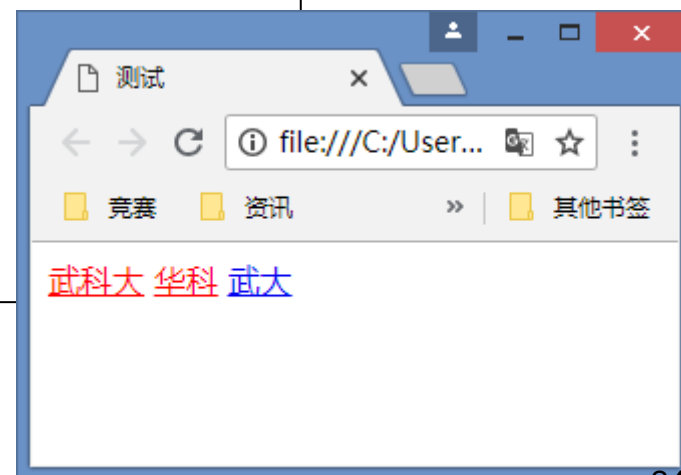
子串匹配属性选择器

| 类型 | 描述 |
|----------------------------|-----------------------------|
| <code>[abc^="def"]</code> | 选择 abc 属性值以 "def" 开头的元素 |
| <code>[abc\$="def"]</code> | 选择 abc 属性值以 "def" 结尾的元素 |
| <code>[abc*="def"]</code> | 选择 abc 属性值中包含子串 "def" 的所有元素 |

```
a[href*=ust] {  
  color: red;  
}
```

href属性含有"ust"子串的a标签字体为红色


```
<a href="http://www.wust.edu.cn/">武科大</a>  
<a href="http://www.hust.edu.cn/">华科</a>  
<a href="http://www.whu.edu.cn/">武大</a>
```



注意：选择器优先级

- 通用选择器 (*)
- 元素(类型)选择器
- 类选择器
- 属性选择器
- 伪类
- ID 选择器
- 内联样式

从低到高



!important 规则例外(不推荐使用): !important样式优先级最高
示例: `div{color:red !important;}`

[【返回】](#)

6. 后代选择器（包含选择器）

```
ul li a {  
  color:red;  
}
```

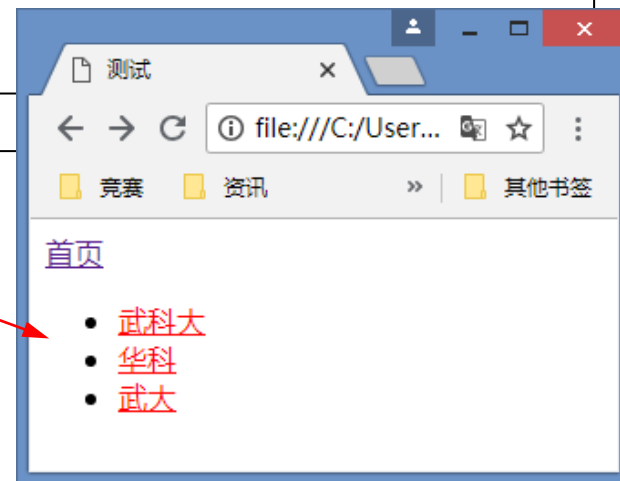
用空格分隔选择器

可以选择作为某元素后代的元素(可以跨代)

本例含义：作为 ul 后代的 li 的后代 a 元素字体为红色

试一试：ul a 或者 div a

```
<div id="nav">  
  <a href="#">首页</a>  
  <ul class="menu">  
    <li> <a href="http://www.wust.edu.cn/">武科大</a> </li>  
    <li> <a href="http://www.hust.edu.cn/">华科</a> </li>  
    <li> <a href="http://www.whu.edu.cn/">武大</a> </li>  
  </ul>  
</div>
```



子选择器（直接儿子）

用>连接选择器

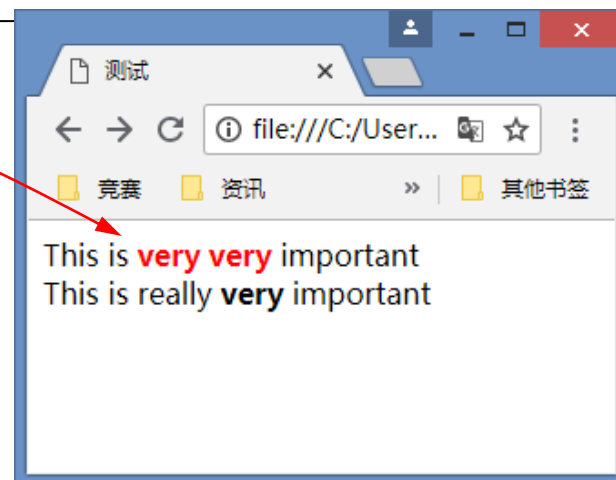
只能选择作为某元素直接子元素的元素

```
div > strong {  
  color:red;  
}
```

本例含义：div 所有直接儿子为 strong 元素字体为红色

```
<div>This is  
  <strong>very</strong>  
  <strong>very</strong> important  
</div>  
<div>This is  
  <span>really  
    <strong>very</strong>  
  </span> important  
</div>
```

这个没有匹配



思考：table.company td > p 含义

【返回】

7. 相邻兄弟选择器

用+连接选择器

选择紧跟在另一元素后的元素(第一个相邻的)，二者有相同父元素

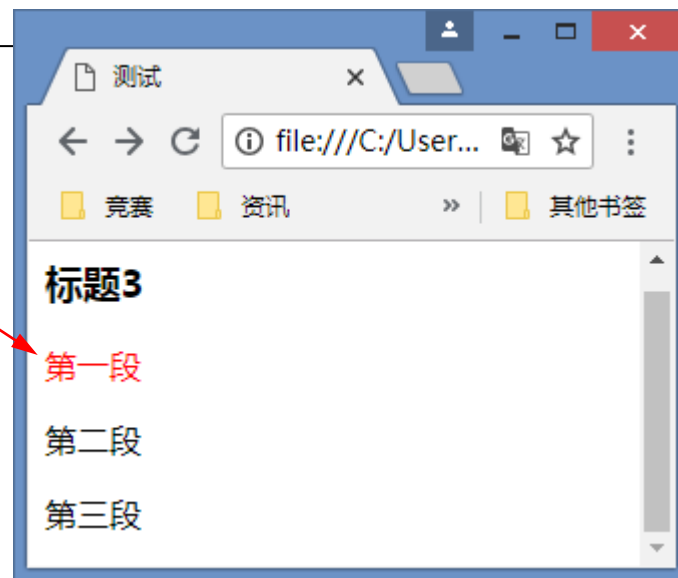
```
h3 + p {  
  color: red;  
}
```

含义：h3之后**第一个相邻**的p兄弟(h3 和 p 拥有共同的父元素)
注意：不是 h3 的所有兄弟 p，而是第一个兄弟

试一试：p+p 选择器

```
<div>  
  <h3>标题3</h3>  
  <p>第一段</p>  
  <p>第二段</p>  
  <p>第三段</p>  
</div>
```

其他两个p没有匹配



同级兄弟选择器

用~连接选择器

选择同级的元素(后面相邻的兄弟), 二者有相同父元素

```
h3 ~ p {  
  color: red;  
}
```

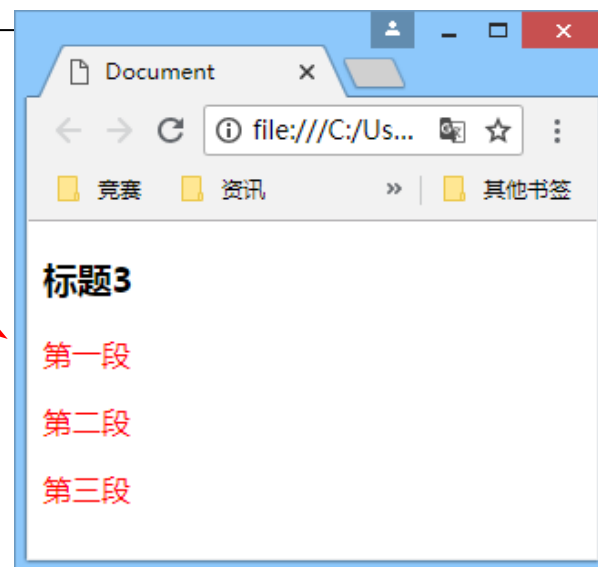
含义: 与h3同级的所有p兄弟(h3 和 p 拥有共同的父元素)

所有兄弟p都匹配

试一试: p~p 选择器

```
<div>  
  <h3>标题3</h3>  
  <p>第一段</p>  
  <p>第二段</p>  
  <p>第三段</p>  
</div>
```

思考: 本例如果将第一个p移到h3上面结果如何?



[【返回】](#)


8. 伪类和伪元素

| 选择器 | 例子 | 例子描述 | CSS |
|--|----------------|------------------------------------|-----|
| <u>:link</u> | a:link | 选择所有未被访问的链接。 | 1 |
| <u>:visited</u> | a:visited | 选择所有已被访问的链接。 | 1 |
| <u>:active</u> | a:active | 选择活动链接。 | 1 |
| <u>:hover</u> | a:hover | 选择鼠标指针位于其上的链接。 | 1 |
| <u>:focus</u> | input:focus | 选择获得焦点的 input 元素。 | 2 |
| <u>:first-letter</u> | p:first-letter | 选择每个 <p> 元素的首字母。 | 1 |
| <u>:first-line</u> | p:first-line | 选择每个 <p> 元素的首行。 | 1 |
| <u>:first-child</u> | p:first-child | 选择属于父元素的第一个子元素的每个 <p> 元素。 | 2 |
| <u>:before</u> | p:before | 在每个 <p> 元素的内容之前插入内容。 | 2 |
| <u>:after</u> | p:after | 在每个 <p> 元素的内容之后插入内容。 | 2 |
| <u>:lang(language)</u> | p:lang(it) | 选择带有以 "it" 开头的 lang 属性值的每个 <p> 元素。 | 2 |


| 选择器 | 例子 | 例子描述 | CSS |
|------------------------------------|-----------------------|-------------------------------|-----|
| <u>:first-of-type</u> | p:first-of-type | 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。 | 3 |
| <u>:last-of-type</u> | p:last-of-type | 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。 | 3 |
| <u>:only-of-type</u> | p:only-of-type | 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。 | 3 |
| <u>:only-child</u> | p:only-child | 选择属于其父元素的唯一子元素的每个 <p> 元素。 | 3 |
| <u>:nth-child(<i>n</i>)</u> | p:nth-child(2) | 选择属于其父元素的第二个子元素的每个 <p> 元素。 | 3 |
| <u>:nth-last-child(<i>n</i>)</u> | p:nth-last-child(2) | 同上，从最后一个子元素开始计数。 | 3 |
| <u>:nth-of-type(<i>n</i>)</u> | p:nth-of-type(2) | 选择属于其父元素第二个 <p> 元素的每个 <p> 元素。 | 3 |
| <u>:nth-last-of-type(<i>n</i>)</u> | p:nth-last-of-type(2) | 同上，但是从最后一个子元素开始计数。 | 3 |
| <u>:last-child</u> | p:last-child | 选择属于其父元素最后一个子元素每个 <p> 元素。 | 3 |
| <u>:root</u> | :root | 选择文档的根元素。 | 3 |
| <u>:empty</u> | p:empty | 选择没有子元素的每个 <p> 元素（包括文本节点）。 | 3 |
| <u>:target</u> | #news:target | 选择当前活动的 #news 元素。 | 3 |
| <u>:enabled</u> | input:enabled | 选择每个启用的 <input> 元素。 | 3 |
| <u>:disabled</u> | input:disabled | 选择每个禁用的 <input> 元素 | 3 |
| <u>:checked</u> | input:checked | 选择每个被选中的 <input> 元素。 | 3 |
| <u>:not(<i>selector</i>)</u> | :not(p) | 选择非 <p> 元素的每个元素。 | 3 |
| <u>::selection</u> | ::selection | 选择被用户选取的元素部分。 | 3 |

示例1：

```
<a href="http://www.wust.edu.cn" target="_blank">武科大</a>
```

```
a:link { /* 未访问的链接 */   
    color: red;  
}
```

```
a:visited { /* 已访问的链接 */  
    color: grey;  
}
```

```
a:hover { /* 鼠标移动到链接上 */   
    color: purple;  
}
```

```
a:active { /* 选定的链接(鼠标按下时) */  
    color: blue;  
}
```

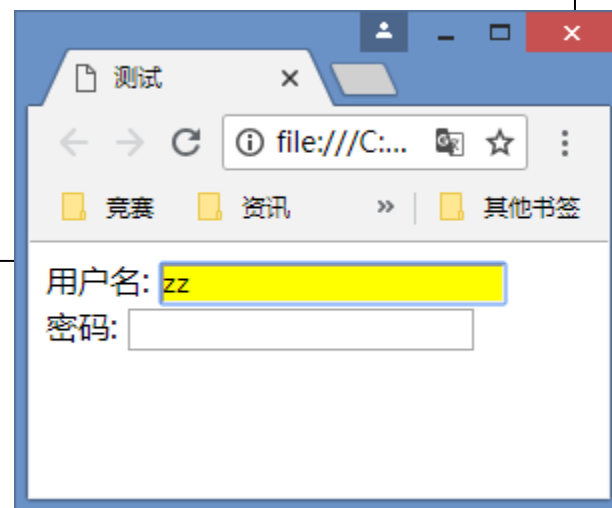
注意：如果谷歌浏览器出现a:link无效
解决：清理一下浏览器缓存即可

示例2: :focus向获得焦点的元素添加样式

```
input:focus {  
    background-color: yellow;  
}
```

功能：当input元素获得焦点时
背景色变为黄色

```
<form>  
    用户名:<input type="text" name="username" />  
    <br />  
    密码: <input type="text" name="psd" />  
</form>
```



示例3: :first-child选择元素的第一个子元素

```
p:first-child {  
  color: red;  
}
```

含义：作为第一个子元素的 p 元素
注意：不是 p 元素的第一个子元素

试一试： div:first-child 选择器

```
<body>
```

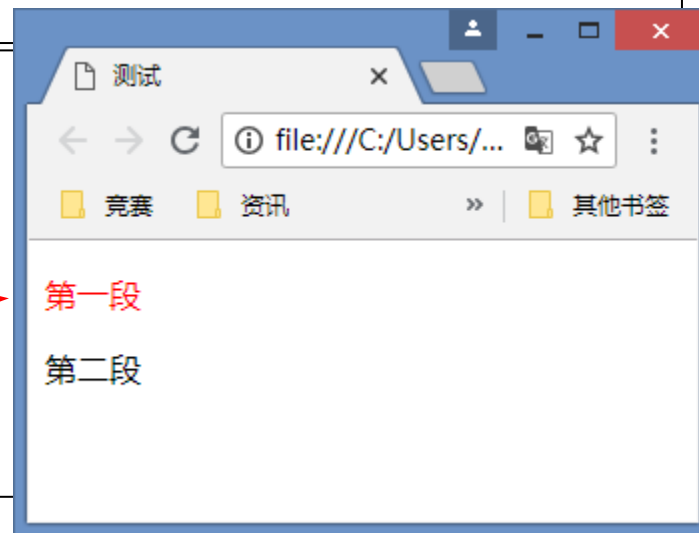
```
  <div>
```

```
    <p>第一段</p>
```

```
    <p>第二段</p>
```

```
  </div>
```

```
</body>
```



:first-child 理解：本例

首先，作为第一个子元素的有 div(body的第一个)和第一个 p(div的第一个)
然后，p:first-child 选择的是 p 的那个

:first-child示例： 结果如何？

```
p > i:first-child {  
    color:red;  
}
```

```
<body>  
    I am a <i>strong</i> man.  
    <p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>  
    <p>I am a <i>strong</i> man. I am a <i>strong</i> man.</p>  
</body>
```

示例4: :before在元素之前插入新内容

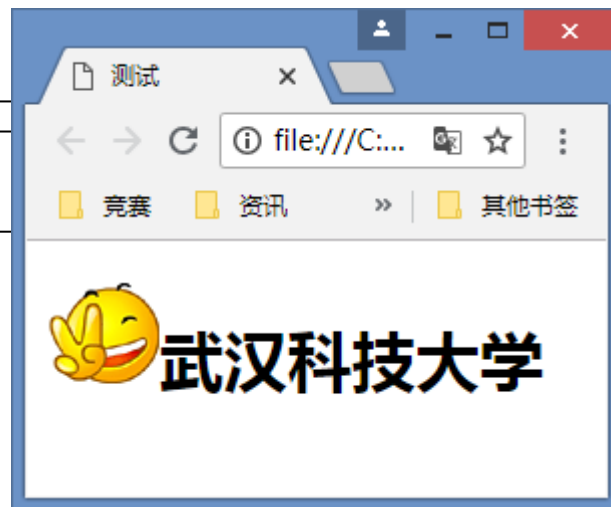
含义：在 h1 元素之前添加内容

```
h1:before {  
    content: url("images/1.gif");  
}
```

<h1>武汉大学</h1>



注：CSS3中推荐写法是 ::before、::after

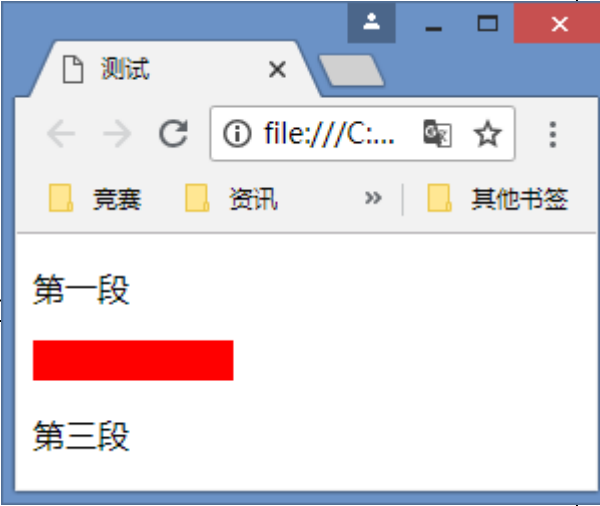


示例5: :empty匹配没有子元素的每个元素

含义：匹配没有内容的 p 元素

```
p:empty {  
    width: 100px;  
    height: 20px;  
    background-color: red;  
}
```

```
<p>第一段</p>  
<p></p>  
<p>第三段</p>
```



测试

file:///C:...

竞赛 资讯 其他书签

第一段

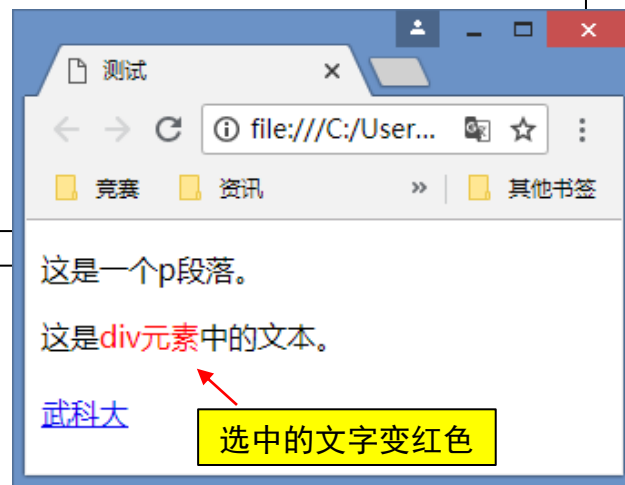
第三段

示例6: ::selection匹配被用户选取的部分

```
::selection {  
    color: #ff0000;  
}
```

效果：当选择文字时文字变成红色

```
<p>这是一个p段落。</p>  
<div>这是div元素中的文本。</div>  
<br>  
<a href="http://www.wust.edu.cn" target="_blank">武科大</a>
```



【返回】

4.4 CSS常用属性

- 背景属性
- 字体属性
- 文本属性
- 列表属性
- 尺寸属性

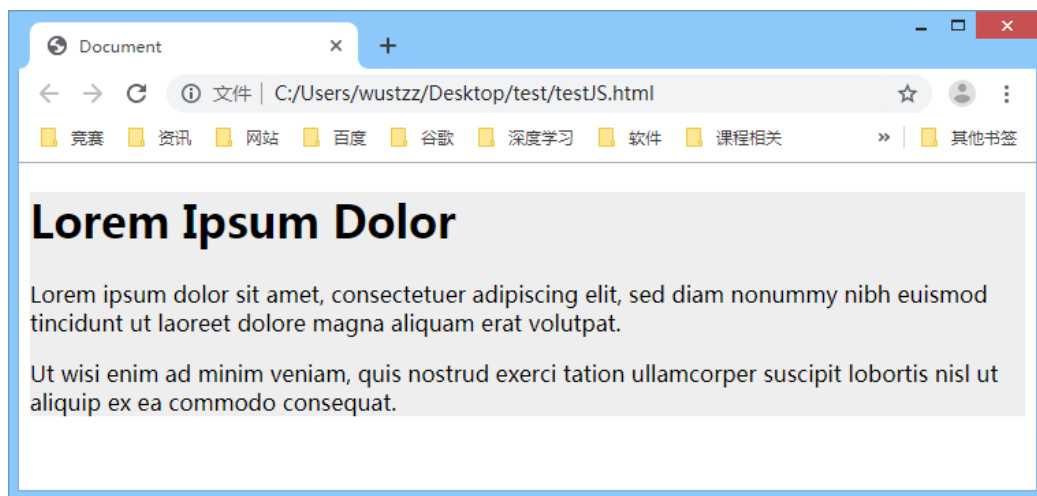
【返回】

1. 背景属性

| 属性 | 描述 |
|------------------------------|--|
| background | 简写属性 ，作用是将背景属性设置在一个声明中 background: color img_url repeat attachment position / size |
| background-attachment | 背景图像是否固定或者随着页面的其余部分滚动 |
| background-color | 设置元素的背景颜色 |
| background-image | 把图像设置为背景 |
| background-position | 设置背景图像的起始位置 |
| background-repeat | 设置背景图像是否及如何重复 |
| background-clip | 规定背景的绘制区(剪裁) |
| background-origin | 规定背景图片的定位区域 |
| background-size | 规定背景图片的尺寸 |

示例1：div添加背景色

```
#example1 {  
    background-color:#eee;  
}
```



示例1：div内容

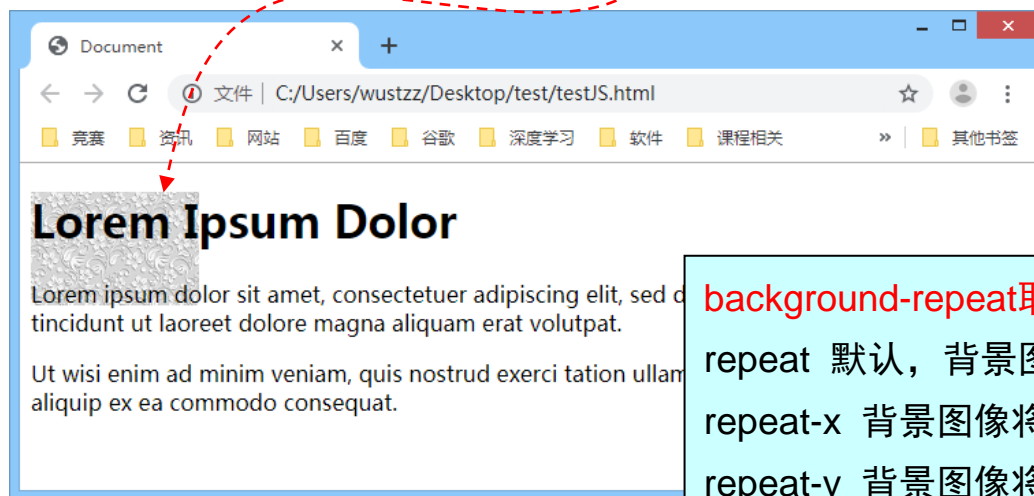
```
<div id="example1">  
  <h1>Lorem Ipsum Dolor</h1>  
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy  
nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>  
  <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit  
lobortis nisl ut aliquip ex ea commodo consequat.</p>  
</div>
```

示例2: div添加背景图像

```
#example1 {  
    background-image: url(images/布纹.jpg);  
    background-repeat: no-repeat;  
}
```

简写形式

background: url(images/布纹.jpg) no-repeat;



background-repeat取值:

repeat 默认, 背景图像将在垂直方向和水平方向重复
repeat-x 背景图像将在水平方向重复
repeat-y 背景图像将在垂直方向重复
no-repeat 背景图像将仅显示一次

示例3：固定背景图像

在页面中多加些内容，滚动窗口时观察背景效果

```
body {  
    background-image: url(images/布纹.jpg) repeat-x;  
    background-attachment: fixed;  
}
```

背景图像滚动情况

scroll：默认值。背景图像会随着页面其余部分的滚动而移动。
fixed：当页面的其余部分滚动时，背景图像不会移动。

简写形式：

```
background: url(images/布纹.jpg) repeat-x fixed ;
```

[【返回】](#)

2. 字体属性

| 属性 | 描述 |
|-------------------------|----------------------------------|
| font | 简写属性。作用是把所有针对字体的属性设置在一个声明中。 |
| font-family | 设置字体系列。 |
| font-size | 设置字体的尺寸。 |
| font-size-adjust | 当首选字体不可用时，对替换字体进行智能缩放。 |
| font-stretch | 对字体进行水平拉伸。 |
| font-style | 设置字体风格。 |
| font-variant | 以小型大写字体或者正常字体显示文本。 |
| font-weight | 设置字体的粗细（normal bold 100-900） |

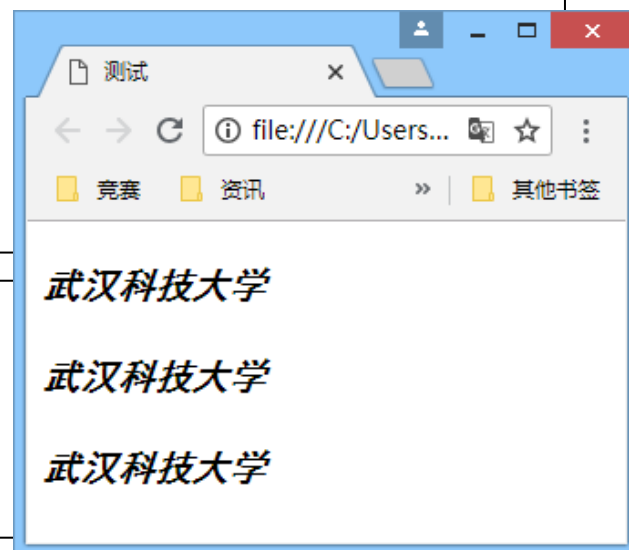
400 等同于 normal，而 700 等同于 bold

示例：

```
p {  
    font-size: 20px;  
    font-family: 黑体,arial,sans-serif;  
    font-style: italic;  
    font-weight: bold;  
    line-height: 30px; /*行高*/  
}
```

如果浏览器不支持第一个字体，
则会尝试下一个

```
<p>武汉科技大学</p>  
<p>武汉科技大学</p>  
<p>武汉科技大学</p>
```



[【返回】](#)

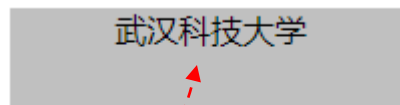
3. 文本属性

| 属性 | 描述 |
|-----------------|--|
| color | 设置文本颜色 |
| direction | 设置文本方向。 |
| line-height | 设置行高。 |
| letter-spacing | 设置字符间距。 |
| text-align | 对齐元素中的文本。left right center justify(两端对齐) |
| vertical-align | 设置元素的垂直对齐方式(top text-top middle bottom text-bottom等) |
| text-decoration | 向文本添加修饰。(a标签常用) |
| text-indent | 首行缩进。 |
| text-transform | 控制元素中的字母(字母大小写)。 |
| unicode-bidi | 设置文本方向。 |
| white-space | 设置元素中空白的处理方式(如空格合并)。 |
| word-spacing | 设置字间距。 |

| 属性 | 描述 |
|---------------------|---|
| hanging-punctuation | 规定标点字符是否位于线框之外。 |
| punctuation-trim | 规定是否对标点字符进行修剪。 |
| text-align-last | 设置如何对齐最后一行或紧挨着强制换行符之前的行。 |
| text-emphasis | 向元素的文本应用重点标记以及重点标记的前景色。 |
| text-justify | 规定当 text-align 设置为 "justify" 时所使用的对齐方法。 |
| text-outline | 规定文本的轮廓。 |
| text-overflow | 规定当文本溢出包含元素时发生的事情。 |
| text-shadow | 向文本添加阴影。 |
| text-wrap | 规定文本的换行规则。 |
| word-break | 规定非中日韩文本的换行规则。 |
| word-wrap | 允许对长的不可分割的单词进行分割并换行到下一行。 |

示例1：div文字水平居中

```
div.center {  
    background-color: silver;  
    width:200px;  
    height: 50px;  
    text-align: center;  
}
```



```
<div class="center">武汉科技大学</div>
```

思考题：将div换成span能居中么？

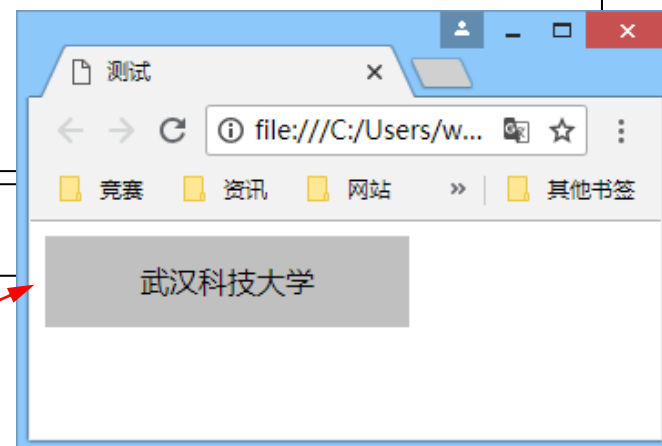
(需要添加 display:block; 原因：行内元素宽高是自动值，修改无效，详见3.4.6节)

示例2: div文字水平垂直居中

```
div.center {  
    background-color: silver;  
    width:200px;  
    height: 50px;  
    text-align: center;  
    line-height: 50px;  
}
```

设置行高与div等高，可以达到文字垂直居中效果

```
<div class="center">武汉科技大学</div>
```



示例3：图片垂直居中

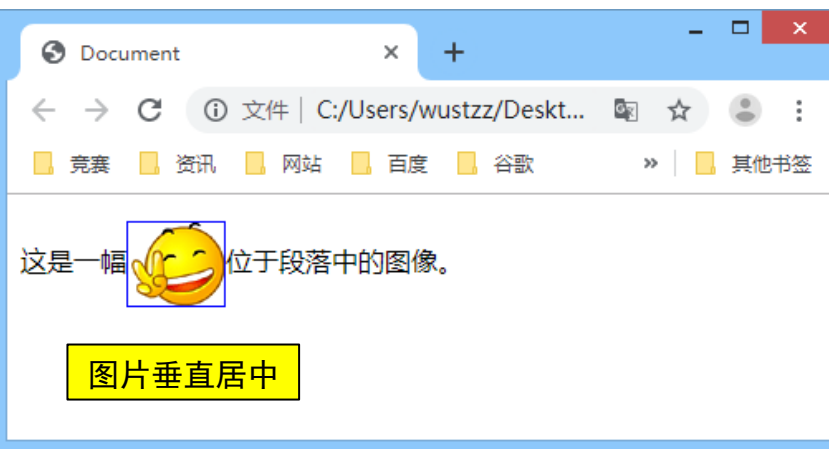
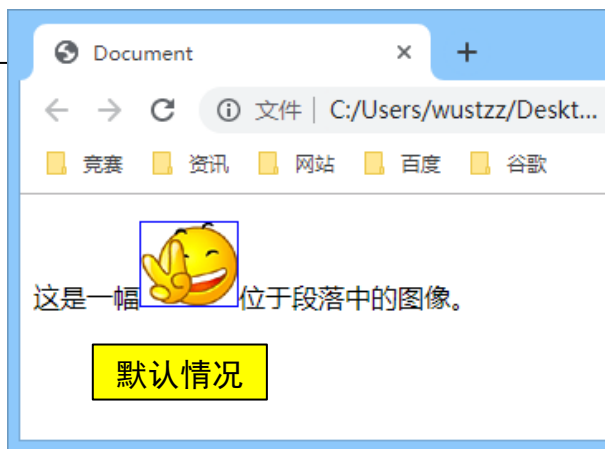
```
img {  
    border: 1px solid blue;  
    vertical-align: middle;  
}
```

备注：vertical-align用来指定行内元素或表格单元格元素的垂直对齐方式，对于块级元素不起作用

<p>

这是一幅位于段落中的图像。

</p>

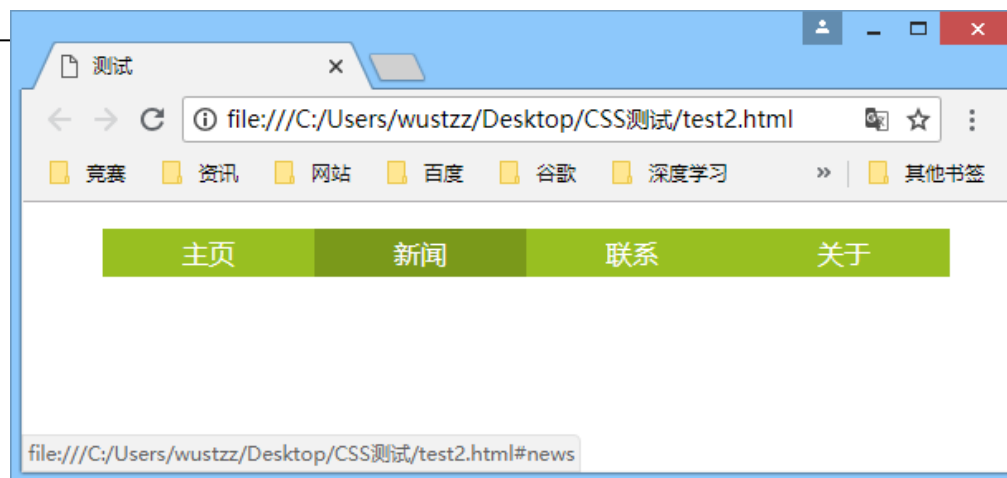


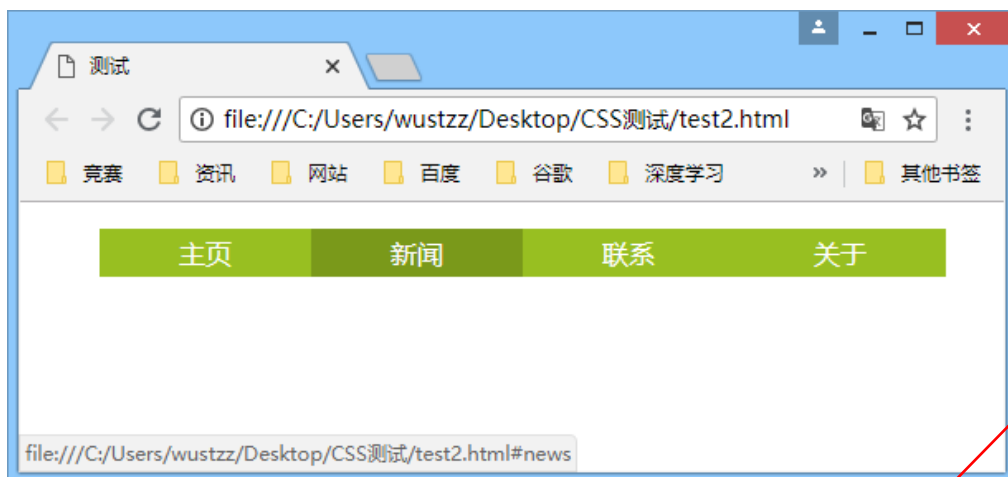
示例4: text-decoration用法 (导航栏)

```
<ul>
  <li><a href="#home">主页</a></li>
  <li><a href="#news">新闻</a></li>
  <li><a href="#contact">联系</a></li>
  <li><a href="#about">关于</a></li>
</ul>
```

text-decoration 取值:

- none:无装饰
- underline:下划线
- overline:上划线
- line-through:删除线





a元素默认为行内元素，改成块元素

```
ul {  
    list-style-type: none;  
}  
li {  
    float: left;  
}
```

有关float浮动效果
详见"定位和浮动"节

```
a:link,a:visited {  
    display: block;  
    width: 120px;  
    text-align: center;  
    padding: 4px; /*内边距*/  
    color: #fff;  
    background-color: #98bf21;  
    text-decoration: none;  
}
```

```
a:hover,a:active {  
    background-color: #7A991A;  
}
```

去掉a元素默认的修饰

[【返回】](#)

4. 列表属性

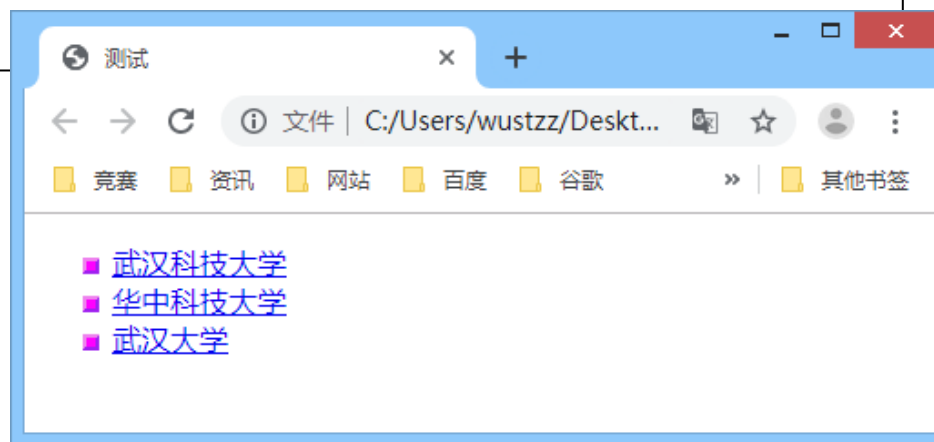
一般是ul标签使用

| 属性 | 描述 |
|----------------------------------|---------------------------|
| <code>list-style</code> | 简写属性。用于把所有用于列表的属性设置于一个声明中 |
| <code>list-style-image</code> | 将图象设置为列表项标志。 |
| <code>list-style-position</code> | 设置列表中列表项标志的位置。 |
| <code>list-style-type</code> | 设置列表项标志的类型。 |

示例：list-style-image样式

```
<ul style="list-style-image:url(images/sqpurple.gif);">  
  <li><a href="http://www.wust.edu.cn">武汉科技大学</a></li>  
  <li><a href="http://www.hust.edu.cn">华中科技大学</a></li>  
  <li><a href="http://www.whu.edu.cn">武汉大学</a></li>  
</ul>
```

用小图片(9×9)



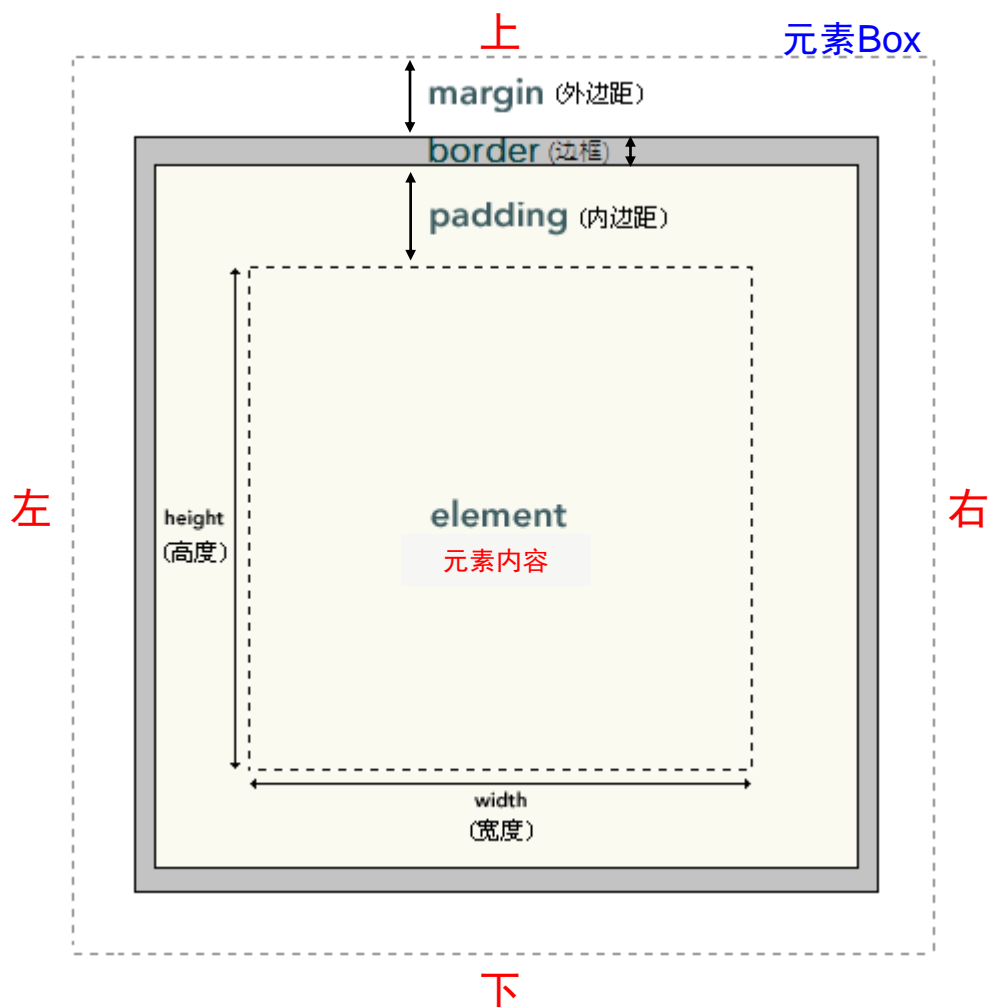
【[返回](#)】

5. 尺寸属性

| 属性 | 描述 |
|-------------------------|-----------------------------|
| <code>height</code> | 设置元素的高度。 |
| <code>max-height</code> | 设置元素的最大高度(元素可以比指定值矮，但不能比其高) |
| <code>max-width</code> | 设置元素的最大宽度(元素可以比指定值窄，但不能比其宽) |
| <code>min-height</code> | 设置元素的最小高度。 |
| <code>min-width</code> | 设置元素的最小宽度。 |
| <code>width</code> | 设置元素的宽度。 |

[【返回】](#)

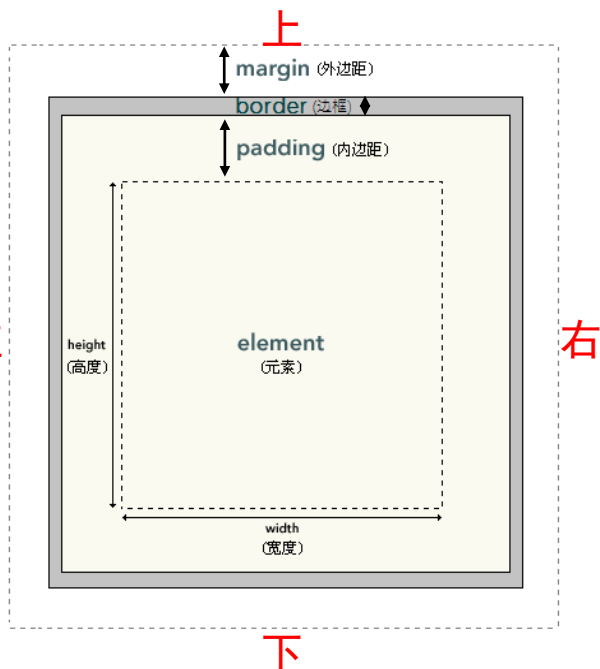
4.5 CSS盒子模型



- 内边距、边框和外边距都是可选的，默认值是零
- 外边距margin默认是透明的，因此不会遮挡其它任何元素
- 内边距padding默认也是透明的
- 背景应用于由内容和内边距、边框组成的区域(不含margin)
- width 和 height 指的是内容区域的宽度和高度
- 增加内边距、边框和外边距不会影响内容区域的尺寸，但是会增加元素盒子的总尺寸（下页有说明）

备注：Box外面是与position相关的属性如：
left、right、top、bottom值

元素的总宽度和高度计算公式



元素的总宽度

= 宽度 width
+ 左内边距 padding-left
+ 右内边距 padding-right
+ 左边框 border-left
+ 右边框 border-right
+ 左外边距 margin-left
+ 右外边距 margin-right

元素的总高度

= 高度 height
+ 上内边距 padding-top
+ 下内边距 padding-bottom
+ 上边框 border-top
+ 下边框 border-bottom
+ 上外边距 margin-top
+ 下外边距 margin-bottom

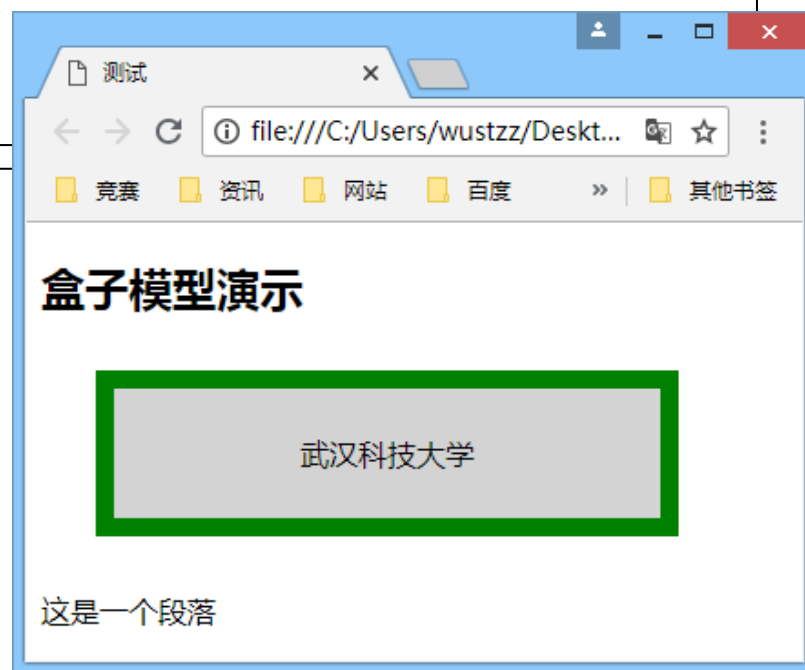
盒子模型示例：要会计算总宽度或高度

```
<h2>盒子模型演示</h2>
```

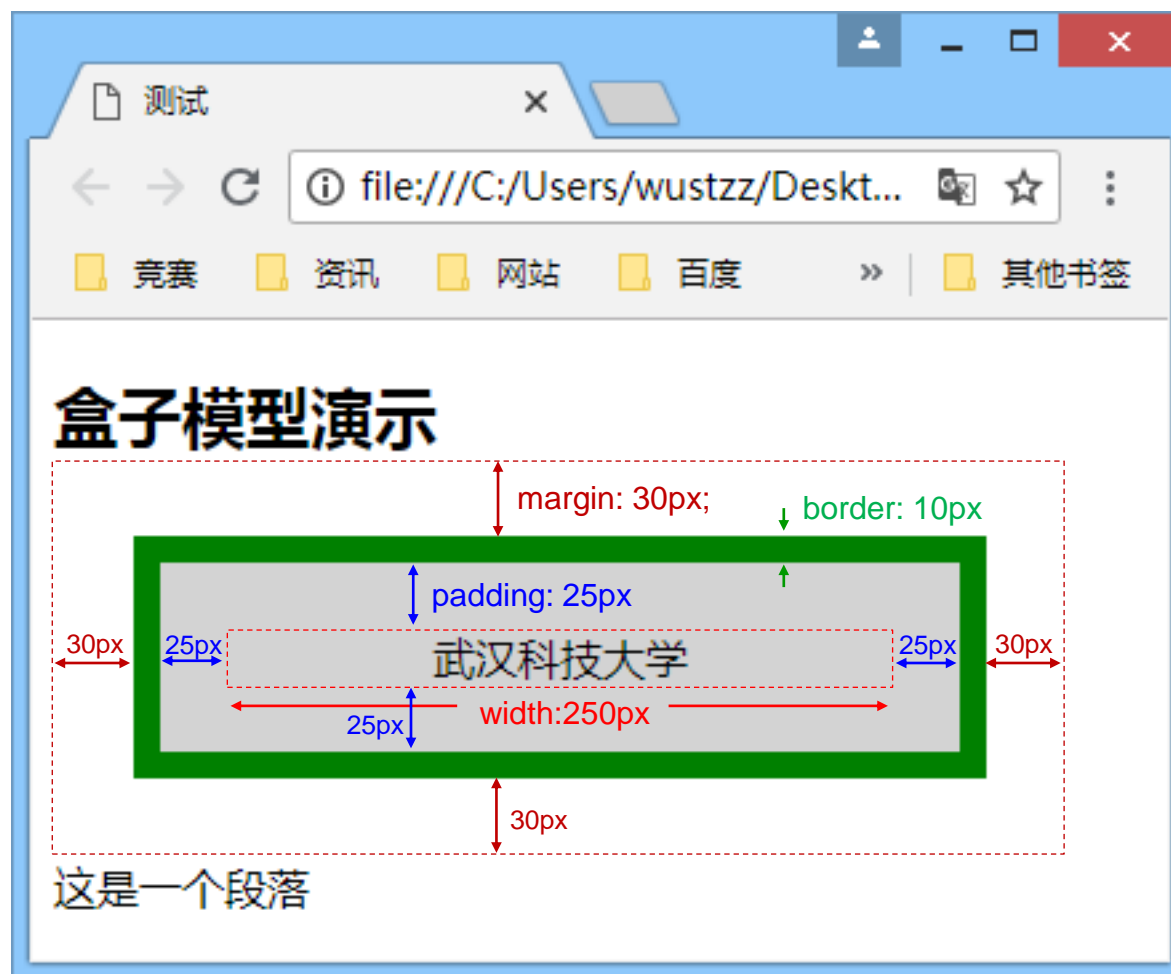
```
<div>武汉科技大学计算机学院</div>
```

```
<p>这是一个段落</p>
```

```
div {  
    background-color: lightgrey;  
    text-align: center;  
    width: 250px;  
    border: 10px solid green;  
    padding: 25px;  
    margin: 30px;  
}
```



div总宽度=250+25+25+10+10+30+30=380px



1. 边框属性

| 属性 | 描述 |
|----------------------------------|-------------------|
| <code>border</code> | 在一个声明中设置所有的边框属性。 |
| <code>border-bottom</code> | 在一个声明中设置所有的下边框属性。 |
| <code>border-bottom-color</code> | 设置下边框的颜色。 |
| <code>border-bottom-style</code> | 设置下边框的样式。 |
| <code>border-bottom-width</code> | 设置下边框的宽度。 |
| <code>border-color</code> | 设置四条边框的颜色。 |
| <code>border-left</code> | 在一个声明中设置所有的左边框属性。 |
| <code>border-left-color</code> | 设置左边框的颜色。 |
| <code>border-left-style</code> | 设置左边框的样式。 |
| <code>border-left-width</code> | 设置左边框的宽度。 |

边框属性（续）

| 属性 | 描述 |
|---------------------------------|-------------------|
| <code>border-right</code> | 在一个声明中设置所有的右边框属性。 |
| <code>border-right-color</code> | 设置右边框的颜色。 |
| <code>border-right-style</code> | 设置右边框的样式。 |
| <code>border-right-width</code> | 设置右边框的宽度。 |
| <code>border-style</code> | 设置四条边框的样式。 |
| <code>border-top</code> | 在一个声明中设置所有的上边框属性。 |
| <code>border-top-color</code> | 设置上边框的颜色。 |
| <code>border-top-style</code> | 设置上边框的样式。 |
| <code>border-top-width</code> | 设置上边框的宽度。 |
| <code>border-width</code> | 设置四条边框的宽度。 |

| 属性 | 描述 |
|---|--|
| <code>border-bottom-left-radius</code> | 定义边框左下角的形状。 |
| <code>border-bottom-right-radius</code> | 定义边框右下角的形状。 |
| <code>border-image</code> | 简写属性，设置所有 <code>border-image-*</code> 属性。 |
| <code>border-image-outset</code> | 规定边框图像区域超出边框的量。 |
| <code>border-image-repeat</code> | 图像边框平铺(repeated)、铺满(rounded)或拉伸(stretched)。 |
| <code>border-image-slice</code> | 规定图像边框的向内偏移。 |
| <code>border-image-source</code> | 规定用作边框的图片。 |
| <code>border-image-width</code> | 规定图片边框的宽度。 |
| <code>border-radius</code> | 简写属性，设置所有四个 <code>border-*-radius</code> 属性。 |
| <code>border-top-left-radius</code> | 定义边框左上角的形状。 |
| <code>border-top-right-radius</code> | 定义边框右下角的形状。 |
| <code>box-decoration-break</code> | 定义盒模型断开效果 |
| <code>box-shadow</code> | 向方框添加一个或多个阴影。 |

border的简写示例：

```
div {  
    background-color: lightgrey;  
    text-align: center;  
    width: 250px;  
    border: 10px solid green; ← 简写  
    padding: 25px;  
    margin: 30px;  
}
```

border: 10px solid green; 的简写属性包括：

- border-width: 10px;
- border-style: solid
- border-color: green

border-style取值及其效果

none: 默认无边框

dotted: 定义一个点线边框

dashed: 定义一个虚线边框

solid: 定义实线边框

double: 定义两个边框。两个边框的宽度和 border-width 的值相同

groove: 定义3D沟槽边框。效果取决于边框的颜色值

ridge: 定义3D脊边框。效果取决于边框的颜色值

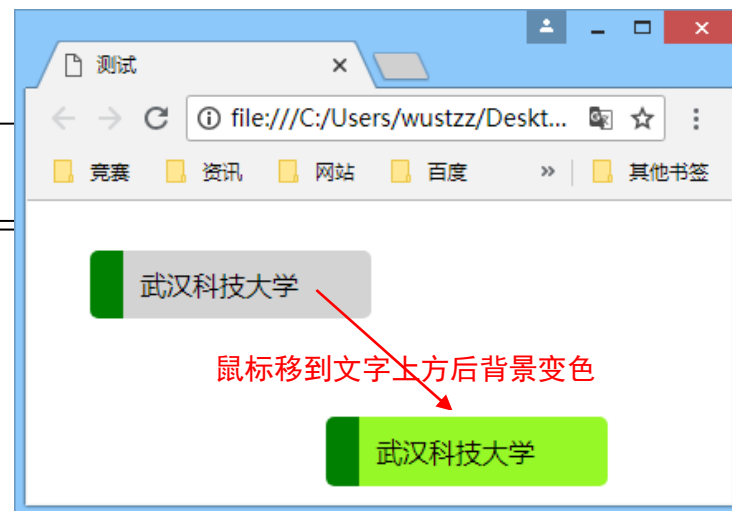
inset: 定义一个3D的嵌入边框。效果取决于边框的颜色值

outset: 定义一个3D突出边框。效果取决于边框的颜色值

示例1：

```
<div>武汉科技大学</div>
```

```
div {  
    background-color: lightgrey;  
    width: 130px;  
    border-left: 20px solid green; /*左边框*/  
    border-radius: 5px; /*圆角*/  
    padding: 10px;  
    margin: 30px;  
}  
  
div: hover {  
    background-color: rgb(151, 248, 39);  
    cursor: pointer;  
}
```



示例2: box-shadow用法 (图像卡片)

```
<div class="card">  
    
  <div class="city">  
    <p>湖北 武汉</p>  
  </div>  
</div>
```



初始:

图片不透明度0.8 (透0.2), 阴影为浅灰色

鼠标移到上方:

图片不透明度为1 (恢复正常), 阴影为深灰色

```
img {  
  width: 100%;  
  opacity: 0.8; /*不透明度0.8*/  
}  
img:hover {  
  opacity: 1; /*完全不透明*/  
}  
div.card {  
  width: 200px;  
  box-shadow: 5px 10px 5px silver;  
  text-align: center;  
}  
div.card:hover {  
  box-shadow: 5px 10px 5px gray; /*阴影颜色变深*/  
  cursor: pointer;  
}  
div.city {  
  padding: 10px;  
}
```

说明：百分数值是相对于其父元素的 width 计算的

box-shadow基本语法:

box-shadow:h-shadow v-shadow blur color ;

h-shadow: 水平阴影长度

v-shadow: 垂直阴影长度

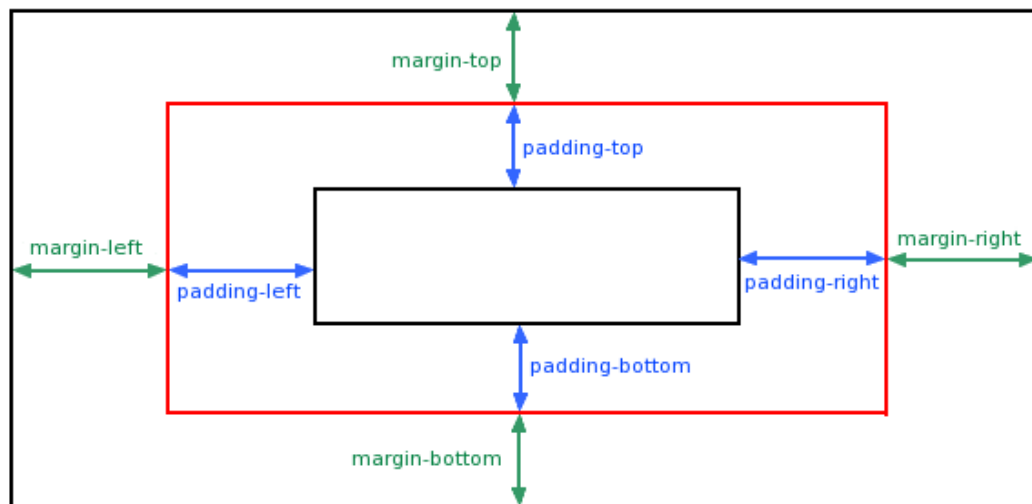
blur: 模糊距离

color: 阴影的颜色

2. padding内边距属性

注：padding值不能为负值

| 属性 | 描述 |
|----------------|------------------------|
| padding | 简写属性。在一个声明中设置元素的内边距属性。 |
| padding-bottom | 设置元素的下内边距。 |
| padding-left | 设置元素的左内边距。 |
| padding-right | 设置元素的右内边距。 |
| padding-top | 设置元素的上内边距。 |



padding的简写:

```
div {  
    background-color: lightgrey;  
    text-align: center;  
    width: 100px;  
    border: 1px solid black;  
    padding: 25px 50px;  
}
```

padding-top: 25px;
padding-right: 50px;
padding-bottom: 25px;
padding-left: 50px;

简写

padding: 25px 50px 25px 50px;
“上右下左”的顺序

如果上=下, 左=右则
进一步简写

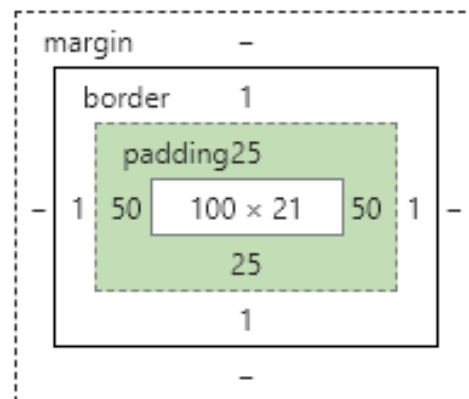
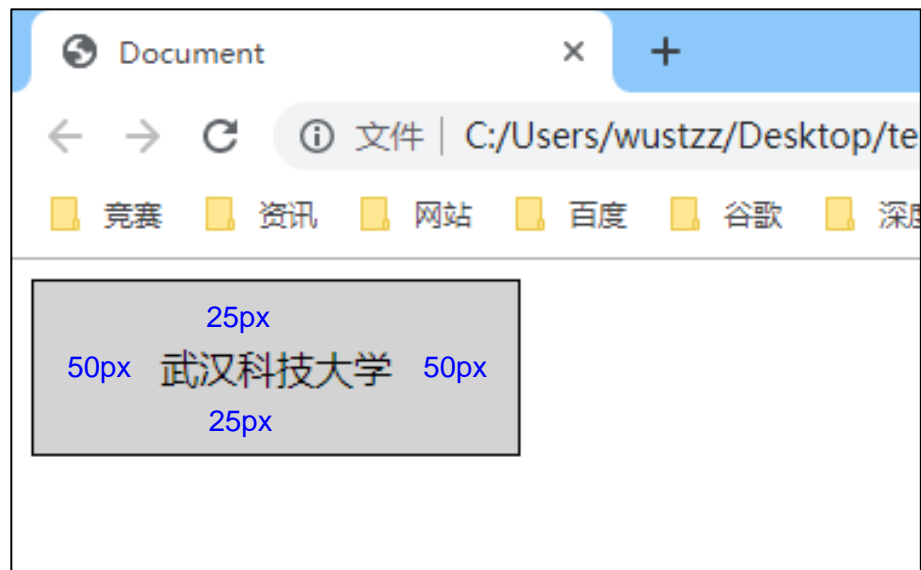
补充1:

如果四个值都相同, 则直接简写为: padding: 25px;

补充2: padding: 25px 50px 75px;

含义: 上25px 右50px 下75px 左省略默认=右=50px

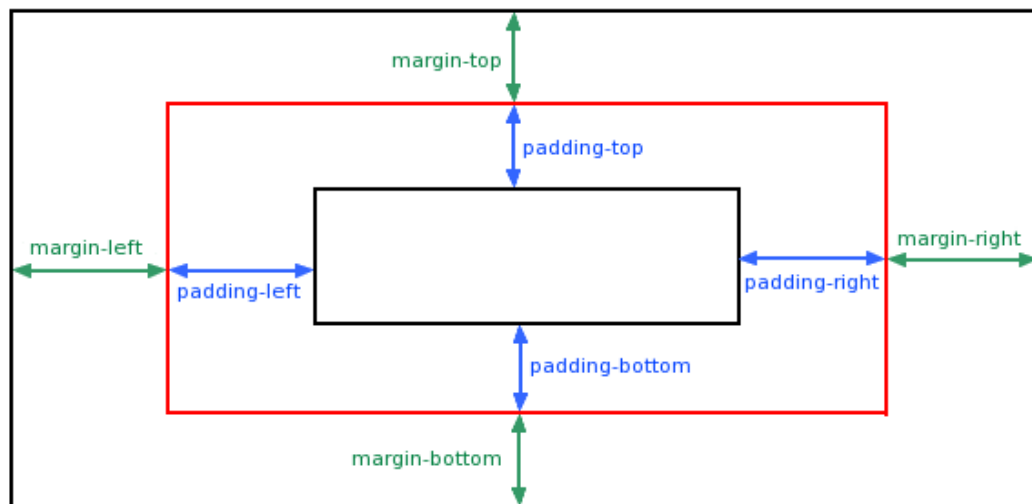
运行效果：



3. margin外边距属性

注：margin值可以为负值

| 属性 | 描述 |
|---------------|------------------------|
| margin | 简写属性。在一个声明中设置元素的外边距属性。 |
| margin-bottom | 设置元素的下外边距。 |
| margin-left | 设置元素的左外边距。 |
| margin-right | 设置元素的右外边距。 |
| margin-top | 设置元素的上外边距。 |



margin的简写：

```
div.margin {  
    border: 1px solid black;  
    width: 300px;  
    margin: 20px 40px;  
}
```

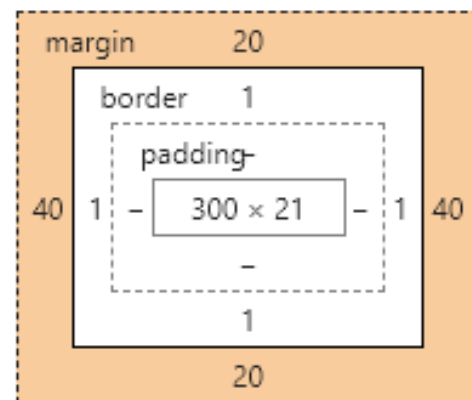
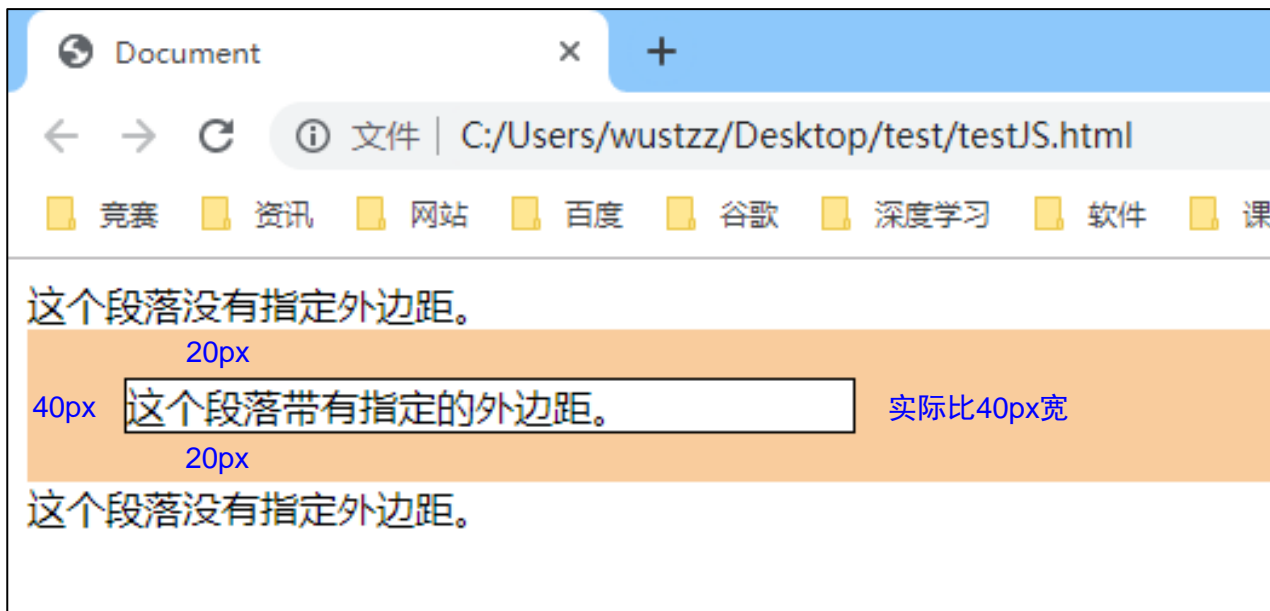
含义：上下外边距=20px 左右外边距=40px

<div>这个段落没有指定外边距。</div>

<div class="margin">这个段落带有指定的外边距。</div>

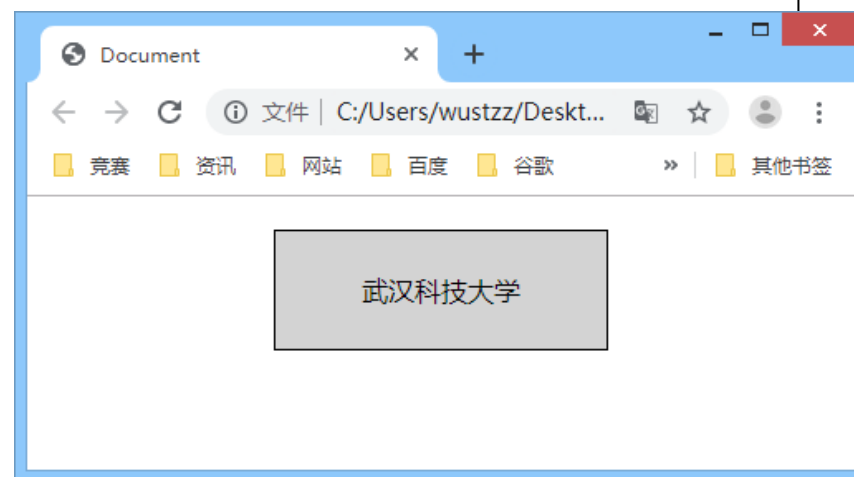
<div>这个段落没有指定外边距。</div>

运行效果：



margin示例：元素居中效果

```
div.center {  
    background-color: lightgrey;  
    text-align: center;  
    width: 100px;  
    padding: 25px 50px ;  
    border: 1px solid black;  
    margin: 20px auto 30px auto;  
}
```



```
<div class="center">武汉科技大学</div>
```

关键：这左右两个边距使用auto！

外边距合并问题

```
* {  
  padding: 0;  
  margin: 0;  
}
```

功能：先将所有元素的内外边距清0，
避免不同浏览器默认样式干扰

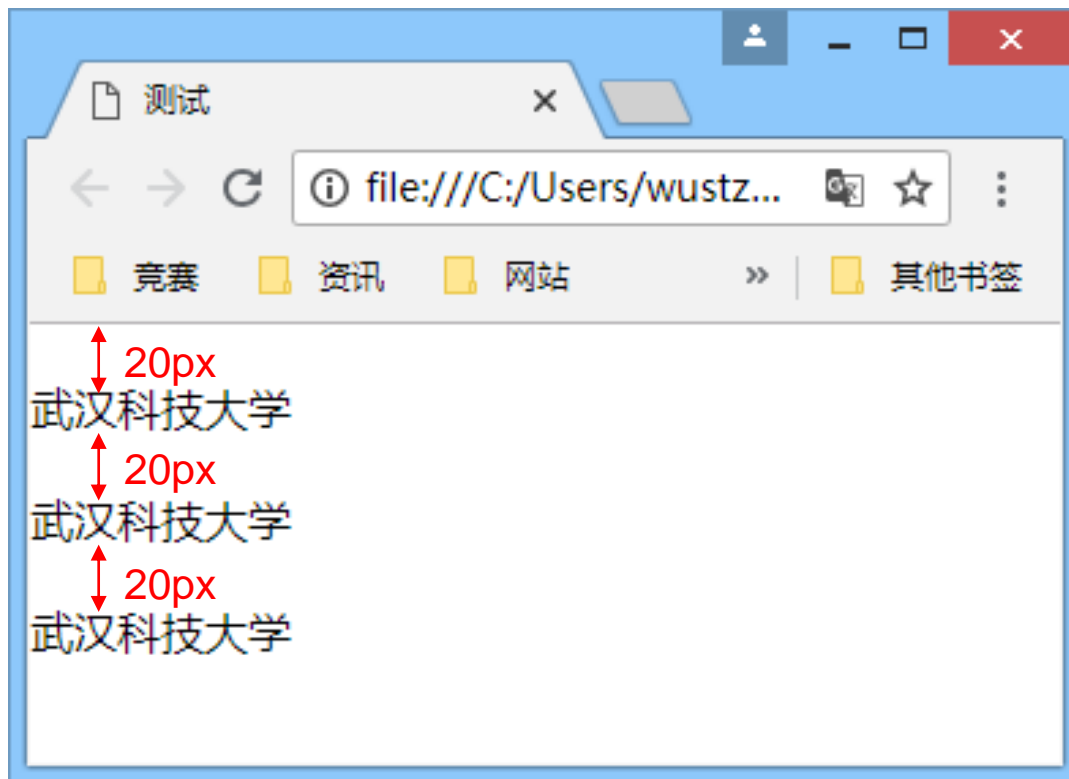


```
div {  
  margin-top: 20px;  
  margin-bottom: 20px;  
}
```

思考问题：
div之间间隔是40px吗？

```
<div>武汉大学</div>  
<div>武汉大学</div>  
<div>武汉大学</div>
```

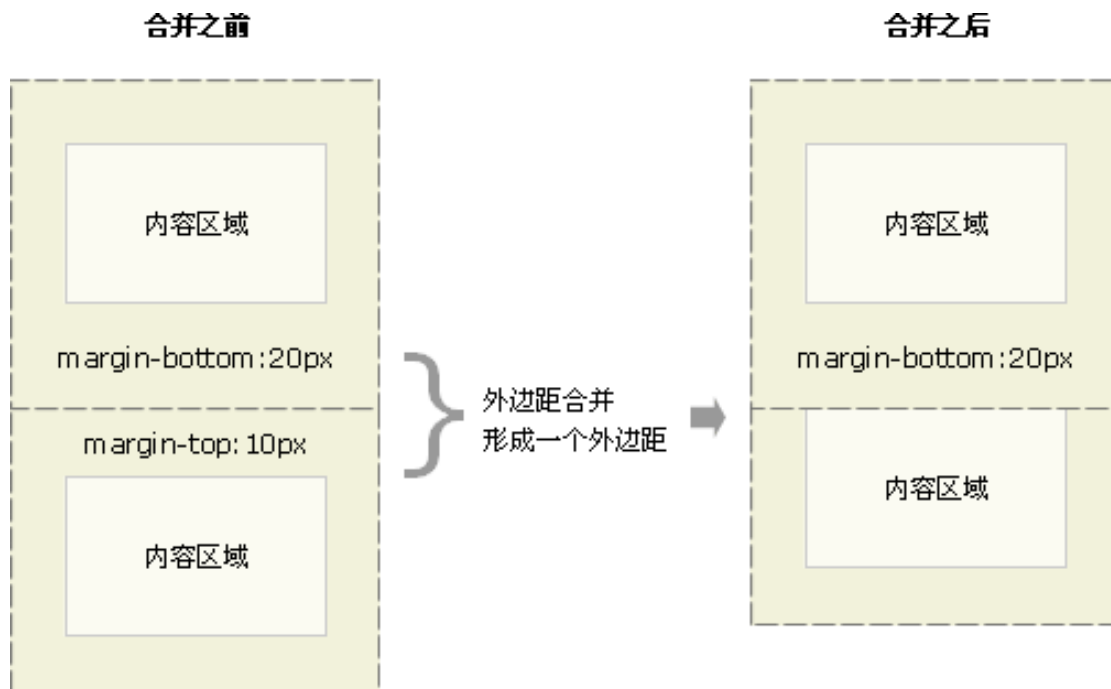
测试结果：上下间距都是20px！



外边距合并

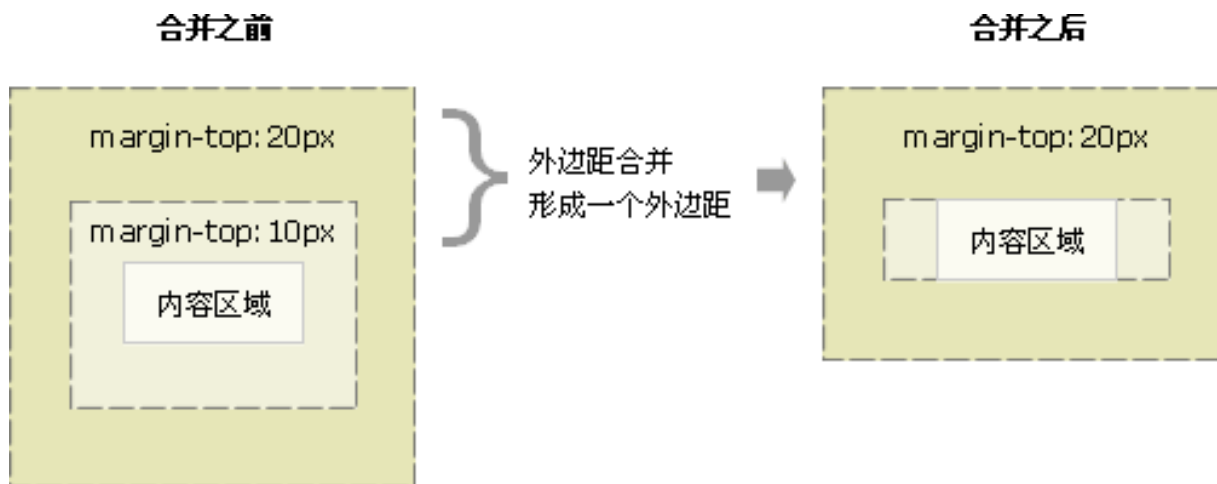
备注：只有普通文档流中块框的垂直外边距才会发生外边距合并。行内框、浮动框或绝对定位之间的外边距不会合并。

- 外边距合并指：当两个垂直外边距相遇时，它们将形成一个外边距
- 合并后外边距的高度等于两个发生合并的外边距的高度中的较大者



外边距合并（续）

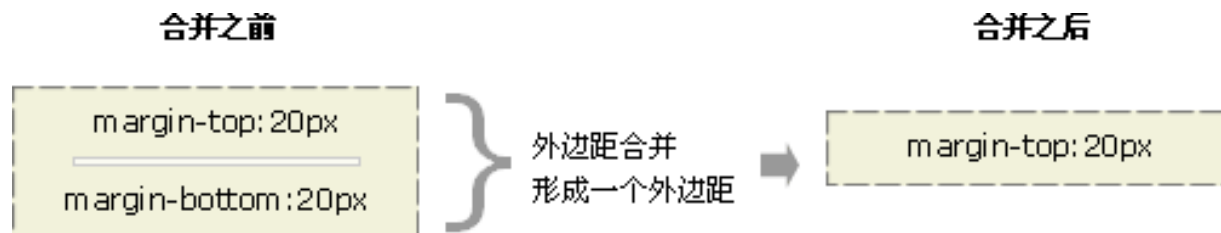
- 当一个元素包含在另一个元素中时（假设没有内边距或边框把外边距分隔开），它们的上和/或下外边距也会发生合并



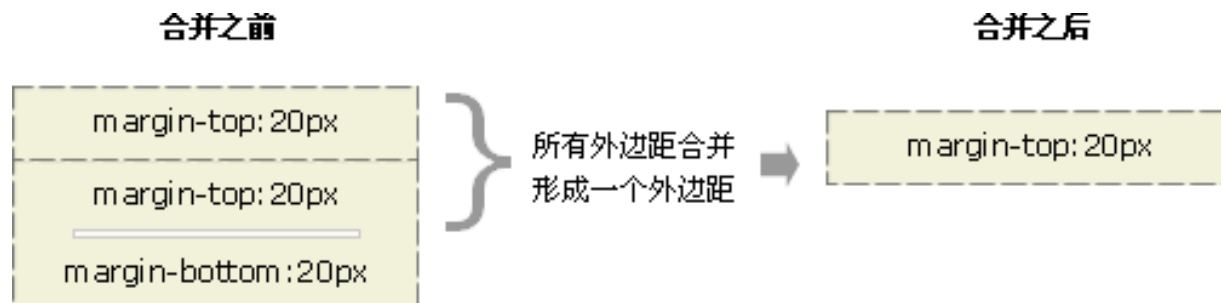
外边距合并（续）

■ 外边距甚至可以与自身发生合并：

假设有一个空元素，它有外边距，但是没有边框或填充。在这种情况下，上外边距与下外边距就碰到了一起，它们会发生合并：



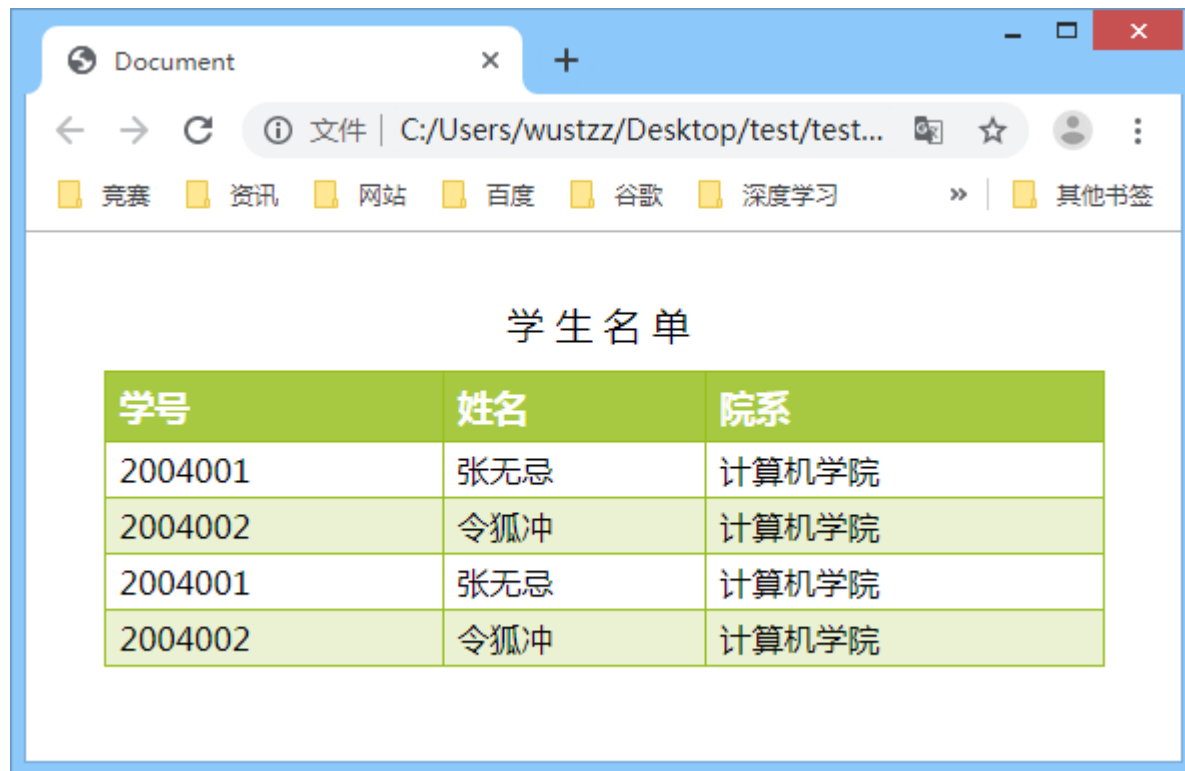
如果这个外边距遇到另一个元素的外边距，它还会发生合并：



边框综合示例：漂亮的表格

```
<table class="table">
  <caption>学生名单</caption>
  <tr>
    <th>学号</th>    <th>姓名</th>    <th>院系</th>
  </tr>
  <tr>
    <td>2004001</td> <td>张无忌</td> <td>计算机学院</td>
  </tr>
  <tr class="alt">
    <td>2004002</td> <td>令狐冲</td> <td>计算机学院</td>
  </tr>
  <tr>
    <td>2004001</td> <td>张无忌</td> <td>计算机学院</td>
  </tr>
  <tr class="alt">
    <td>2004002</td> <td>令狐冲</td> <td>计算机学院</td>
  </tr>
</table>
```

运行效果



The screenshot shows a web browser window with a single tab titled 'Document'. The address bar displays the file path 'C:/Users/wustzz/Desktop/test/test...'. Below the address bar, there are several bookmark icons labeled '竞赛', '资讯', '网站', '百度', '谷歌', and '深度学习', followed by a '其他书签' (Other bookmarks) link. The main content area of the browser displays a table titled '学生名单' (Student List).

| 学号 | 姓名 | 院系 |
|---------|-----|-------|
| 2004001 | 张无忌 | 计算机学院 |
| 2004002 | 令狐冲 | 计算机学院 |
| 2004001 | 张无忌 | 计算机学院 |
| 2004002 | 令狐冲 | 计算机学院 |

```
.table {  
    font-family: 微软雅黑;  
    width: 500px;  
    margin: 5px auto 20px auto; /*表格居中*/  
    border-collapse: collapse;  
}  
  
.table caption {  
    font-size: 1.2em;  
    letter-spacing: 5px;  
    margin-bottom: 10px;  
}  
  
.table th, .table td {  
    border: 1px solid #98bf21;  
}
```

```
.table th {  
    height: 20px;  
    color: #ffffff;  
    font-size: 1.1em;  
    text-align: left;  
    padding: 5px 7px;  
    background-color: #A7C942;  
}  
  
.table td {  
    color: #000000;  
    padding: 3px 7px;  
}  
  
.table tr.alt td {  
    background-color: #EAF2D3;  
}
```

[【返回】](#)

综合示例1



综合示例2

欢迎注册会员

手机号码: 必填

创建密码: 必填

注册邮箱: 必填

验证码: FJqR

性别: ☐ 男 ☐ 女

生日: 

年龄:

籍贯:

个人学历:

月薪: 5000

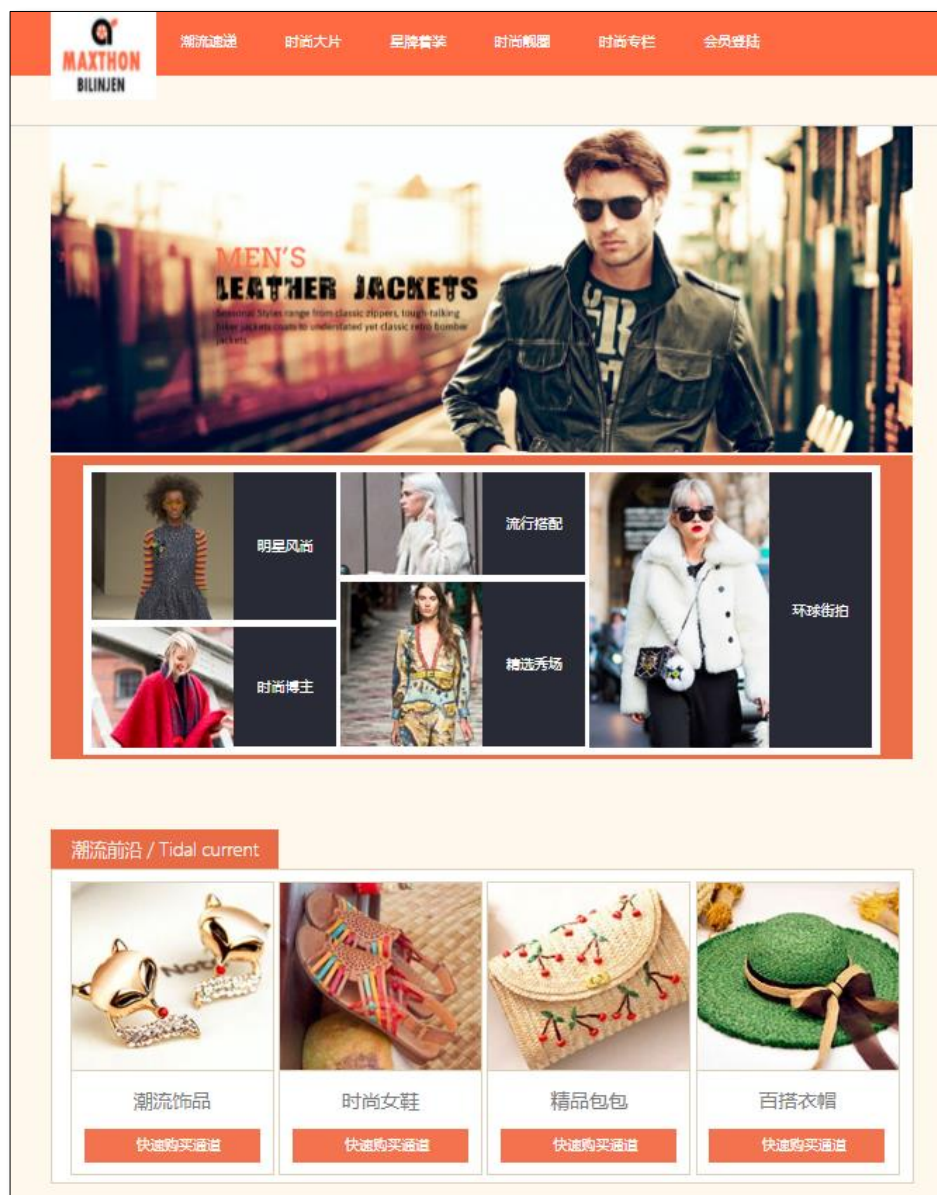
个人爱好: ☐ 唱歌 ☐ 跑步 ☐ 游泳

个人照片: 未选择任何文件

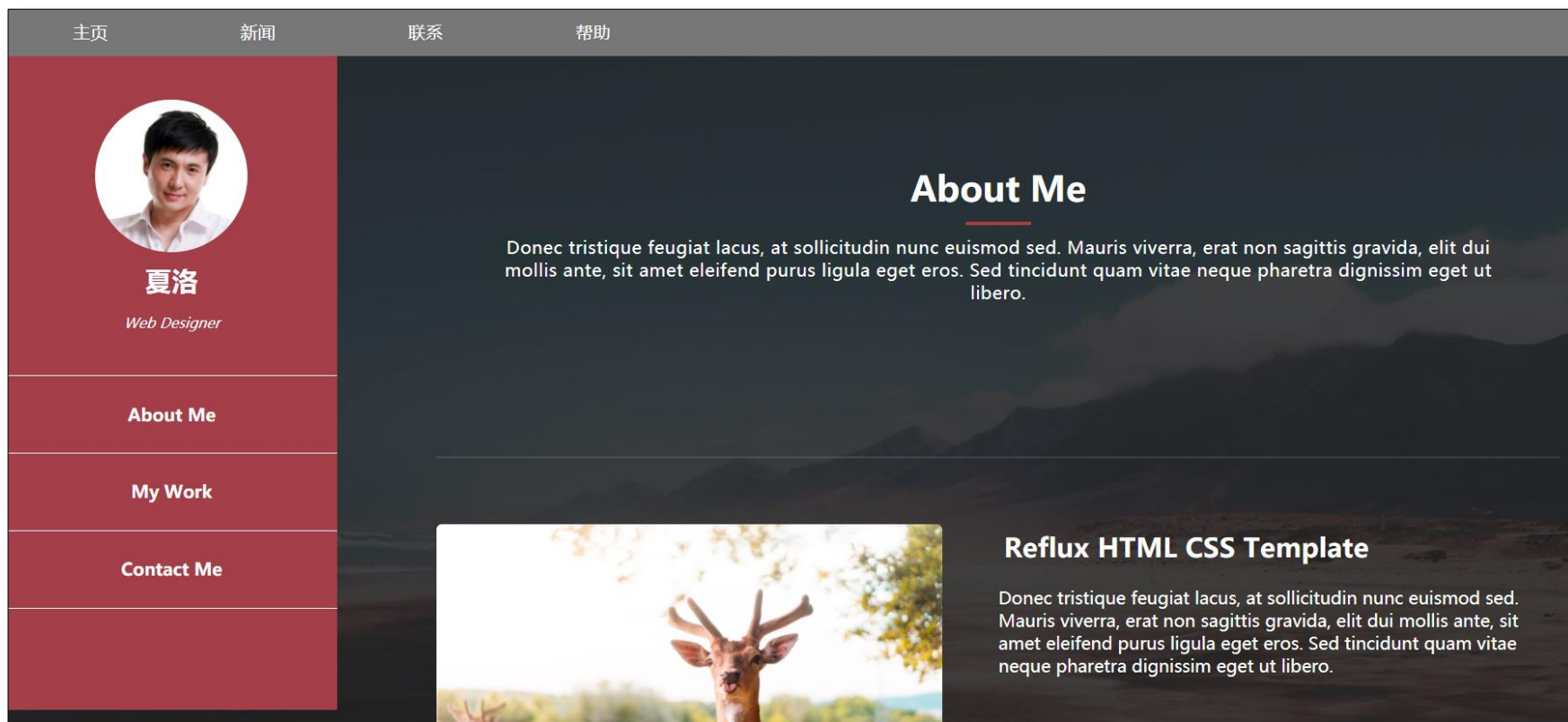


个人简介: 

综合示例3



综合示例4



综合示例5



[【返回】](#)

附录1：display 显示属性

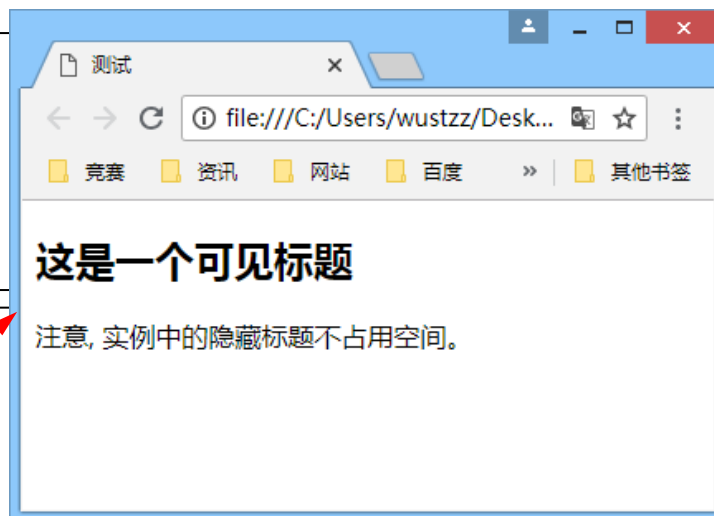
| display属性值(常见部分) | 描述 |
|------------------|-------------------------------------|
| none | 此元素不会被显示。 |
| block | 此元素将显示为 块级元素 ，此元素前后会带有换行符。 |
| inline | 默认。此元素会被显示为 内联元素 ，元素前后没有换行符。 |
| inline-block | 行内块元素 。 |
| flex | 弹性布局 |

■ display:none 隐藏元素

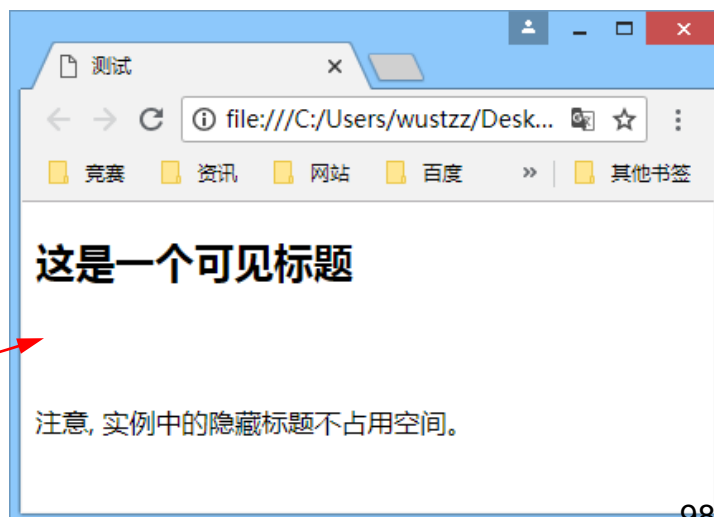
注：不会占用任何空间

```
.hidden {  
    display: none;  
}
```

```
<h2>这是一个可见标题</h2>  
<h2 class="hidden">这是一个隐藏标题</h2>  
<p>注意, 实例中的隐藏标题不占用空间。</p>
```



补充：visibility:hidden;隐藏的元素仍占用原有空间



■ display:inline 内联元素（行内元素）

```
div {
```

```
    display: inline;
```

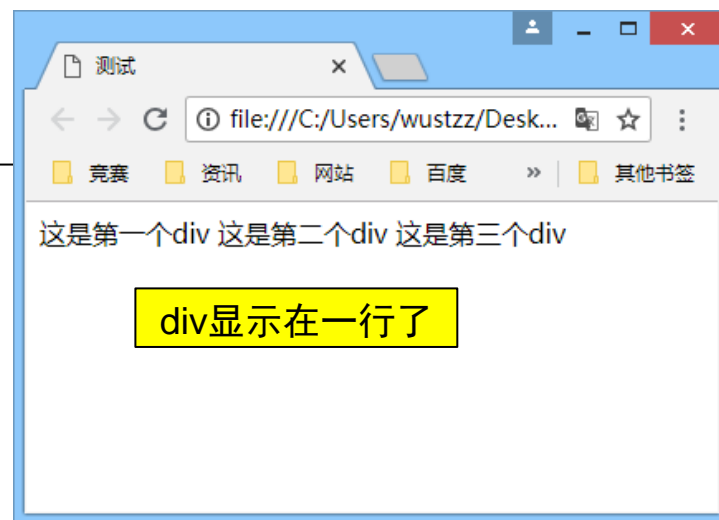
将div显示为内联元素

```
}
```

<div>这是第一个div</div>

<div>这是第二个div</div>

<div>这是第三个div</div>



display:inline 行内元素(内联元素)特点

- 和其他元素都在一行；
- 高、行高以及顶和底内、外边距不可改变；
- 宽度就是它的文字或图片的宽度，也不可改变。
- 如span、a、label、strong、b、i、em等标签
- 前面示例：span: width+text-align:center 文字居中无效
原因：width不可修改

■ display:block 块级元素

```
span {
```

```
    display: block;
```

将span显示为块元素

```
}
```

<h2>武汉科技大学</h2>

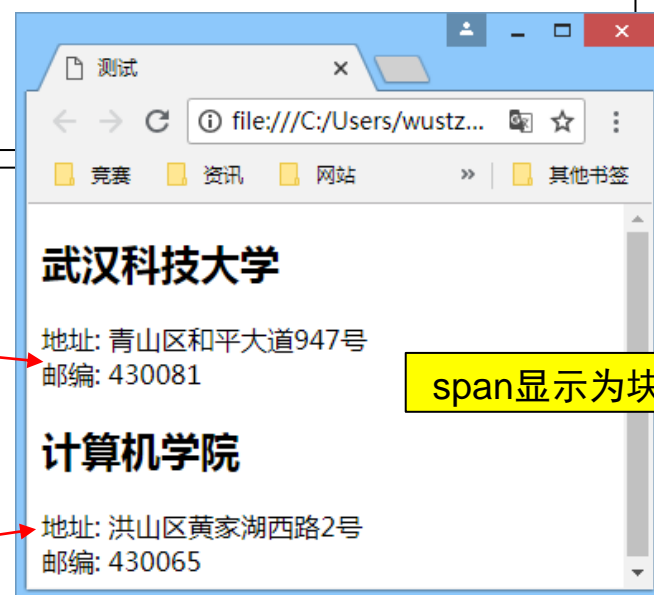
地址: 青山区和平大道947号

邮编: 430081

<h2>计算机学院</h2>

地址: 洪山区黄家湖西路2号

邮编: 430065



display:block 块级元素特点

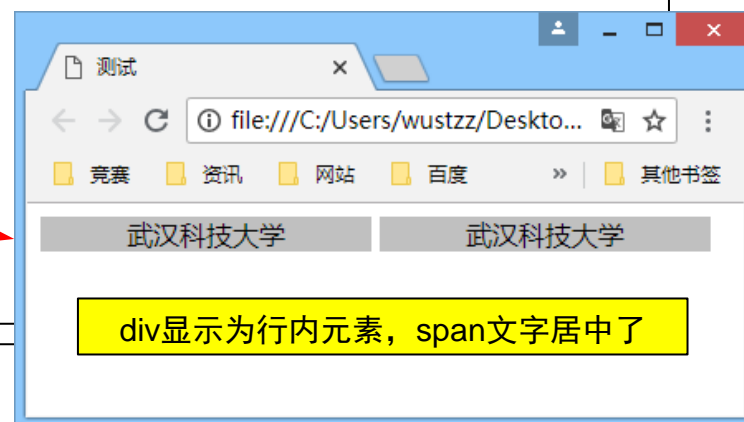
- 总在新行上开始，而且其后元素也必须另起一行显示；
- 高、行高以及内、外边距都可控制；
- 宽度缺省是它的容器的100%，除非设定一个宽度。
- 如div、p、h1-h6、form、ul、li、table、tr 等标签
- 前面示例：div: width+text-align:center 文字居中有效

■ display:inline-block 行内块元素

示例：span文字居中对齐

```
.center{  
    display:inline-block;  
    background-color: silver;  
    width:200px;  
    text-align: center;  
}
```

```
<div class="center">武汉科技大学</div>  
<span class="center">武汉科技大学</span>
```



display:inline-block 行内块元素特点

- inline和block特点结合
- 元素为内联元素，但元素内容作为块级呈现
- 表现为同行显示，并可修改宽高、内外边距等属性。
- 如 input、img、select、td元素
- 前面示例span文字居中：
 - 先变成 display:inline-block模式（不改变原始的同行显示性质）
 - 然后设置：width+text-align:center width起作用

[【返回】](#)

附录2: position 定位属性

■ position属性五个值:

没有定位 则 left, right, top, bottom, z-index 属性无效

■ **static**: 默认值, 没有定位, 元素出现在正常的流中

■ **relative**: 相对定位, 元素的定位是相对其原本位置

经常使用

■ **absolute**: 绝对定位, 相对于 static 定位以外的第一个父元素进行定位, 如果都没有, 就是相对于body

■ **fixed**: 固定定位, 总是相对于浏览器窗口定位

■ **sticky**: 粘性定位, 基于用户的滚动位置来定位

注: body和浏览器窗口大小有区别, 但坐标轴是一样的

■ relative定位

- 相对定位元素的定位是相对其原本位置
- 相对定位元素的内容可能和其他元素重叠，但它原本所占的空间仍然保留（注意）

```
div.pos_left {  
    position: relative; /*相对自身正常位置*/  
    left: -20px;  
}
```

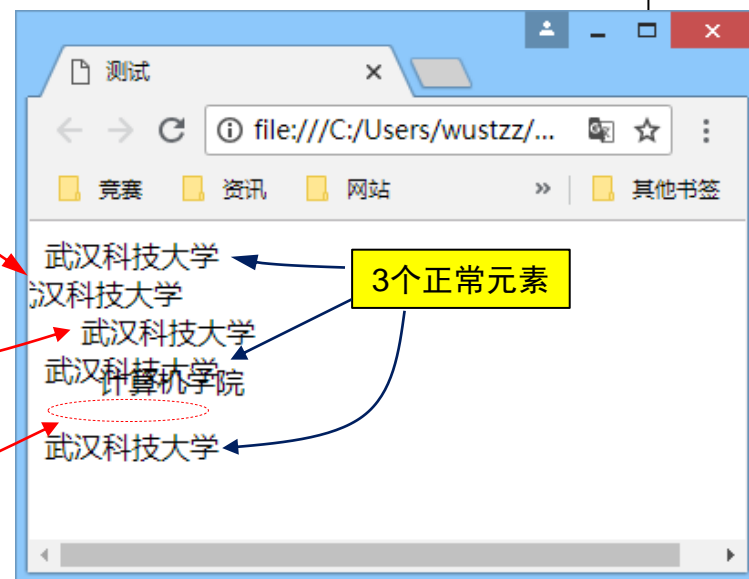
元素的原始左侧
位置减去 20 像素

```
div.pos_right {  
    position: relative;  
    left: 20px;  
}
```

元素的原始左侧
位置增加 20 像素

```
div.pos_top {  
    position: relative;  
    left: 30px;  
    top: -15px;  
}
```

元素重叠，
原有空间仍然保留



页面内容：

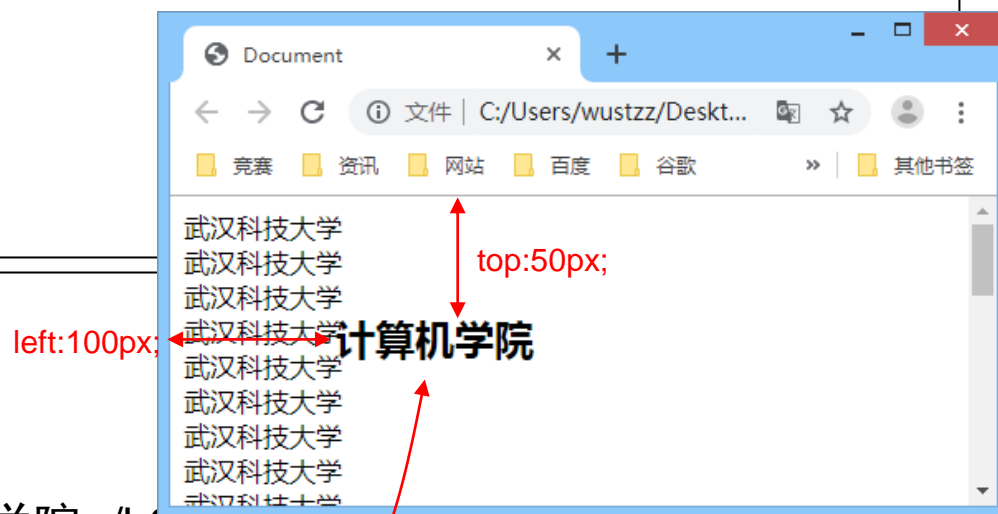
```
<div>武汉大学</div>  
<div class="pos_left">武汉大学</div>  
<div class="pos_right">武汉大学</div>  
<div>武汉大学</div>  
<div class="pos_top">计算机学院</div>  
<div>武汉大学</div>
```

■ absolute定位

- 绝对定位是相对于已定位的第一个父元素进行定位，若无则相对于body
- 用绝对定位,可以把一个元素放在页面任何位置，当窗口滚动时元素可能会被滚掉
- 绝对定位使元素的位置与文档流无关，因此不占据空间
- 绝对定位的元素可能和其他元素重叠

```
h2.absolute {
    position:absolute; /*绝对定位*/
    left:100px;
    top:50px;
}
```

```
<body>
  <div>武汉大学</div>
  <div>武汉大学</div>
  <h2 class="absolute">计算机学院</h2>
  <div>武汉大学</div>
  <div>武汉大学</div>
  ...省略多个div
</body>
```



本例 h2 绝对定位相对的是 body 元素

理解：绝对定位相对于第一个已定位(absolute或relative)父元素

```
<body>
  <div class="parent">
    <h2 class="absolute">计算机学院</h2>
  </div>
</body>
```

可看出h2父元素是div

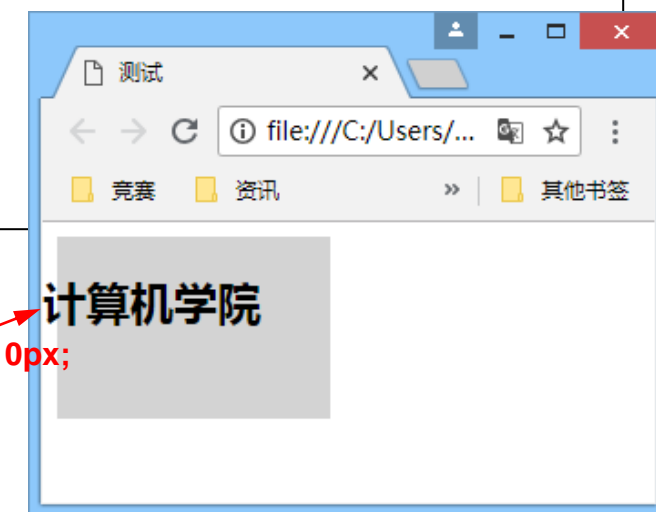
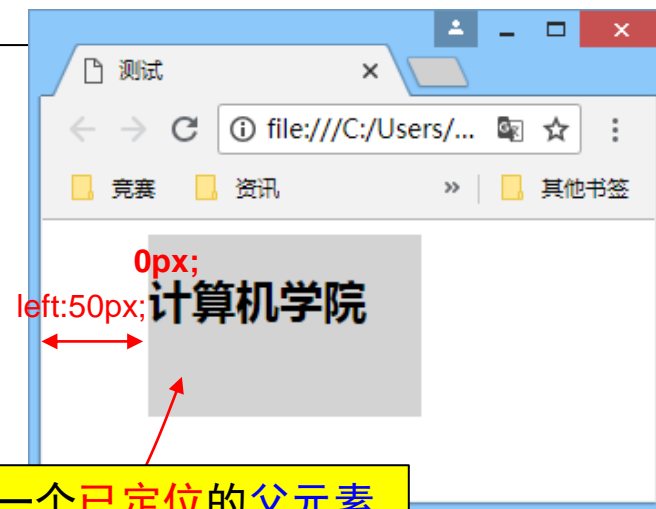
测试示例

```
div.parent {  
    position: absolute; /*已定位*/  
    left: 50px;  
    width: 150px;  
    height: 100px;  
    background-color: lightgray;  
}  
  
h2.absolute {  
    position: absolute; /*绝对定位*/  
    left: 0px;  
}
```

div是h2的第一个已定位的父元素
因此绝对定位的left相对的是div

比较一下：如将div的position定位属性去掉，则h2绝对定位相对的是body。

由于此时div没有定位(默认static)，其 left属性无效。

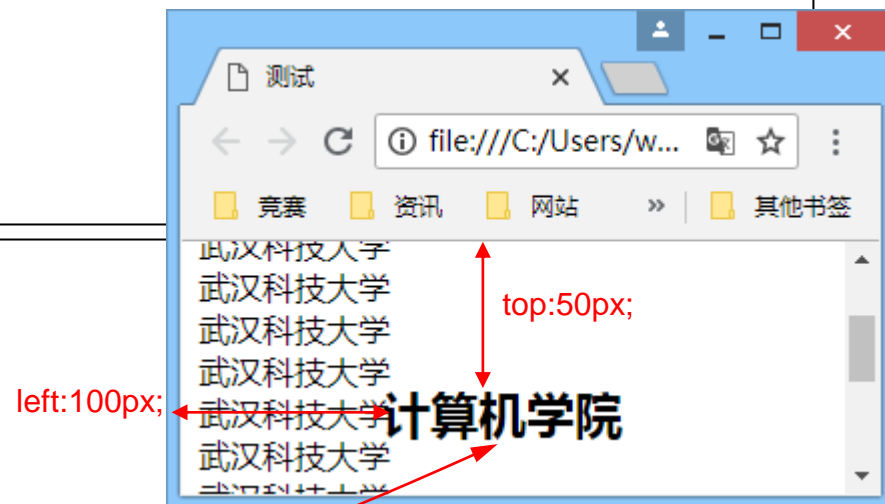


■ fixed定位

- 固定定位的元素位置总是相对于浏览器窗口定位
- 不论窗口滚动与否，元素都会留在那个位置(表现为浮在上方)
- 固定定位使元素的位置与文档流无关，因此不占据空间
- 固定定位的元素可能和其他元素重叠

```
h2.fixed {  
    position:fixed; /*总是相对于浏览器窗口*/  
    left:100px;  
    top:50px;  
}
```

```
<div>武汉大学</div>  
<div>武汉大学</div>  
<h2 class="fixed">计算机学院</h2>  
<div>武汉大学</div>  
<div>武汉大学</div>  
...下面省略多个div
```



页面滚动时fixed定位的元素不会移动，像是浮在窗口上

■ sticky定位

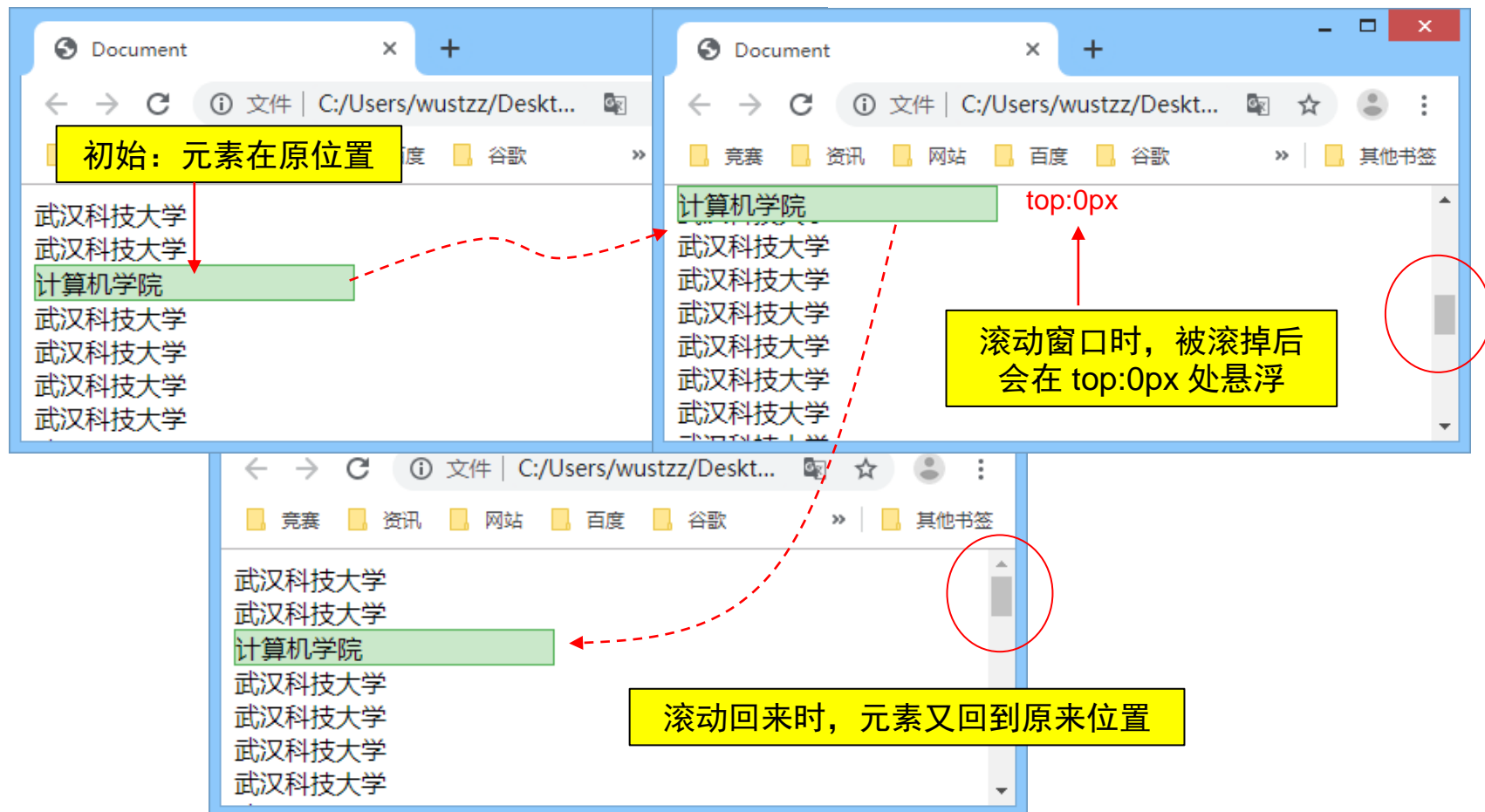
- 粘性定位依赖于用户的滚动位置
- 在跨越特定阈值前表现为relative相对定位，之后表现为fixed固定定位
- 特定阈值指的是 top, right, bottom 或 left 之一，才可使粘性定位生效

```
div.sticky {  
    position: sticky; /*粘性定位要配合位置值才有效果*/  
    top: 0px;  
    width: 100px;  
    background-color: #cae8ca;  
    border: 2px solid #4CAF50;  
}
```

```
<div>武汉大学</div>  
<div>武汉大学</div>  
<div class="sticky">计算机学院</div>  
<div>武汉大学</div>  
<div>武汉大学</div>  
<div>武汉大学</div>  
...下面省略多个div
```

运行情况见下页

sticky定位运行情况:



关于z-index 属性：指定一个元素的堆叠顺序

- z-index使用前提：元素一定是定位元素
- z-index 默认值是 0，其值可为正、负值
- z-index 值大的元素在前面（上面）
 - z-index相当于定义了一个垂直显示区的z轴。如果为正数，则离用户更近，为负数则表示离用户更远。
- 没有指定z-index或者值相等，最后定位的元素将被显示在最前面
- 在无z-index参与的情况，有position的元素在前面

z-index示例:

```
img {  
    position: absolute; /*已定位*/  
    left: 0px;  
    top: 0px;  
    z-index: -1;  
}
```

图片位于文字下方



```

```

```
<div>
```

```
    <h1>Lorem Ipsum Dolor</h1>
```

```
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy  
    nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

```
    <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit  
    lobortis nisl ut aliquip ex ea commodo consequat.</p>
```

```
</div>
```

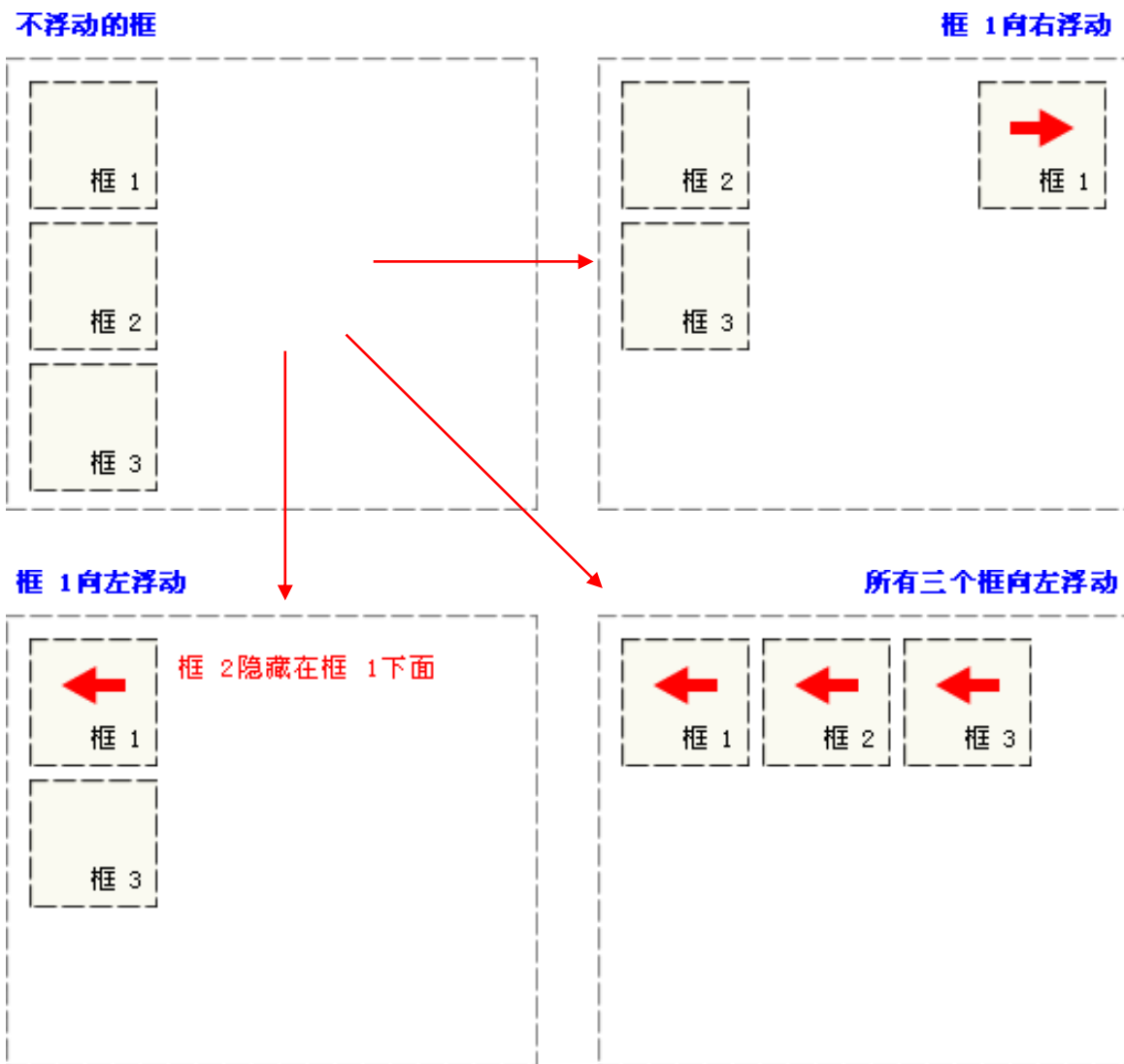
[【返回】](#)

附录3: float 属性

- float属性会使元素向左或向右移动，其周围的元素也会重新排列
- 一个浮动元素会尽量向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止
- 浮动元素之后的元素将围绕它，浮动元素之前的元素不受影响
- 使用clear属性可以消除浮动影响

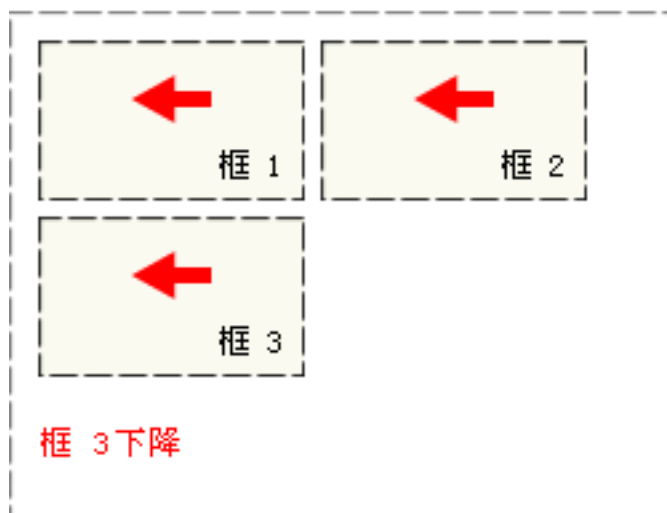
注：元素浮动后意味着元素只能左右移动而不能上下移动

浮动图示

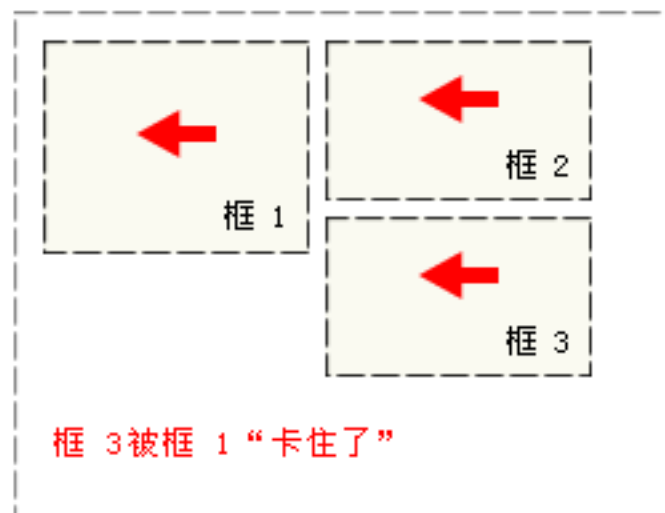


浮动图示

所有三个框向左浮动



如果包含框太窄，无法容纳水平排列的三个浮动元素，那么其它浮动块向下移动



如果浮动元素的高度不同，那么当它们向下移动时可能被其它浮动元素“卡住”

float示例1:

```
img {  
    float:right;  
}
```

图片浮动到右边
文本流将环绕在它左边

```

```

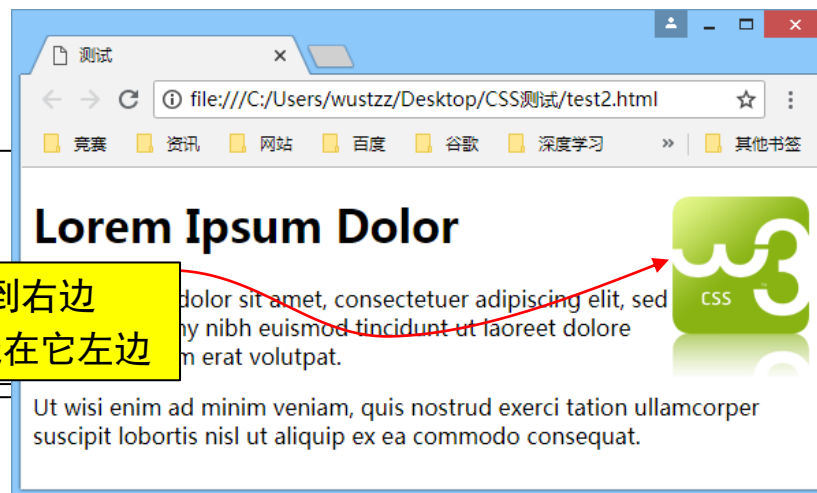
```
<div>
```

```
    <h1>Lorem Ipsum Dolor</h1>
```

```
    <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy  
    nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.</p>
```

```
    <p>Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit  
    lobortis nisl ut aliquip ex ea commodo consequat.</p>
```

```
</div>
```



float示例2:

```
  
  
  

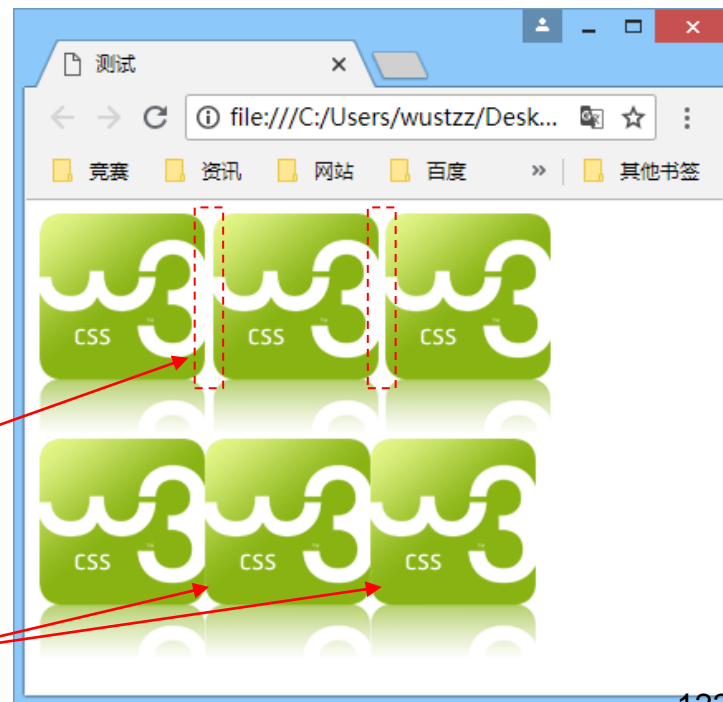
```

先看常规没有浮动效果:

- 由于img属于内联(行内)元素, 所以会显示在一行, 空间不够时会自动换行
- 前面三张图片之间出现空白有间隔, 但后面三个img没有间隔
原因: 浏览器会把inline元素间的空白字符(空格、换行、tab等)渲染成一个空格

中间有空白间隔

中间没有空白间隔



img添加float后：

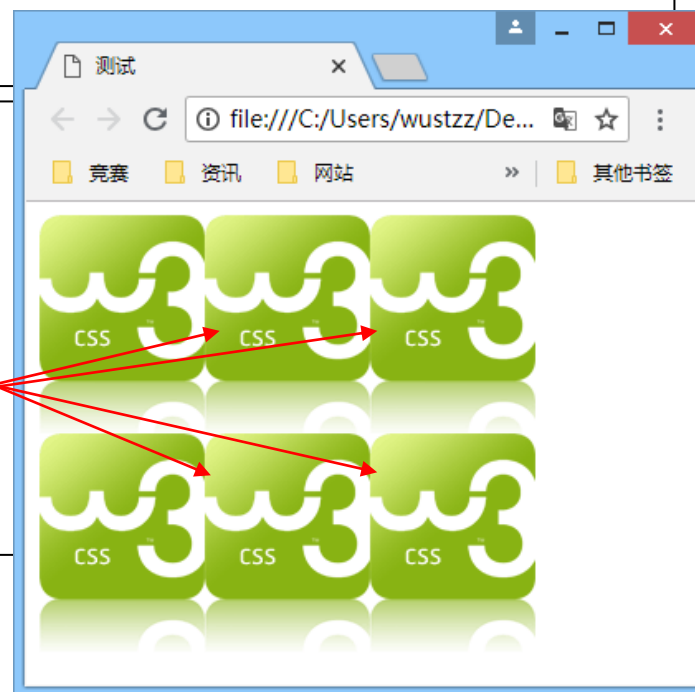
```
img {  
    float:left;  
}
```

图片向左边浮动，彼此相邻无空白间隔

```
  
  
  
  
  

```

中间都没有空白间隔



clear属性示例:

```
img {  
    float:left;  
    margin:5px;  
}
```

```
  
  
  
<h3>这是一个标题</h3>  
  
  

```

先看没有添加clear的效果:

- 前面3个浮动到左边
- <h3>受到左浮动影响环绕在右边 (注:<h3>没有设置浮动, 但自身是块级元素, 后面的元素会另起一行)
- 其后3个元素会尽量向左浮动(体会"尽量"一词):
- 由于当前行在<h3>下面还有较多空白, 因此后面3个的两个就尽量浮到<h3>下方(会被前面浮动卡住), 最后一个当前行无法容下只好换到新的一行(注意它的位置)



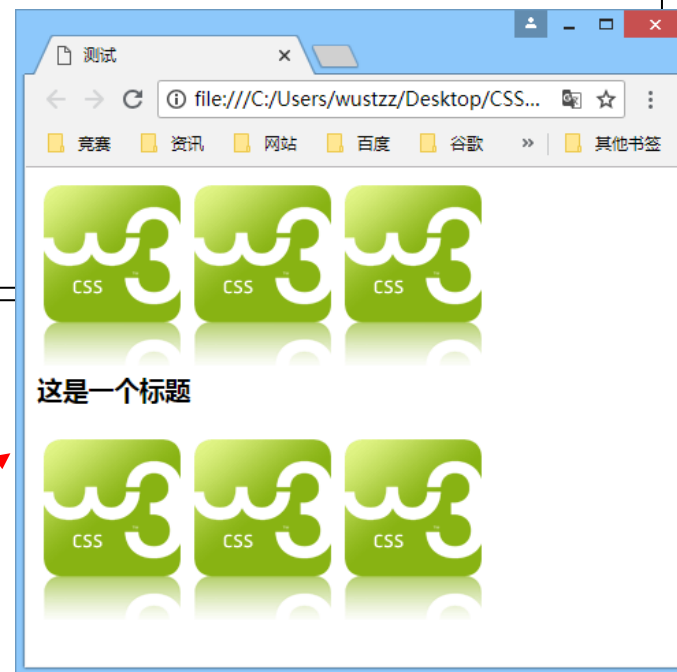
h3添加clear属性后：

```
img {  
    float:left;  
    margin:5px;  
}  
  
.clear {  
    clear: both; /*在左右两侧均不允许浮动元素*/  
}
```

clear 属性规定元素的哪一侧
不允许其他浮动元素
取值：none|left|right|both

```
  
  
  
<h3 class="clear">这是一个标题</h3>  
  
  

```



【完】