



Java 软件工程师

(第一阶段)



目 录

序言	5
第 1 章 HTML网页制作入门	6
1.1 本章目标.....	6
1.2 内容	6
1. 2. 1 W3C组织介绍.....	6
1. 2. 2 什么是HTML.....	6
1. 2. 3 主流浏览器介绍.....	7
1. 2. 4 HTML文件的全局架构标签.....	7
1. 2. 5 HTML的注释.....	8
1.3 本章作业.....	8
第 2 章 HTML常用标签	9
2.1 本章目标.....	9
2.2 内容	9
2. 2. 1 基本块级标签.....	9
2. 2. 2 常用于页面布局的块级标签.....	13
2. 2. 3 行级标签.....	25
2. 2. 4 其他标签.....	29
2.3 本章作业.....	34
第 3 章 HTML布局--表格	36
3.1 本章目标.....	36
3.2 内容	37
3. 2. 1 表格的基本结构.....	37
3. 2. 2 表格的跨行跨列.....	38
3. 2. 3 表格的高级标签.....	40
3. 2. 4 表格实现图文布局.....	41
3. 2. 5 表单实现登录布局.....	44
3. 2. 6 表格布局的缺点.....	44
3.3 本章作业.....	45
第 4 章 表单元素	45
4.1 本章目标.....	45
4.2 内容	45
4. 2. 1 文本框.....	47
4. 2. 2 密码框.....	49
4. 2. 3 按钮框.....	50
4. 2. 4 单选按钮框.....	50
4. 2. 5 复选框.....	51
4. 2. 6 文件域.....	52
4. 2. 7 下拉列表.....	53
4. 2. 8 多行文本框.....	54
4. 2. 9 隐藏域.....	54

4.2.10	readonly and disabled.....	55
4.3	本章作业.....	57
第 5 章	窗口划分frameset 和 frame	58
5.1	本章目标.....	58
5.2	内容.....	58
5.2.1	frameset.....	58
5.2.2	Iframe.....	60
5.2.3	Frame.....	62
5.3	本章作业.....	63
第 6 章	CSS级联样式表.....	63
6.1	本章目标.....	63
6.2	内容.....	63
6.2.1	CSS概念及语法.....	63
6.2.2	CSS的引用方式.....	64
6.2.3	CSS选择器.....	64
6.2.4	常用的CSS属性讲解.....	71
6.2.5	常用的CSS属性参考字典.....	75
6.2.6	盒子模型.....	78
6.2.7	浮动.....	80
6.2.8	定位.....	81
6.2.9	DIV内嵌.....	85
6.2.10	分级属性.....	86
6.2.11	鼠标属性.....	87
6.2.12	CSS应用.....	87
6.3	本章作业.....	104
第 7 章	JavaScript语言基础.....	105
7.1	本章目标.....	105
7.2	内容.....	105
7.2.1	JavaScript概述.....	105
7.2.2	JavaScript特点.....	106
7.2.3	JavaScript变量.....	106
7.2.4	常用的输入/输出.....	107
7.2.5	JavaScript数据类型.....	108
7.2.6	JavaScript注释（单行、多行）	113
7.2.7	JavaScript运算符.....	114
7.2.8	JavaScript逻辑控制语句.....	116
7.2.9	常用的系统对象.....	129
7.2.10	函数.....	146
7.2.11	自定义函数.....	149
7.2.12	函数的匿名调用.....	150
7.2.13	变量的作用域.....	151
7.2.14	数组.....	152
7.2.15	BOM对象（Window对象）	157
7.2.16	DOM对象.....	170
7.2.17	表单基本验证技术.....	198

第 8 章	JAVA语言基础	211
8.1	本章目标	211
8.2	内容	211
8.2.1	程序的定义	211
8.2.2	开发环境配置	212
8.2.3	MyEclipse简介	214
8.2.4	用Myeclipse创建一个类的方法	215
8.2.5	用Myeclipse运行一个类的方法	216
8.2.6	运行第一个java程序	216
8.2.7	JAVA注释	218
8.2.8	JAVA变量数据类型	219
8.2.9	JAVA编码规范	224
8.2.10	JAVA运算符	225
8.2.11	JAVA 逻辑控制语句	229
8.2.12	类和对象	252
8.2.13	数组	287
8.2.14	常用集合接口	300



序言

南昌思诚科技有限公司(seecen)是一家面向企业、高等院校、大学生，提供信息技术咨询与人才服务的专业机构。公司拥有一支在信息系统研发、项目管理、人才服务等方面拥有丰富运营管理经验的专业团队，业务范围涵盖企业信息系统咨询、人才服务、软件产品服务、软件人才定制岗前实训等方面。公司总部在南昌，同时在深圳建立了办事处。

2014年7月，南昌思诚科技有限公司正式成为Oracle(甲骨文)公司在江西地区唯一官方WDP授权合作伙伴，负责为江西地区高校及个人提供Oracle原厂商认证培训与咨询服务。

基于企业需求的建构主义实训体系

建构主义教学理论与传统教学理念之间最大的差别就在于学习观的不同。建构主义学习观强调学生在教学过程中的主体地位，坚持教学要以学生为中心，要求教师由知识的传授者变成学生学习的帮助者和促进者。传统教学理念局限性和弊端在于它忽视了对学习者能动性的开发与学习自主性的提高。

思诚科技凭借自身对国内软件行业发展趋势、软件企业用人需求、以及大学生就业现状的深刻理解和准确把握，依托Oracle大学强大的技术支持和培训经验，创造性地将企业需求与建构主义教学思想，融入到大学生岗前实训过程中，提出“基于企业需求的建构主义实训体系”。按照合作企业用人需求，定制培养软件人才。企业参与人才选拔和培养过程，将企业特定项目和知识点要求，融入教学大纲，并分阶段考核，缩短员工到岗后的学习和适应周期。



欢迎关注思诚科技官方微信

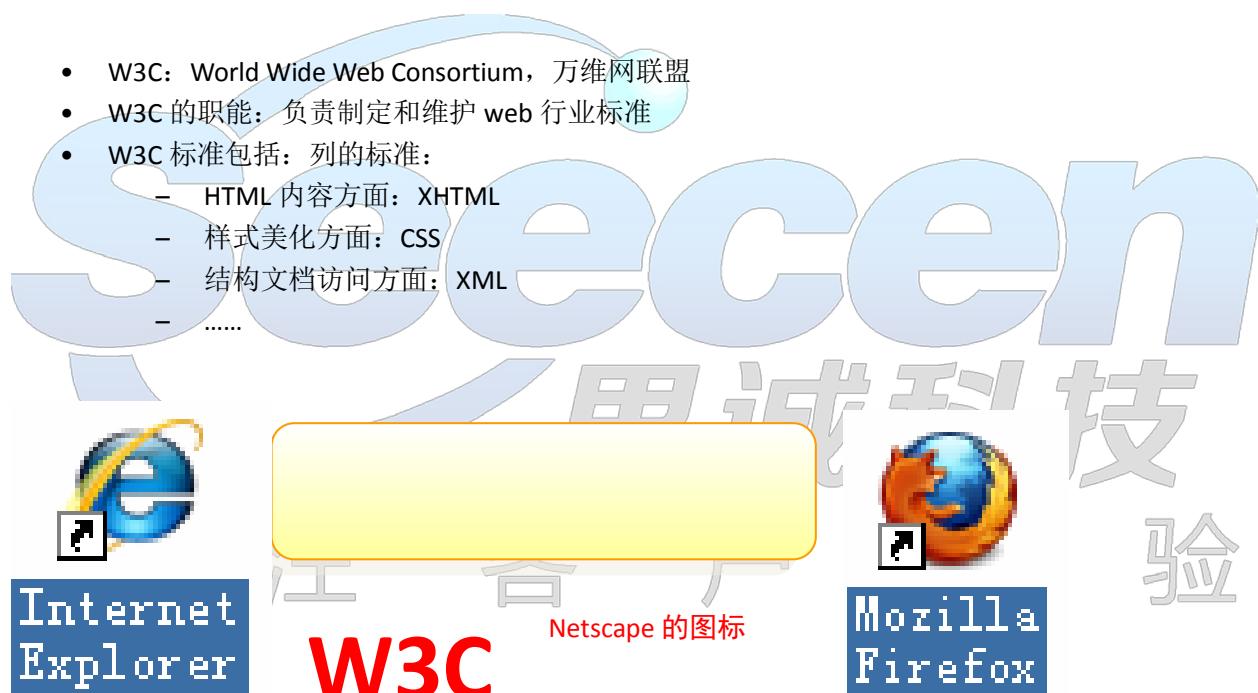
第 1 章 HTML 网页制作入门

1.1 本章目标

- 了解 W3C 组织的概念及职责
- HTML 标记语言的结构
- 主流浏览器

1.2 内容

1.2.1 W3C 组织介绍



1.2.2 什么是 HTML

超文本标记语言 HTML (hypertext markup language) , 是一种用来设计网页的标记语言, 用该语言编写的文件, 以.html 或者.htm 为后缀, 并且由浏览器端解释执行, 生成相应的界面。

html 由 W3C 联盟来规范, 1997 年, 较为成熟的版本 html4 发布, 目前最新的为 html5 版本, 因为 html5 需要浏览器及服务器端的支持, 目前还不成熟。

1.2.3 主流浏览器介绍

IE 浏览器 (微软公司)
火狐浏览器 (firefox)
google 浏览器 (Google Chrome)
safari (苹果公司浏览器)
opera (挪威浏览器)
还有 360 浏览器、搜狗浏览器 ...

1.2.4 HTML 文件的全局架构标签

Html 的基本结构分为两大部分：头（head）和体（body）。

- 1) html 头标记，写描述页面的数据
- 2) html 体标记，是页面的显示内容



- 3) html 全局架构中的其他修饰部分

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<!-- 下面的 meta 标签表示浏览器读到的是一个 html 格式文件，字符编码是 utf-8 -->
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<!--以下 meta 标签是提供搜索关键字和内容描述信息，方便搜索引擎的搜索-->
<meta name= "keywords" content= "淘宝,网上购物,在线交易,交易市场" />
<meta name= "description" content= "淘宝网-亚洲最大、最安全的网上交易平台，提供各类服饰、美容、家居、数码、....." />
</head>
<body>
</body>
</html>
```

1.2.5 HTML 的注释

语法:

```
<!-- -->
```

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>html 注释</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
本人找工作的标准:
<ol>
<li>工资要高</li>
<!-- 被注释的部分是不会在浏览器端显示的 <li>干活要少</li>-->
</ol>
</body>
</html>
```

效果:



1.3 本章作业

练习 html 的全局架构标签书写，并能说出各部分代表的意义；

第 2 章 HTML 常用标签

2.1 本章目标

使用 HTML 的基本结构创建网页

使用行级和块级标签组织页面内容

使用图像标签实现图文并茂的页面

2.2 内容

概要:

HTML 标签由开始标签和结束标签组成

开始标签是被括号包围的元素名

结束标签是被括号包围的斜杠和元素名

某些 HTML 元素没有结束标签, 比如
<hr />

2.2.1 基本块级标签

块级标签: 指一些默认不会在同一行的标签元素, 如果需要在同一行需要设置浮动属性。

2.2.1.1 标题标签

定义: 标题 (Heading) 是通过<h1> - <h6>等标签进行定义的。

请确保将 HTML heading 标签只用于标题。不要仅仅是为了产生粗体或大号的文本而使用标题。

搜索引擎使用标题为您的网页的结构和内容编制索引。

因为用户可以通过标题来快速浏览您的网页, 所以用标题来呈现文档结构是很重要的。

应该将 h1 用作主标题 (最重要的), 其后是 h2 (次重要的), 再其次是 h3, 以此类推。

<h1> 定义最大的标题。<h6> 定义最小的标题。

语法:

<h1>标题</h1>

.....

<h6>标题</h6>

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
```

```
</head>
<body>
<h1>一级标题</h1>
<h2>二级标题</h2>
<h3>三级标题</h3>
<h4>四级标题</h4>
<h5>五级标题</h5>
<h6>六级标题</h6>
</body>
</html>
```

效果：



2. 2. 1. 2 段落标签

定义：可以把 html 文档分成若干，段落是通过 `<p>` 标签定义的，浏览器会自动地在段落的前后添加空行。（`<p>` 是块级元素）

语法：

`<p>` 段落：南昌思诚科技有限公司是一家面向企业提供咨询以及解决方案的 IT 机构，目前的项目拓展已涉及江西邮政、江西电信、江西移动等多家名企.....`</p>`

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
```

```
<meta http-equiv="content-type" content="text/html ;charset=utf-8">

</head>

<body>
<h1>北京欢迎您</h1>
<p>北京欢迎您，有梦想谁都了不起！</p>
<p>有勇气就会有奇迹。</p>

</body>
</html>
```

效果：



2.2.1.3 水平线标签

定义：<hr>标签在 HTML 页面中创建一条水平线。

水平分隔线（horizontal rule）可以在视觉上将文档分隔成各个部分。

可选的属性有

属性	值	描述
align	<ul style="list-style-type: none">• center• left• right	<p>不赞成使用。请使用样式取代它。</p> <p>规定 hr 元素的对齐方式。</p>
noshade	<i>noshade</i>	不赞成使用。 请使用样式取代它。

		规定 <code>hr</code> 元素的颜色呈现为纯色。
<code>size</code>	<ul style="list-style-type: none"><code>pixels</code>	不赞成使用。 请使用样式取代它。 规定 <code>hr</code> 元素的高度（厚度）。
<code>width</code>	<ul style="list-style-type: none"><code>pixels</code><code>%</code>	不赞成使用。 请使用样式取代它。 规定 <code>hr</code> 元素的宽度。

语法:

`<hr/>`

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<h1>北京欢迎您</h1>
<hr/>
<p>北京欢迎您，有梦想谁都了不起！</p>
<p>有勇气就会有奇迹。</p>
</body>
</html>
```

效果:



2.2.2 常用于页面布局的块级标签

2.2.2.1 有序列表标签

定义: ol 是 ordered list (有序列表) 的缩写。Ol 标签通常和 li 一起使用, 列表项目使用数字进行标记。

有序列表始于标签。每个列表项始于标签。列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

语法:

```
<ol>
  <li>列表项 1</li>
  .....
  <li>列表项 N</li>
</ol>
```

ol 英文可以理解为 order list
| 英文可以理解为 list

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<h3>驾校报考流程:</h3>
<ol>
<li>填写表格</li>
<li>缴纳报名费用</li>
<li>领取驾校用书</li>
<li>到指定地点学习</li>
</ol>
</body>
</html>
```

效果:



2.2.2.2 无序列表标签

定义: ul, 是 unordered lists (无序列表) 的缩写, ul 通常和 li 一起使用, 里面 li 元素是无序的。

无序列表是一个项目的列表, 此列项目使用粗体圆点 (典型的小黑圆圈) 进行标记。

无序列表始于标签。每个列表项始于。

列表项内部可以使用段落、换行符、图片、链接以及其他列表等等。

语法:

```
<ul>
  <li>列表项 1</li>
  .....
  <li>列表项 N</li>
</ul>
```

ul 英文可以理解为 unordered list

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<h3>注册步骤:</h3>
<ul>
<li>发起申请</li>
<li>填写表格</li>
<li>注册成功</li>
</ul>
</body>
</html>
```

效果：



2.2.2.3 定义描述标签

定义：<dl>标签定义了定义列表（definition list），<dt>标签定义了定义列表中的项目（即术语部分），<dd>在定义列表中定义条目的定义部分。

语法：

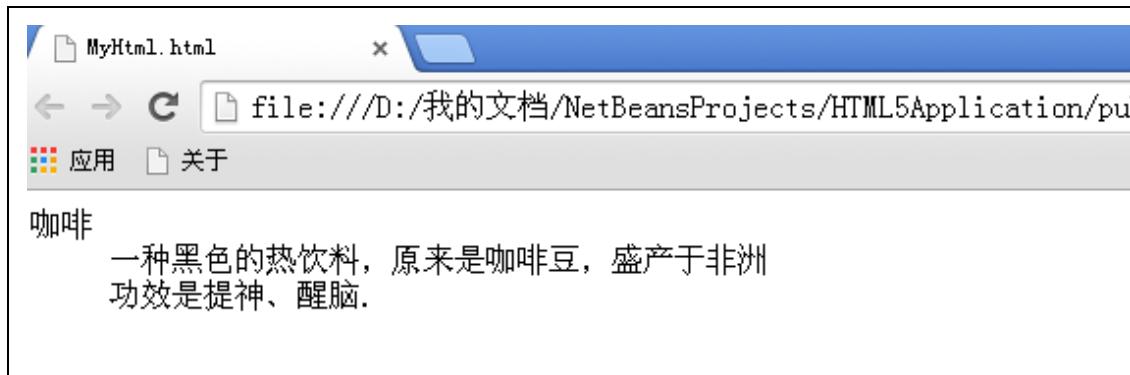
```
<dl>
  <dt>标题</dt>
  <dd>描述 1</dd>
  .....
  <dd>描述 N</dd>
</dl>
```

案例 1：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>MyHtml1.html</title>
    <meta http-equiv="content-type" content="text/html ;charset=utf-8">
  </head>

  <body>
    <dl>
      <dt>咖啡</dt>
      <dd>一种黑色的热饮料，原来是咖啡豆，盛产于非洲。</dd>
      <dd>功效是提神、醒脑。</dd>
    </dl>
  </body>
</html>
```

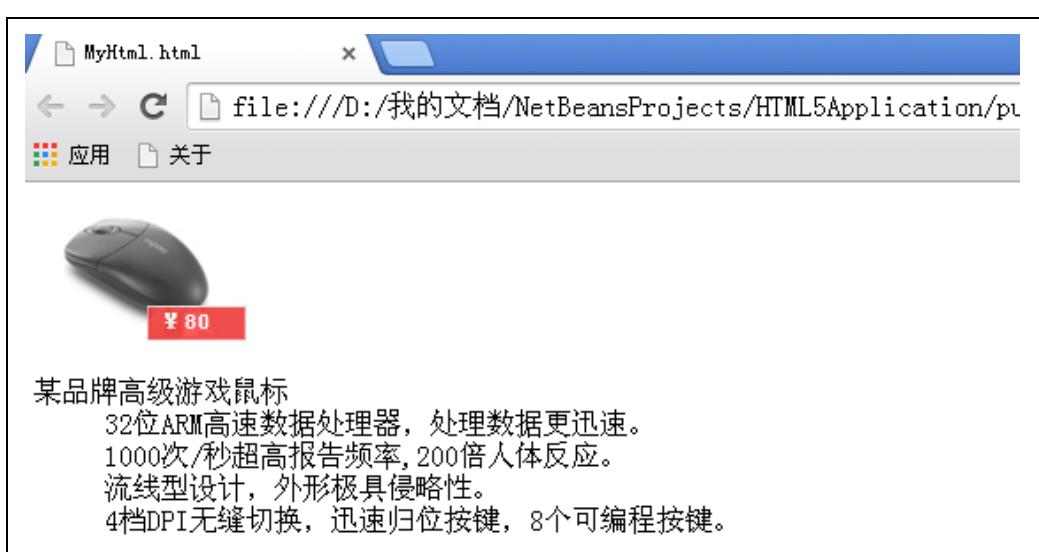
效果:



案例 2：用定义描述标签实现图文混编效果

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ; charset=utf-8">
</head>
<body>
    
    <dl>
        <dt>某品牌高级游戏鼠标</dt>
        <dd>32位ARM高速数据处理器，处理数据更迅速。</dd>
        <dd>1000次/秒超高报告频率，200倍人体反应。</dd>
        <dd>流线型设计，外形极具侵略性。</dd>
        <dd>4档DPI无缝切换，迅速归位按键，8个可编程按键。</dd>
    </dl>
</body>
</html>
```

效果:



2.2.2.4 表格标签

定义: <table>标签定义 HTML 表格。

表格由<table>标签来定义。每个表格均有若干行(由<tr>标签定义),每行被分割为若干单元格(由<td>标签定义)。字母 td 指表格数据 (table data), 即数据单元格的内容。数据单元格可以包含文本、图片、列表、段落、表单、水平线、表格等等。

更复杂的 HTML 表格也可能包括 caption、col、colgroup、thead、tfoot 以及 tbody 元素。

可选的属性

属性	值	描述
align	left center right	不赞成使用。请使用样式代替。 规定表格相对周围元素的对齐方式。
bgcolor	• <i>rgb(x,x,x)</i> • <i>#xxxxxxxx</i> • <i>colorname</i>	不赞成使用。请使用样式代替。 规定表格的背景颜色。
border	<i>pixels</i>	规定表格边框的宽度。
cellpadding	<i>pixels</i> %	规定单元边沿与其内容之间的空白。
cellspacing	<i>pixels</i> %	规定单元格之间的空白。
frame	void above below hsides lhs rhs vsides box border	规定外侧边框的哪个部分是可见的。
rules	none groups rows cols all	规定内侧边框的哪个部分是可见的。
summary	<i>text</i>	规定表格的摘要。
width	% <i>pixels</i>	规定表格的宽度。

语法:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>

<table>      <!-- 表格 -->
<tr>        <!-- 行 -->
<td></td>    <!-- 列 -->
</tr>
<tr>
<td></td>
</tr>
...
</table>
</body>
</html>
```

表格的主要属性:

```
<table border="1" width="40%" cellpadding="0" cellspacing="0">
<tr>
<td>姓名</td>
<td>年龄</td>
</tr>
<tr>
<td>张三</td>
<td>22</td>
</tr>
</table>
```

border: 边框的宽度, 单位是像素 (缺省值是 0)

width: 表格的宽度可以用百分比(表示该表格占父标记的宽度), 也可以是绝对值

cellpadding: 单元格内容下单元格之间的空隙

cellspacing: 单元格下单元格之间的空隙

align: 水平对齐, 值有"center","right","left"

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<table border="1" width="30%" cellspacing="5px" cellpadding="5px">
<tr>
<td>百度</td>
<td>天涯</td>
<td>新浪</td>
</tr>
<tr>
<td>谷歌</td>
<td>搜狐</td>
<td>天猫</td>
</tr>
</table>
</body>
</html>
```

效果：



表格的表头使用<th>标签进行定义。

大多数浏览器会把表头显示为粗体居中的文本：

```
<table border="1">
<tr>
<th>Heading</th>
<th>Another Heading</th>
</tr>
```

```
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

在浏览器显示如下：

Heading	Another Heading
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

在一些浏览器中，没有内容的表格单元显示得不太好。如果某个单元格是空的（没有内容），浏览器可能无法显示出这个单元格的边框。

```
<table border="1">
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td></td>
<td>row 2, cell 2</td>
</tr>
</table>
```

浏览器可能会这样显示：

row 1, cell 1	row 1, cell 2
	row 2, cell 2

注意：这个空的单元格的边框没有被显示出来。为了避免这种情况，在空单元格中添加一个空格占位符，就可以将边框显示出来。

```
<table border="1">
```

```
<tr>  
<td>row 1, cell 1</td>  
<td>row 1, cell 2</td>  
</tr>  
  
<tr>  
<td>&nbsp;</td>  
<td>row 2, cell 2</td>  
</tr>  
</table>
```

在浏览器中显示如下：

```
row 1, cell 1  row 1, cell 2  
              row 2, cell 2
```

如果要实现以下效果，让一个表格占据多列或者多行需要使用 rowspan 和 colspan 属性

row1	row2
	row4

上面表格中 row1 单元格跨越了两行显示，代码如下

```
<table border="1">  
  
<tr>  
  
<td rowspan=2>row1</td>  
  
<td>row2</td>  
  
</tr>  
  
<tr>  
  
<td>row4</td>  
  
</tr>  
</table>
```

我们使用了 td 标签中的 rowspan (跨行) 属性，该属性的值表示单元格占几行显示，同样，我们需要一个单元格占据几列如下效果：

Row1	
row3	row4

上面表格中 row1 占据了两个单元格，代码如下：

```
<table border="1">  
  
<tr>  
  
<td colspan=2>row1</td>  
  
</tr>  
  
<tr>  
  
<td>row3</td>  
  
<td>row4</td>  
  
</tr>  
  
</table>
```

这里我们使用了 `colspan` (跨列) 属性，该属性指定当前单元格在表格中占据多少列。

2.2.2.5 表单标签

定义：`<form>` 标签用于为用户输入创建 HTML 表单。

表单能够包含 `input` 元素，比如文本字段、复选框、单选框、提交按钮等等。

表单还可以包含 `menus`、`textarea`、`fieldset`、`legend` 和 `label` 元素。

表单用于向服务器传输数据。

注：`form` 元素是块级元素，其前后会产生折行。

语法：

```
<form>
```

```
</form>
```

语法详解：

```
<form action="" method="" enctype="">  
    input 标记  
    非 input 标记  
</form>
```

`action` 属性：表单提交之后由哪一个程序来处理

`method` 属性：表单提交方式

`enctype` 属性：设置表单的 MIME 编码

说明：`form` 一般都是和表单 `table` 一起使用的

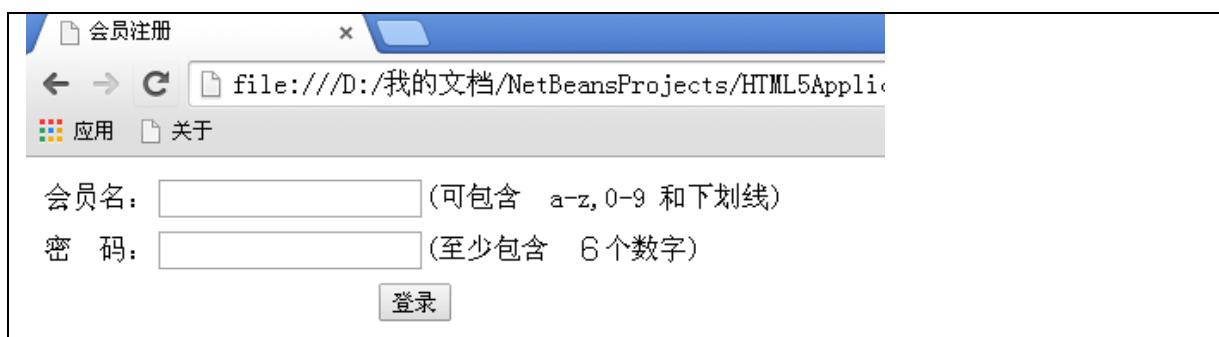
```
<form>  
<table>  
<tr>  
<td></td>  
</tr>
```

```
<tr>
<td></td>
</tr>
...
</table>
</form>
```

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>会员登录页面</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<form action="" method="post">
<table>
<tr>
<td>会员名: <input type="text" />(可包含 a-z, 0-9 和下划线)</td>
</tr>
<tr>
<td>密 码: <input type="password" />(至少包含 6 个数字)</td>
</tr>
<tr>
<td align="left">
<input type="submit" value="登录">
<input type="reset" name="重置"/>
</td>
</tr>
</table>
</form>
</body>
</html>
```

效果：



2.2.2.6 分区 DIV 标签

定义: <div> 英文 (division) 区域的意思。

<div> 标签可以把文档分割为独立的、不同的部分。它可以用作严格的组织工具，并且不使用任何格式与其关联。

如果用 `id` 或 `class` 来标记<div>, 那么该标签的作用会变得更加有效。

<div> 是一个块级元素。这意味着它的内容自动地开始一个新行。实际上，换行是<div>固有的唯一格式表现。可以通过<div>的 `class` 或 `id` 应用额外的样式。

不必为每一个<div>都加上类或 `id`, 虽然这样做也有一定的好处。

可以对同一个<div>元素应用 `class` 或 `id` 属性, 但是更常见的情况是只应用其中一种。这两者的主要差异是, `class` 用于元素组 (类似的元素, 或者可以理解为某一类元素), 而 `id` 用于标识单独的唯一的元素。

注: <div> 是一个块级元素, 也就是说, 浏览器通常会在 `div` 元素前后放置一个换行符。

提示: 请使用<div>元素来组合块级元素, 这样就可以使用样式对它们进行格式化。

语法:

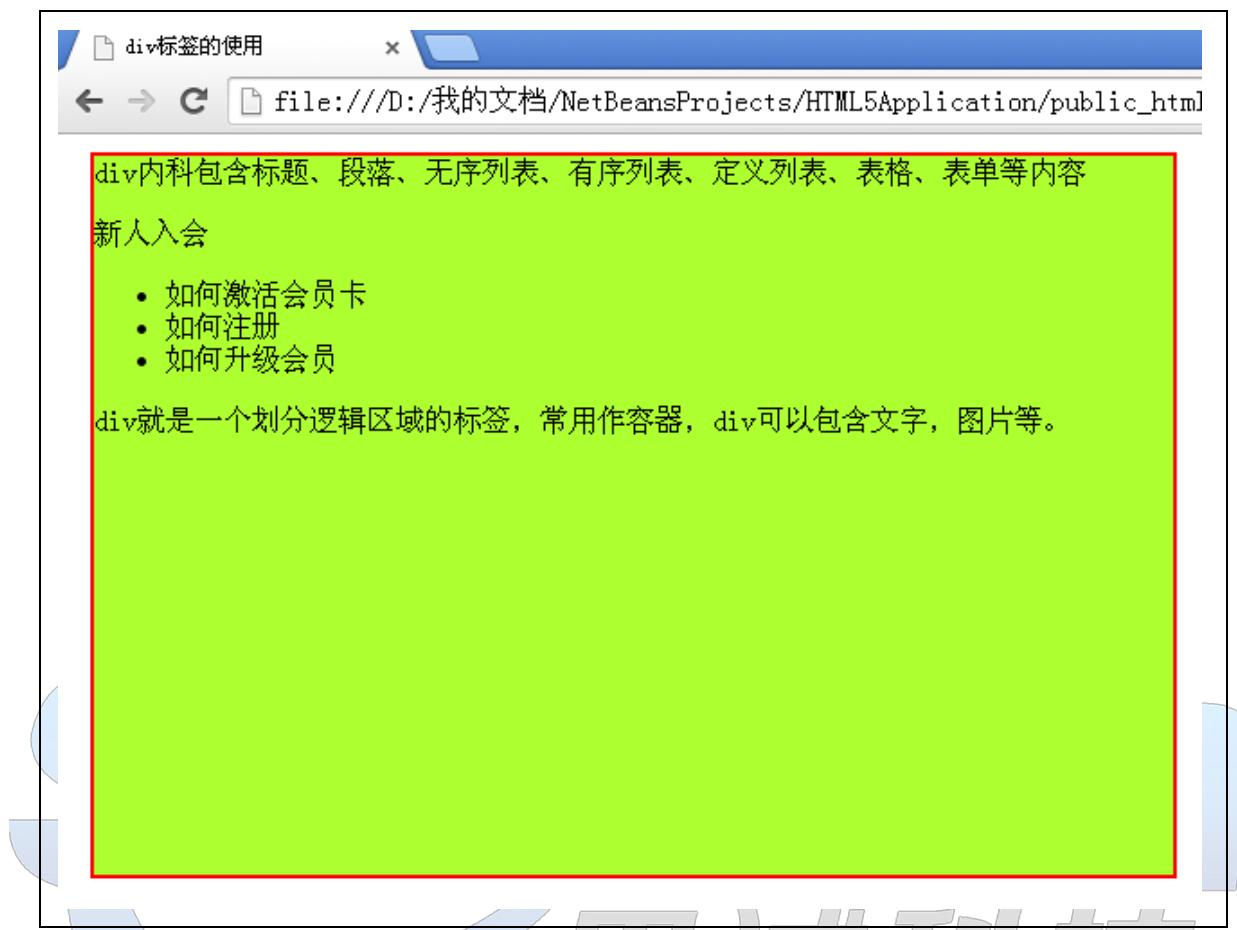
```
<div>  
.....  
</div>
```

案例:



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
<title>div 标签的使用</title>  
<meta http-equiv="content-type" content="text/html ;charset=utf-8">  
</head>  
<body>  
<div style="border: 2px solid red; width: 600px; height: 400px; background-color: greenyellow; margin: 10px auto auto 10px; ">  
    div 内可包含标题、段落、无序列表、有序列表、定义列表、表格、表单等内容  
<p>新人入会</p>  
<ul>  
<li>如何激活会员卡</li>  
<li>如何注册</li>  
<li>如何升级会员</li>  
</ul>  
    div 就是一个划分逻辑区域的标签, 常用作容器, div 可以包含文字, 图片等。  
</div>  
</body>  
</html>
```

效果：



2.2.3 行级标签

行级标签：按行逐一显示

2.2.3.1 图像标签

定义：在 HTML 中，图像由标签定义。

是空标签，意思是说，它只包含属性，并且没有闭合标签。

请注意，从技术上讲，标签并不会在网页中插入图像，而是从网页上链接图像。标签创建的是被引用图像的占位空间。

标签有两个必需的属性：src 属性 和 alt 属性。

必须的属性

属性	值	描述
alt	<i>text</i>	规定图像的替代文本。
src	<i>URL</i>	规定显示图像的 URL。

可选的属性

属性	值	描述
align	<i>top</i> <i>bottom</i> <i>middle</i> <i>left</i> <i>right</i>	不推荐使用。规定如何根据周围的文本来排列图像。
border	<i>pixels</i>	不推荐使用。定义图像周围的边框。
height	<i>pixels</i> %	定义图像的高度。
hspace	<i>pixels</i>	不推荐使用。定义图像左侧和右侧的空白。
ismap	<i>URL</i>	将图像定义为服务器端图像映射。
longdesc	<i>URL</i>	指向包含长的图像描述文档的 URL。
usemap	<i>URL</i>	将图像定义为客户器端图像映射。
vspace	<i>pixels</i>	不推荐使用。定义图像顶部和底部的空白。
width	<i>pixels</i> %	设置图像的宽度。

语法:

<body>

</body>

案例:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>行级标签 img</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>

</body>
</html>
```

效果：



2.2.3.2 范围标签

定义：标签被用来组合文档中的行内元素。

提示：请使用来组合行内元素，以便通过样式来格式化它们。

注：span 没有固定的格式表现。当对它应用样式时，它才会产生视觉上的变化。

语法：
文本等行级内容

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>行级标签 span</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8" >
</head>
<body>

<p>商品价格：仅售<span style=" font-size: 70px;color: red;">1</span>元</p>
</body>
</html>
```

效果：



2.2.3.3 换行标签

定义：
可插入一个简单的换行符。

标签是空标签（意味着它没有结束标签，因此这是错误的：
</br>）。在 XHTML 中，把结束标签放在开始标签中，也就是
。

请注意，
标签只是简单地开始新的一行，而当浏览器遇到 <p>标签时，通常会在相邻的段落之间插入一些垂直的间距。

如果您希望文本流在内联表格或图像的下一行继续输出，请使用 clear 属性，该属性有三个可选的值：left、right 或者 all，每个值都代表一个边界或两边的边界。

注释：请使用
来输入空行，而不是分割段落。

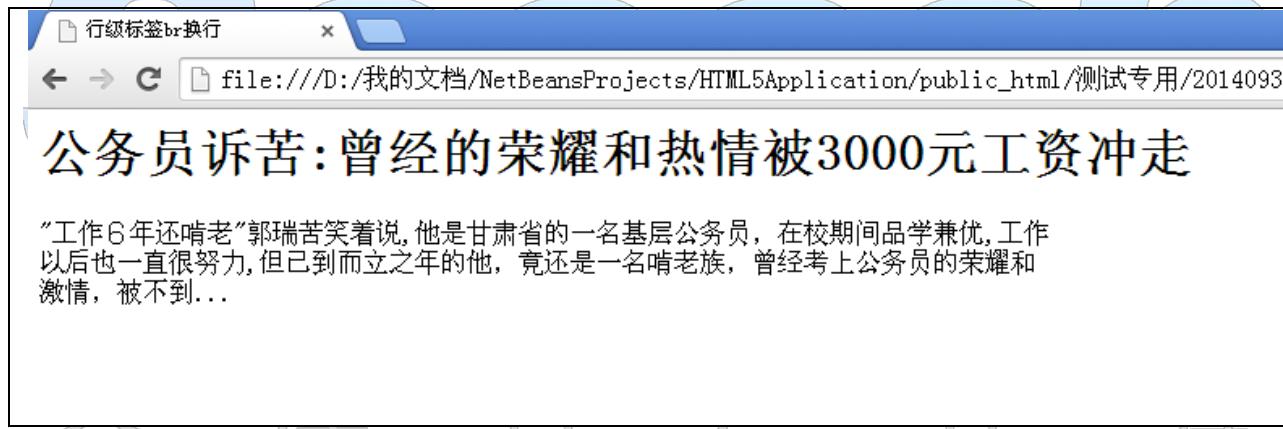
语法：

和<p>的区别：
前后不换行

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>行级标签 br 换行</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<h1>公务员诉苦：曾经的荣耀和热情被 3000 元工资冲走</h1>
"工作 6 年还啃老"郭瑞苦笑着说，他是甘肃省的一名基层公务员，在校期间品学兼优，工作
<br/>
以后也一直很努力，但已到而立之年的他，竟还是一名啃老族，曾经考上公务员的荣耀和<br>
激情，被不到...
</body>
</html>
```

效果：



2.2.4 其他标签

2.2.4.1 超链接标签

定义：超链接可以是一个字，一个词，或者一组词，也可以是一幅图像，您可以点击这些内容来跳转到新的文档或者当前文档中的某个部分。

当您把鼠标指针移动到网页中的某个链接上时，箭头会变为一只小手。

我们通过使用[<a>](#)标签在 HTML 中创建链接。

[<a>](#)标签定义超链接，用于从一张页面链接到另一张页面。

[<a>](#)元素最重要的属性是 href 属性，它指示链接的目标。

在所有浏览器中，链接的默认外观是：

- 未被访问的链接带有下划线而且是蓝色的
- 已被访问的链接带有下划线而且是紫色的
- 活动链接带有下划线而且是红色的

提示：如果不使用 href 属性，则不可以使用如下属性：download, hreflang, media, rel, target 以及 type 属性。

提示：被链接页面通常显示在当前浏览器窗口中，除非您规定了另一个目标（target 属性）。

提示：请使用 CSS 来设置链接的样式。

语法：

```
<a href="链接地址" target="目标窗口位置">链接热点文本或图像</a>
```

说明：地址是相对于当前所在目录的“相对地址”

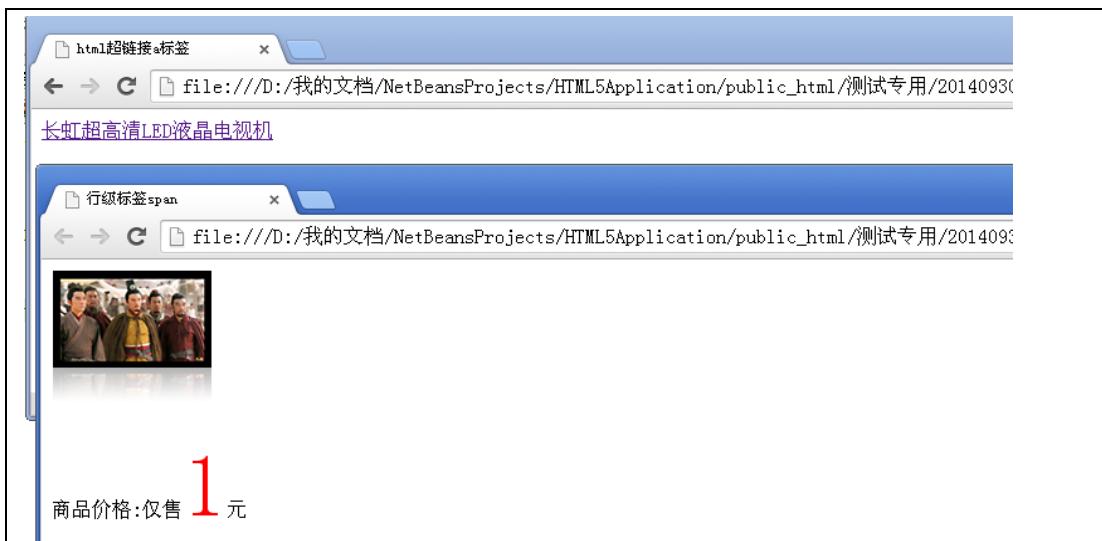
target 属性：指定在哪个窗口打开链接，值可以指定为：

- _self： 在当前窗口中打开（缺省）
- _blank： 新窗口中打开
- title： 提示信息

案例 1：文字链接：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>html 超链接 a 标签</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<a href="span.html" target="_blank">长虹超高清 LED 液晶电视机</a>
</body>
</html>
```

效果：



案例 2：图片链接

解析：

src：对于 img 标签，src 指定图片的地址

width：宽度

height：高度

border：边框（为 0 表示没有边框）

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>html 超链接 a 标签-图片链接</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<a href="span.html" target="_blank"></a>
</body>
</html>
```

超链接的三种形式：

页面间链接 锚链接 功能性链接

运行下面的全部案例之前先编写如下 span.html 文件保存到目录下

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>行级标签 span</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>

<p>商品价格：仅售<span style=" font-size: 70px;color: red;">1</span>元</p>
</body>
</html>
```

页面间链接：

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>html 超链接—页面间链接</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
```

```
<body>
<a href="span.html" target="_blank">长虹超高清 LED 液晶电视机</a>
</body>
</html>
```

效果：



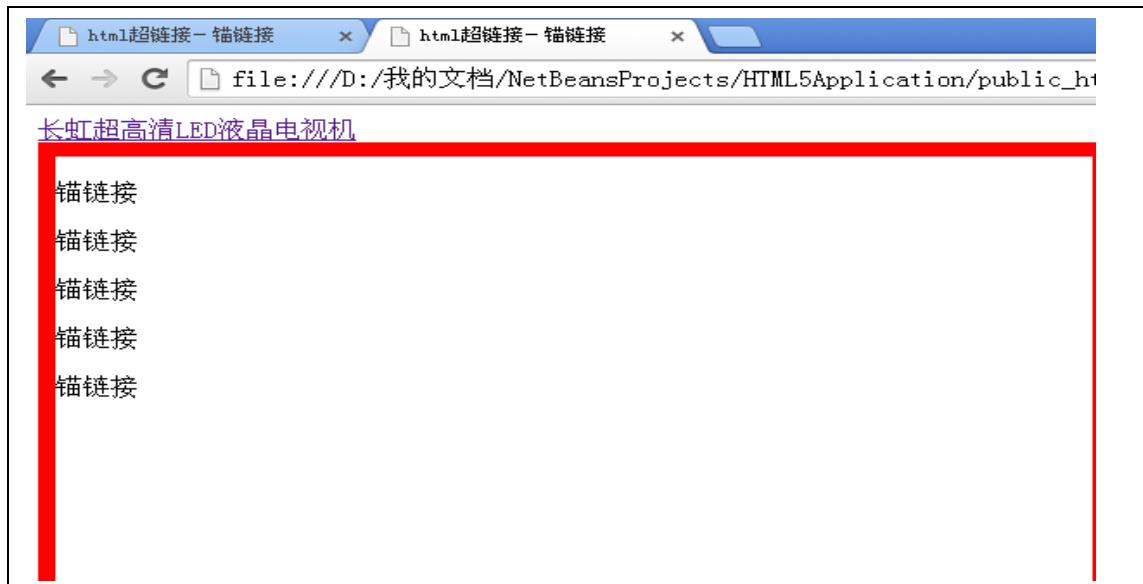
锚链接：

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>html 超链接 - 锚链接</title>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
</head>
<body>
<a href="#television">长虹超高清 LED 液晶电视机</a>
<div style="height: 1000px; width: 600px; border: 10px solid red;">
<p>锚链接</p>
<p>锚链接</p>
<p>锚链接</p>
<p>锚链接</p>
<p>锚链接</p>
</div>

<a name="television">电视机的详细参数列表</a>
</body>
</html>
```

效果:



功能性链接:

案例:



效果:



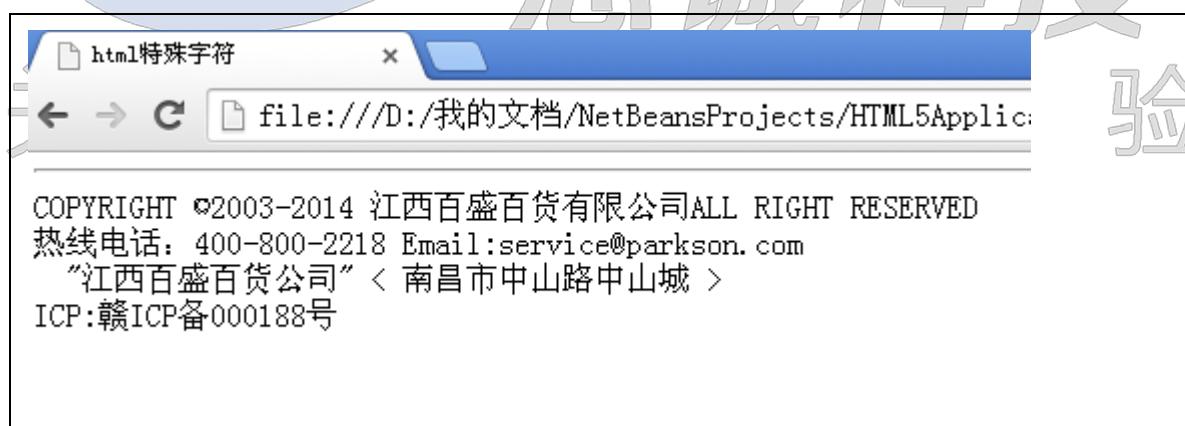
2.2.4.2 特殊符号

- 空格:
- 大于(>): >
- 小于(<): <
- 引号 ("") : "
- 版权号() : ©

案例:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day01_html_Demo16_html 标签特殊符号</title>
</head>
<body>
COPYRIGHT &copy;2003-2014 江西百盛百货有限公司 ALL RIGHT RESERVED<br/>
热线电话: 400-800-2218 Email:service@parkson.com<br/>
&quot;江西百盛百货公司&quot; <南昌市中山路中山城><br/>
ICP:赣 ICP 备 000000 号
</body>
</html>
```

效果:



2.3 本章作业

作业 1：用所学的标签完成下面静态网页的 HTML 编写



唐诗三百首 - Windows Internet Explorer

收藏夹 唐诗三百首

唐诗三百首

目录

第一首：静夜思
第二首：忆江南
第三首：长恨歌

静夜思

作者：李白

床前明月光，疑是地上霜。举头望明月，低头思故乡。



常用的页面布局的块级标签练习 - Windows Internet Explorer

收藏夹 常用于页面布局的块...

商品信息

产品类别

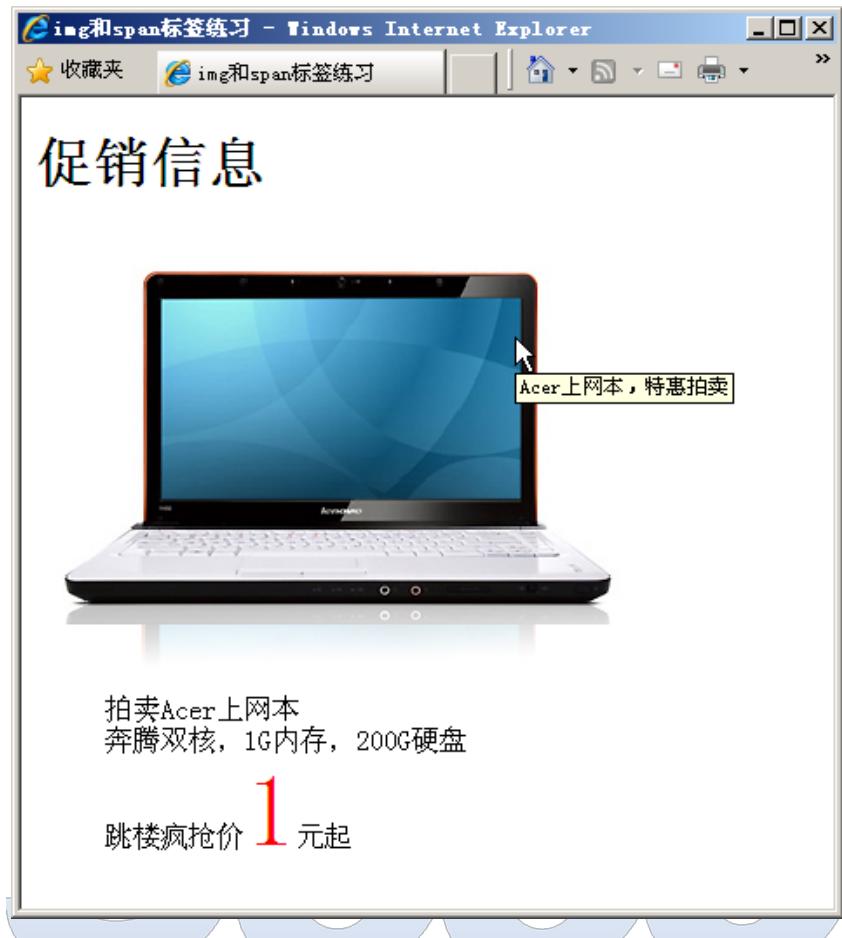
- 数码
 - 笔记本
 - 手机
 - 家电
- 美容
- 服装

联想电脑

产品型号：联想IdeaPad Y450A-TFU(NBA纪念版)
价格：4999元
所在地：北京

购物流程

1. 确认购买信息
2. 付款到贵美
3. 确认收货
4. 付款给商家
5. 双方评价



第 3 章 HTML 布局--表格

3.1 本章目标

使用表格显示数据

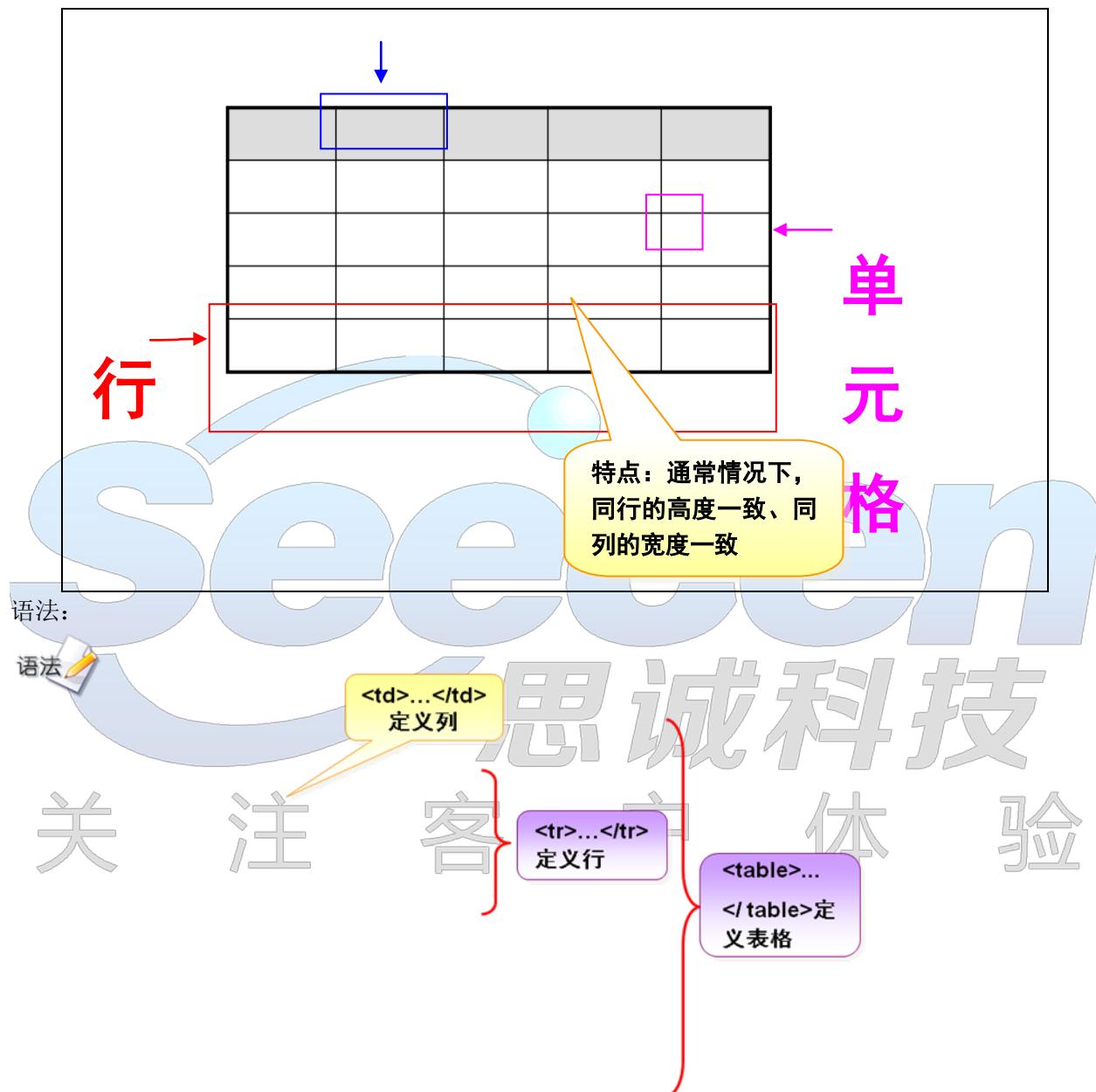
使用表格实现页面布局

思 诚 科 技

注 客 户 体 验

3.2 内容

3.2.1 表格的基本结构



案例:

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta charset="utf-8" />
<title>创建表格</title>
</head>
<body>
<table border="2"><!-- border 表示边框的宽度是 2 -->
<tr><!-- 第一行 -->
<td>1 行 1 列</td>
<td>1 行 2 列</td>
<td>1 行 3 列</td>
</tr>
<tr><!-- 第二行 -->
<td>2 行 1 列</td>
<td>2 行 2 列</td>
<td>2 行 3 列</td>
</tr>
<tr><!-- 第三行 -->
<td>3 行 1 列</td>
<td>3 行 2 列</td>
<td>3 行 3 列</td>
</tr>
</table>
</body>
</html>
```

效果：



1 行 1 列	1 行 2 列	1 行 3 列
2 行 1 列	2 行 2 列	2 行 3 列
3 行 1 列	3 行 2 列	3 行 3 列

3.2.2 表格的跨行跨列

案例：

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>表格跨列</title>
</head>
<body>
<table border="2"><!-- border 表示边框的宽度是 2 -->
<tr>
```

```
<td colspan="3">学生成绩一览表</td><!-- 跨 3 列 -->
</tr>
<tr>
<td rowspan="3">张三</td><!-- 跨了 2 行 -->
<td>语文</td>
<td>98</td>
</tr>
<td>数学</td>
<td>92</td>
</tr>
<td>英语</td>
<td>87</td>
</tr>
</tr>
<tr>
<td rowspan="3">赵五</td>
<td>语文</td>
<td>78</td>
</tr>
<td>数学</td>
<td>62</td>
</tr>
<td>英语</td>
<td>80</td>
</tr>
</tr>
</table>
</body>
</html>
```

效果：

学生成绩一览表		
张三	语文	98
	数学	92
	英语	87
赵五	语文	78
	数学	62
	英语	80

3.2.3 表格的高级标签

案例：

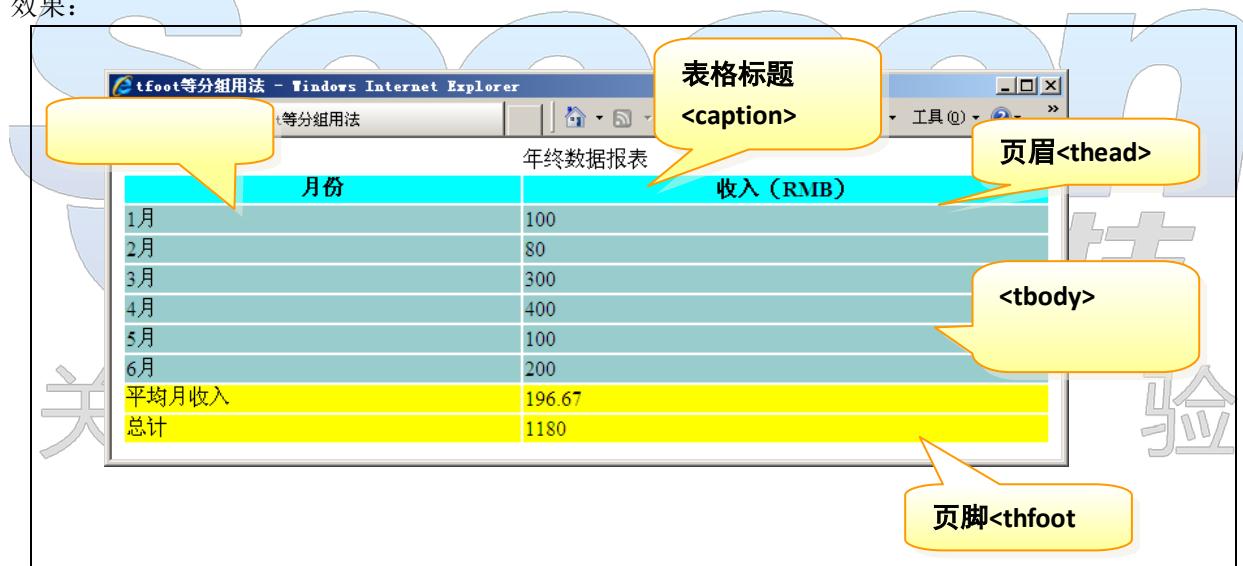
```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>表格跨列</title>
</head>
<body>
<table width="50%" border="2">
<caption>年终数据报表</caption><!-- 定义表格标题 -->
<thead style="background-color: #00ffff"><!-- 定义表格的页眉-->
<tr>
<th>月份</th><!-- 定义表格的表头-->
<th>收入 (RMB)</th>
</tr>
</thead>
<tbody style="background-color: #b77e5e"><!-- 定义表格的主体-->
<tr>
<td>1 月</td>
<td>100</td>
</tr>
<tr>
<td>2 月</td>
<td>80</td>
</tr>
<tr>
<td>3 月</td>
<td>300</td>
</tr>
<tr>
<td>4 月</td>
<td>400</td>
</tr>
<tr>
<td>5 月</td>
<td>100</td>
</tr>
<tr>
<td>6 月</td>
<td>200</td>
```

```
</tr>
</tbody>

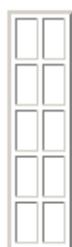
<tfoot style="background-color: yellow;">
<tr>
<td>平均月收入</td>
<td>196.17</td>
</tr>
<tr>
<td>总计</td>
<td>1180</td>
</tr>
</tfoot>
</table>
</body>
</html>
```

效果:



3.2.4 表格实现图文布局

下面是一个案例的分析思路:



①确定行列数：5行2列



②写出5行2列表格

③确定跨行跨列的单元格：

1行1列单元格跨2列

2行1列单元格跨4行

④增加rowspan和colspan属性

删除多余单元格

公告栏		全部分类
	大学要求老师开网店	
	安全锤网上大热销	
	商城竟成网购试衣间	
	2万网上开十间连锁店	

实现代码：

```
.....  
<table border="1px">  
<tr>  
<td colspan="2"></td>  
</tr>  
<tr>  
<td rowspan="4"></td>  
<td>大学要求老师开网店</td>  
</tr>  
<tr>  
<td>安全锤网上大热销</td>  
</tr>  
.....  
</tr>  
</table>  
.....
```

整个是 5 行 2 列的表格，
实际布局时 border="0" 隐

：跨 2 列

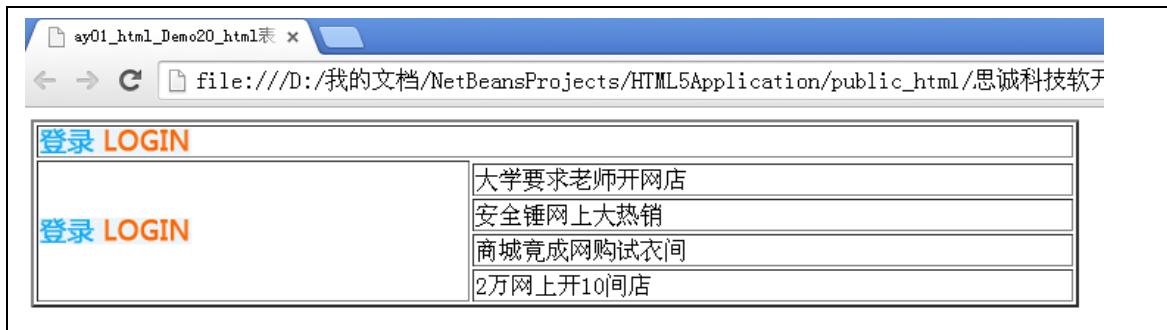
图片：跨 4 行

公告栏		全部分类
	大学要求老师开网店	
	安全锤网上大热销	
	商城竟成网购试衣间	
	2万网上开十间连锁店	

案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>ay01_html_Demo20_html 表格图文布局</title>
</head>
<body>
<table border="2px" width="50%">
<tr>
<td colspan="2"></img></td>
</tr>
<tr>
<td rowspan="5">
</img>
</td>
</tr>
<tr>
<td>大学要求老师开网店</td>
</tr>
<tr>
<td>安全锤网上大热销</td>
</tr>
<tr>
<td>商城竟成网购试衣间</td>
</tr>
<tr>
<td>2万网上开10间店</td>
</tr>
</table>
</body>
</html>
```

效果：



3.2.5 表单实现登录布局

案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day01_html_Demo21_html 表格登录</title>
</head>
<body>
<form action="" method="post">
<table>
<tr>
<td>用户名:<input type="text" name="username" id="username"></input></td>
</tr>
<tr>
<td>密 &nbsp;码:<input type="password" name="passwd" id="passwd"></input></td>
</tr>
<tr>
<td><input type="button" value="登录"></input></td>
</tr>
</table>
</form>
</body>
</html>
```



效果：



3.2.6 表格布局的缺点

表格布局不适合不规则的复杂页面，这种场合需使用 DIV+CSS 布局（后续学习）

3.3 本章作业

1、实现一个跨行跨列的商品类目表格

The screenshot shows a Microsoft Internet Explorer window titled "跨行跨列练习 - Windows Internet Explorer". The page content is titled "商品类目" (Product Categories). It displays a table with two rows. The first row has three columns: "虚拟" (Virtual) with entries "移动" (Mobile), "充值卡" (Recharge Card), and "梦幻" (Fantasy); and "联通" (U Mobile), "彩票" (Lottery), and "QQ". The second row has three columns: "护肤" (Skincare) with entries "美容护肤" (Beauty Skincare), "彩妆" (Cosmetics), and "个人护理" (Personal Care); and "美体" (Fitness), "香水" (Perfume), and "保健" (Healthcare).

虚拟	移动	联通
	充值卡	彩票
	梦幻	QQ
护肤	美容护肤	美体
	彩妆	香水
	个人护理	保健

2、课堂中的案例熟练掌握

第 4 章 表单元素

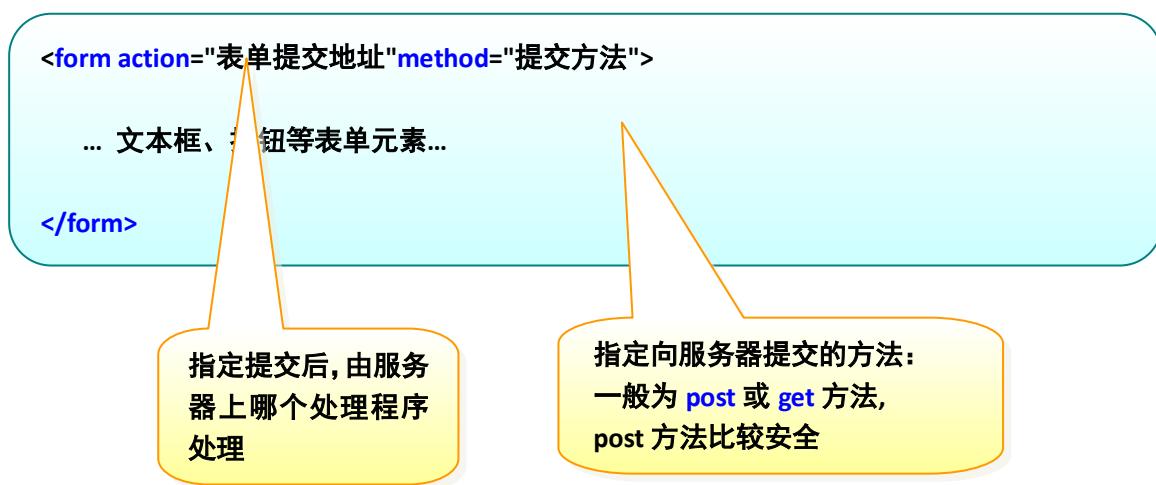
4.1 本章目标

使用超链接实现页面间导航

使用各种表单元素实现注册页面

4.2 内容

表单的基本语法:



案例:

表单提交地址和方法的设置

```
<form action="login.jsp" method="post">  
    <p>用户名: <input name="username" type="text" size="20" maxlength="10" />  
    <p>密 码: <input name="pwd" type="password" size="20" />  
    <p><input type="submit" name="btn" value="提交" />&nbsp;&nbsp;  
    <input name="reset" type="reset" value="重填" />  
</form>
```

表单输入元素: **input**

效果:

关注客户体验



表单元素的基本格式：



定义：表单元素，多数情况下被用到的表单标签是输入标签（`<input>`）。输入类型是由类型属性（`type`）定义的。

`<input>` 标签用于搜集用户信息。

根据不同的 `type` 属性值，输入字段拥有很多种形式。输入字段可以是文本字段、复选框、掩码后的文本控件、单选按钮、按钮等等。

`<input type="" value="" />`

`type` 属性规定 `input` 元素的类型。

属性值

值	描述
<code>button</code>	定义可点击按钮（多数情况下，用于通过 JavaScript 启动脚本）。
<code>checkbox</code>	定义复选框。
<code>file</code>	定义输入字段和“浏览”按钮，供文件上传。
<code>hidden</code>	定义隐藏的输入字段。
<code>image</code>	定义图像形式的提交按钮。
<code>password</code>	定义密码字段。该字段中的字符被掩码。
<code>radio</code>	定义单选按钮。
<code>reset</code>	定义重置按钮。重置按钮会清除表单中的所有数据。



submit	定义提交按钮。提交按钮会把表单数据发送到服务器。
text	定义单行的输入字段，用户可在其中输入文本。默认宽度为 20 个字符。

注：不写 type 时默认为 text

value 属性为 input 元素设定值。

对于不同的输入类型，value 属性的用法也不同：

- type="button", "reset", "submit" - 定义按钮上的显示的文本
- type="text", "password", "hidden" - 定义输入字段的初始值
- type="checkbox", "radio", "image" - 定义与输入相关联的值

注释：<input type="checkbox"> 和 <input type="radio"> 中必须设置 value 属性。

注释：value 属性无法与<input type="file">一同使用。

语法：

```
<input type="text" name="名称" size="数字" value="文本初始值">
```

type 属性：input 标记的具体类型；

 内容可以下写，默认是文本框；

name 属性：标记的一个名称，该名称用于生成一个请求参数；

 如果没有命名，则浏览器下会该数据发送给服务器；

value 属性：缺省值

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>MyHtml.html</title>

<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">

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

</head>

<body>
<form>
<table>
<tr>
```

```
<td>
姓名:
</td>
<td>
<input type="text" name="userName" size="21">
<!-- 指定文本框宽度是 21
    userName 用户服务器端获取数据能有标识
    request.getParameter("userName")
--&gt;
&lt;/td&gt;
&lt;/tr&gt;
&lt;/table&gt;
&lt;/form&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

效果:

姓名:

4. 2. 2 密码框

语法:

```
<input name= "名称" type="password" value="初值" size="数字">
```

案例:

```
<form method="post" action="">
<p>密  &nbsp;&nbsp;码:<br/>
<input name="pass" type="password" size="22" />
</p>
.....
</form>
```

效果:

用户名:

密 码:

4.2.3 按钮框

语法:

```
<input name= "名称" type= "按钮类型" value="按钮文字" src="图片按钮的图片 url">
```

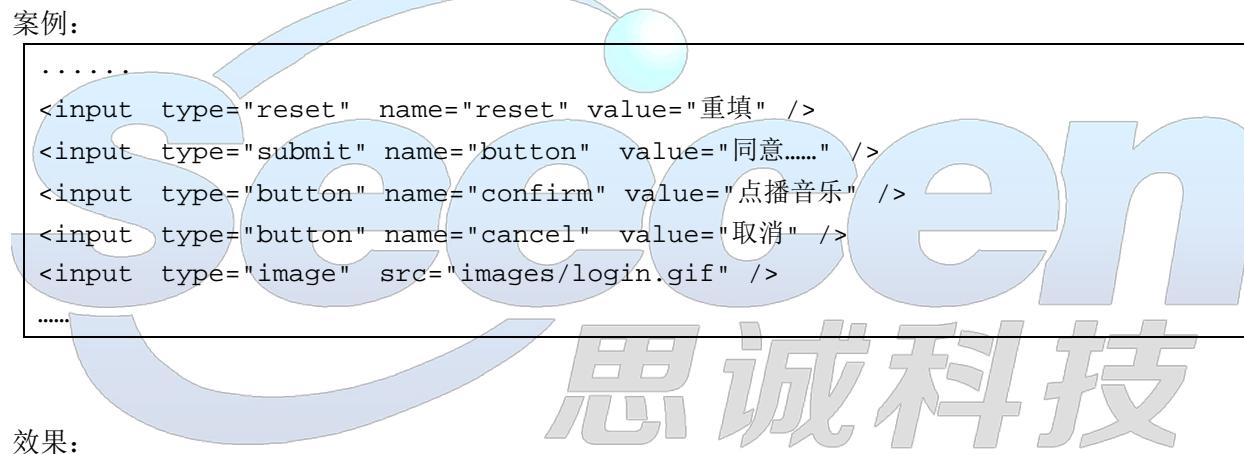
<input name= "名称" type= "按钮类型" value="按钮文字" src="图片按钮的图片 url">

普通按钮: button
提交按钮: submit
重置(清空)按钮: reset
图片按钮: image

案例:

```
.....  
<input type="reset" name="reset" value="重填" />  
<input type="submit" name="button" value="同意....." />  
<input type="button" name="confirm" value="点播音乐" />  
<input type="button" name="cancel" value="取消" />  
<input type="image" src="images/login.gif" />  
....
```

效果:



用户名:

密 码:

4.2.4 单选按钮框

语法:

```
<input type="radio" name="" value="" checked="checked" />
```

备注:

单选按钮应是互斥的，只能选择其中一个

同一组单选按钮: name 必须相同

value属性: 发送给服务器端的值

checked 属性: 就一个值"checked", 表示缺省被选上

案例:

```
....  
<br />性别:  
<input name="gen" type="radio" value="男" checked="checked">  
<input name="gen" type="radio" value="女">  
....
```

效果



案例:

```
....  
爱好:  
<input type="checkbox" name="hobby1" value="sports" />运动  
&nbsp;&nbsp;  
<input type="checkbox" name="hobby2" value="talk" checked="checked" />聊天  
&nbsp;&nbsp;  
<input type="checkbox" name="hobby3" value="play" />玩游戏  
....
```

效果

特点：多选

爱好： 运动 聊天 玩游戏

4.2.6 文件域

语法：

```
<input type="file" name=""/>
```

案例：

```
<html>
<head>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<title>文件域</title>
</head>
<body>
<input type="file" name="files"><br/>
<input type="submit" name="upload" value="上传">
</body>
</html>
```

效果



4.2.7 下拉列表

语法:

```
<select name="指定列表名称" size="行数">  
<option value="选项值" selected="selected">...</option>  
...  
</select>
```

案例:

```
<html>  
  <!--表单的使用-->  
  <head>  
    <meta http-equiv="content-type" content="text/html ;charset=utf-8">  
    <title>select 下拉列表</title>  
  </head>  
  <body>  
    <select name="butn">  
      <option value="0">-请选择月份-</option>  
      <option value="1" selected="selected">星期一</option>  
      <option value="2">星期二</option>  
      <option value="3">星期三</option>  
      <option value="4">星期四</option>  
      <option value="5">星期五</option>  
      <option value="6">星期六</option>  
      <option value="7">星期日</option>  
    </select>  
  </body>  
</html>
```

效果



4.2.8 多行文本框

语法:

```
<textarea name="textarea" cols="列宽" rows="行宽">
```

案例:

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>多行文本框</title>
</head>
<body>
<p><textarea name="textarea" cols="40" rows="6">
欢迎阅读服务条款协议...
</textarea></p>
</body>
</html>
```

效果



4.2.9 隐藏域

语法:

```
<input type="hidden" name="...." value="..." />
```

案例：

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>隐藏域</title>
</head>
<body>
<input type="hidden" name="userId" id="userId" value="88857">
<p><textarea name="textarea" cols="40" rows="6">
欢迎阅读服务条款协议...
</textarea></p>
<script>
    document.write("隐藏域中的值是:");
    "+document.getElementById("userId").value);<!-- 把隐藏的内容打印出来 --&gt;
&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

效果：



4.2.10 readonly and disabled

语法：

readonly: 希望某个框内的内容只允许用户看，不能修改

disabled: 因没达到使用的条件，限制用户使用

案例：只读

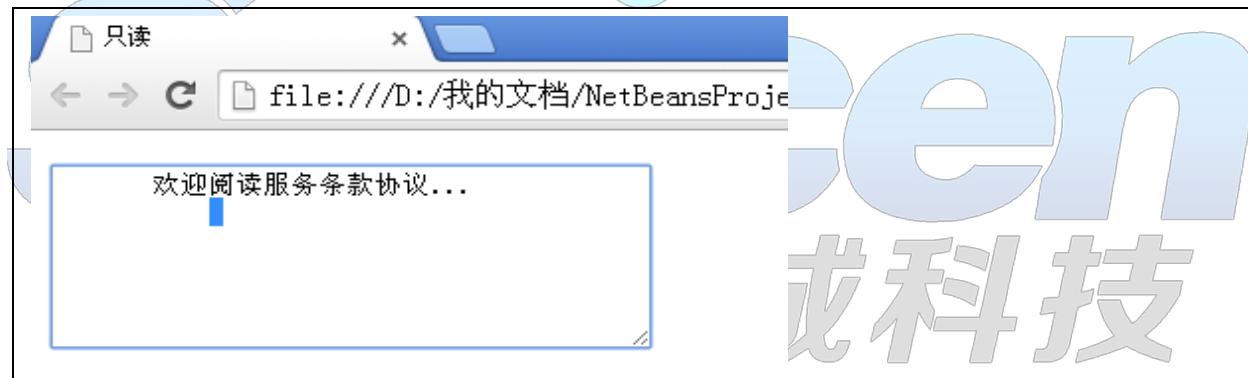
```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>只读</title>
</head>
<body>

<p><textarea name="textarea" cols="40" rows="6" readonly="readonly">欢迎阅读服务条款协议...
</textarea></p>

</body>
</html>
```

效果：



关注客户体验

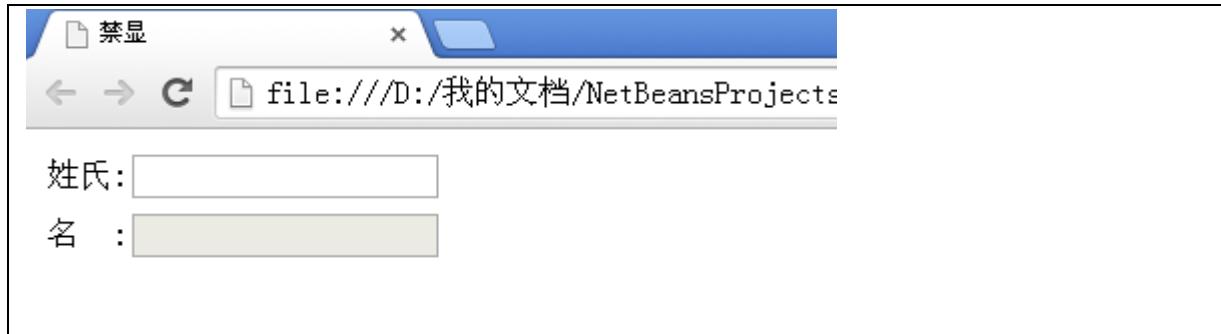
案例：input 按钮的禁止操作

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>禁显</title>
</head>
<body>
<form action="" method="post">
<table>
<tr>
<td>姓氏:<input type="text" name="姓"></td>
</tr>
<tr>
<td>名:<input type="text" name="名" disabled="disabled"></td>
```

```
</tr>
</table>
</form>
</body>
</html>
```

效果：



4.3 本章作业

- 1、编写一个注册页面

作业

file:///D:/我的文档/NetBeansProjects/HTML5Applicat

姓 名: [input]

登录名: [input]

密码: [input]

再次输入密码: [input]

电子邮箱: [input]

性别: 男 女

头像: [input] 未选择任何文件

爱好: 运动 聊天 玩游戏

出生日期: [请选择年] 年 [请选择月] 月 [请选择日] 日

[注册] [重填]

第 5 章 窗口划分 frameset 和 frame

5.1 本章目标

Frameset/frame/iframe 的熟练使用

5.2 内容

通过使用框架，你可以在同一个浏览器窗口中显示不止一个页面

表单元素的基本格式：

5.2.1 frameset

HTML <frameset>标签

定义和用法：

frameset 元素可定义一个框架集。它被用来组织多个窗口（框架）。每个框架存有独立的文档。在其最简单的应用中，frameset 元素仅仅会规定在框架集中存在多少列或多少行。您必须使用 cols 或 rows 属性。

5.2.1.1 frameset 语法：

```
<frameset rows="20%,*">
<frame name="topFrame" src="top.html"/><frameset cols="30,*">
<frame name="leftFrame" src="left.html"/>
<frame name="mainFrame" src="main.html"/></frameset>
</frameset>
```

rows 属性： 将窗口划分成几行

cols 属性： 将窗口划分成几列

frame 标记定义子窗口,其中,src 指定加载的页面

案例：先建立文件 top.html left.html main.html 后运行 html05.html

top.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head></head>
<body style="font-size :30px ;font-style :italic ;">
    top...
</body>
</html>
```

left.html

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
</head>
<body>
<span style="font-size: 40px; font-weight: bold">left</span>
<hr />
<ul>
<li>
<a href="http://www.baidu.com" target="center">百度</a>
</li>
<li>
<a href="http://www.sina.com.cn" target="center">新浪</a>
</li>
<li>
<a href="http://www.qq.com" target="center">腾讯</a>
</li>
</ul>
</body>
</html>
```

main.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head></head>
<body style="font-size: 30px; font-style: italic;">
    main...
</body>
</html>
```

html05.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<!--frameset 的使用-->
<head></head>
<frameset rows="20%, *">
<frame name="topFrame" src="top.html"/><frameset cols="30%, *">
<frame name="leftFrame" src="left.html"/>
<frame name="mainFrame" src="main.html"/></frameset>
</frameset>
</html>
```

效果：



注假如一个框架有可见边框，用户可以拖动边框来改变它的大小。为了避免这种情况发生，可以在`<frame>`标签中加入：`noresize="noresize"`。

为不支持框架的浏览器添加`<noframes>`标签。

重要提示：不能将`<body></body>`标签与`<frameset></frameset>`标签同时使用！不过，假如你添加包含一段文本的`<noframes>`标签，就必须将这段文字嵌套于`<body></body>`标签内。

5.2.2 Iframe

思诚科技

定义：`frame` 元素会创建包含另外一个文档的内联框架（即行内框架）。

属性

属性	值	描述
<code>align</code>	<code>left</code> <code>right</code> <code>top</code> <code>middle</code> <code>bottom</code>	不赞成使用。 请使用样式代替。 规定如何根据周围的元素来对齐此框架。
<code>frameborder</code>	<code>1</code> <code>0</code>	规定是否显示框架周围的边框。
<code>height</code>	<code>pixels</code> <code>%</code>	规定 <code>iframe</code> 的高度。
<code>longdesc</code>	<code>URL</code>	规定一个页面，该页面包含了有关 <code>iframe</code> 的较长描述。
<code>marginheight</code>	<code>pixels</code>	定义 <code>iframe</code> 的顶部和底部的边距。
<code>marginwidth</code>	<code>pixels</code>	定义 <code>iframe</code> 的左侧和右侧的边距。

name	frame_name	规定 iframe 的名称。
scrolling	yes no auto	规定是否在 iframe 中显示滚动条。
src	URL	规定在 iframe 中显示的文档的 URL。
width	pixels %	定义 iframe 的宽度。

5.2.2.1 语法:

```
<iframe src="" width="" height=""></iframe>
```

案例：



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<!--iframe 的使用-->
<head>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body style="font-size :30px ;font-style :italic ;">
你好，世界<br />
<iframe src="html105.html" width="300" height="300"></iframe>
一会儿就要下课了。
</body>
</html>
```

效果：



5.2.3 Frame

定义: <frame>标签定义 frameset 中的一个特定的窗口（框架）。

frameset 中的每个框架都可以设置不同的属性，比如 border、scrolling、noresize 等等。

重要事项: 您不能与<frameset></frameset>标签一起使用<body></body>标签。不过，如果您需要为不支持框架的浏览器添加一个<noframes>标签，请务必将此标签放置在<body></body>标签中！

属性

属性	值	描述
frameborder	0 1	规定是否显示框架周围的边框。
longdesc	URL	规定一个包含有关框架内容的长描述的页面。
marginheight	pixels	定义框架的上方和下方的边距。
marginwidth	pixels	定义框架的左侧和右侧的边距。
name	name	规定框架的名称。
noresize	noresize	规定无法调整框架的大小。
scrolling	yes no auto	规定是否在框架中显示滚动条。
src	URL	规定在框架中显示的文档的 URL。

关注客户体验

5.2.3.1 语法:

```
<html>  
  
<frameset cols="25%,50%,25%">  
  <frame src="frame_a.htm" />  
  <frame src="frame_b.htm" />  
  <frame src="frame_c.htm" />  
</frameset>  
  
</html>
```

效果：

这是A	这是B	这是C
-----	-----	-----

5.3 本章作业

1、熟练掌握课堂中的案例，敲 3 遍

第 6 章 CSS 级联样式表

6.1 本章目标

6.2 内容

6.2.1 CSS 概念及语法

1) CSS 即 cascading stylesheet (层叠样式表)，是用于(增强)控制网页样式并允许将样式信息与网页内容分离的一种标记性语言

2) CSS 为网页提供表现的形式，即按照 w3c 的建议，实现一个比较好的网页设计，应该按照如下的规则：

网页的结构不数据应该写在.html 文件里，网页的表现形式应该写在.css 文件里，网页的行为应该写在.js 文件里。这样做的原因是，将网页的数据、表现、行为分离，方便代码的维护

CSS 的基本语法和定义：

CSS 的规则由两个主要的部分构成：选择器，以及一条或者多条声明；

选择器通常是你需要改变的 HTML 元素，每条声明由一个属性和一个值组成，属性(property)是您希望设置的样式属性，每个属性都有一个值，属性和值被冒号分开。

提示：请使用花括号来包围声明。

```
<head>
    <style type="text/css">
        选择器{
            对象的属性 1: 属性值 1;
            对象的属性 2: 属性值 2;
            .....
        }
    </style>
</head>
```

```
</style>
</head>
示例:
li{color:red;
  font-size:12px;
  font-family:宋体;
}
```

上面这个例子中 li 是选择器，指当前 HTML 所有的 li 元素，color, font-size, font-family 是属性，red, 12px, 宋体是值，这段代码的意思是把当前 HTML 所有的 li 元素里的文本设为红色，字体大小 12 像素，字体为宋体。

6.2.2 CSS 的引用方式

定义：css 有三种引用方式：内嵌、内联、外联

- 1、内嵌样式：直接将样式写在标签中
- 2、内联样式：直接引用，将一组表示 CSS 的字符串作为 style 属性定义至标签内部
当需要定义一个单独的元素样式时，使用
- 3、外联样式：将定义外观的 CSS 样式写在 html 元素标签的外部，然后通过特定的选择器指向该 CSS 样式定义，以达到给 html 标签定义样式的效果。

6.2.3 CSS 选择器

选择器定义：选择器定义了如何查找 html 标记，浏览器会依据选择器，找到匹配的标记，然后施加对应的样式

常用的选择器：标签/元素选择器、类型选择器、ID 选择器、伪类选择器、属性选择器

6.2.3.1 标签选择器

定义：标签选择器指选择器以 HTML 标签元素命名。

6.2.3.1.1 案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day02_html_Demo04_标签选择器.html</title>
<style type="text/css">
  li{
    color:red;
    font-size:28px;
```

```
        font-family: 隶书;
    }
</style>
</head>
<body>
<div>
<ul>
<li>家用电器</li>
<li>各类书籍</li>
<li>手机数码</li>
<li>日用百货</li>
</ul>
</div>
</body>
</html>
```

效果：



6.2.3.2 类选择器(class)

定义：类选择器可以为标有特定 class 的 HTML 元素指定特定的样式。

类选择器以“.”前缀加自定义类名来定义。

语法：.类名（类名自定义） { 属性:属性值 }

}

6.2.3.2.1 案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dt
d">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day02_html_Demo05_类选择器</title>
<style type="text/css">
    .blue{color:blue;}
</style>
</head>
<body>
<ul>
<li class="blue">家用电器</li>
<li>各类书籍</li>
<li class="blue">手机数码</li>
<li>日用百货</li>
</ul>
</body>
</html>
```

上面代码 CSS 中：.blue 是类选择器，color 是属性，blue 是值。代码意思指 html 中有标签 class(类)为 blue 的元素，在当前例子中 class 为 blue 的元素有两个 li，把家用电器和手机数码两个 li 字体颜色变成蓝色。

效果：



6.2.3.3 ID 选择器 (ID)

定义：id 选择器可以为标有特定 id 的 HTML 元素指定特定的样式。

id 选择器以“#”来定义。

语法：#id 名 (id 名自定义) { 属性:属性值}

6.2.3.3.1 案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>大洋百货</title>

<style>
    #menu{
        width:200px; background:#ccc;
        font:bold 14px 宋体;
    }
</style>
</head>
<body>
<div id="menu">
<ul>
<li>家用电器</li>
<li>各类书籍</li>
<li>手机数码</li>
<li>日用百货</li>
</ul>
</div>
</body>
</html>
```

上面代码中：在外部 CSS 样式中，定义了一个 id 选择器名为 menu，width(宽度)为属性，属性值 200px，background(背景)为属性，属性值#ccc 指颜色为灰色，font (字体) 为属性，属性值字体加粗，字体大小 14px 字体为宋体。代码意思指把 html 中标签 id 为 menu，当前为一个 div，定义 div 宽度 200px，背景颜色灰色，div 中所有字体加粗，大小 14 像素，字体为宋体。



6.2.3.4 选择器案例

6.2.3.4.1 案例 1：选择器（多种选择器）-思诚商品分类

```
<html>
```

```
<head>
<title>CSS 级联样式演练</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style>
    #menu{width:200px;background:#ccc;}
    li{font-size:12px;color:#636362;}
    .menu-class{font:bold 14px 宋体;color:#ff7300;}
</style>
</head>
<body>
<div id="menu">
<ul>
<li class="menu-class">家用电器</li>
<li>大家电</li>
<li>洗衣机</li>
<li>电冰箱</li>
<li>平板电视</li>
</ul>
<ul>
<li class="menu-class">日用百货</li>
<li>肥皂</li>
<li>洗衣粉</li>
<li>纸巾</li>
<li>洗发水</li>
</ul>
</div>
</body>
</html>
```



效果：

- **家用电器**
 - 大家电
 - 洗衣机
 - 电冰箱
 - 平板电视

- **日用百货**
 - 肥皂
 - 洗衣粉
 - 纸巾
 - 洗发水

6.2.3.4.2 案例 2：选择器（多种选择器）-修饰思诚商品分类

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict
//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>商城</title>
<style>
#menu{width:200px;height:280px;background:url(images/bg.gif) no-repeat;}
li{font-size:12px;list-style:none;line-height:25px;}
.menu-class{font:bold 14px 宋体;color:#ff7300;line-height:35px;}
.menu-li{color:#636362;float:left;width:50px;}
.space{height:15px;}
p{letter-spacing:5px;text-decoration:underline;}
</style>
</head>
<body>
<p>你好，欢迎访问思诚商城！</p>
<div id="menu">
<ul class="space">
<li></li>
</ul>
<ul>
<li class="menu-class">家用电器</li>
<li class="menu-li">大家电</li>
<li class="menu-li">洗衣机</li>
<li class="menu-li">电冰箱</li>
<li class="menu-li">相机</li>
<li class="menu-li">电视</li>
<li class="menu-li">DVD</li>
<li class="menu-li">相机</li>
<li class="menu-li">电视</li>
<li class="menu-li">DVD</li>
</ul>
<ul>
<li class="menu-class">日用百货</li>
<li class="menu-li">肥皂</li>
<li class="menu-li">洗衣粉</li>
<li class="menu-li">纸巾</li>
<li class="menu-li">洗发水</li>
<li class="menu-li">洁精</li>
<li class="menu-li">毛巾</li>
<li class="menu-li">相机</li>
<li class="menu-li">电视</li>
<li class="menu-li">DVD</li>
</ul>
</div><!--left-list end-->
```

```
</body>  
</html>
```

效果：



案例 3：背景偏移圆角矩形

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<meta http-equiv="Refresh" content="1" />
<title>图像偏移量技术实现圆角矩形</title>
<style>
li{
    width:199px;
    font:12px/20px 宋体;
    list-style:none;text-align:center;
    background:url(images/icon.gif) no-repeat 0px -80px;
}
.title{font:bold 14px/38px 黑体;color:#ff7300;
    background:url(images/icon.gif) no-repeat 0px -50px;
}
.foot{height:14px;background:url(images/icon.gif) no-repeat 0px -100px;}
```

```
</style>
</head>
<body>
<div>
<ul>
<li class="title">商品分类</li>
<li>分类名 1</li>
<li>分类名 2</li>
<li>分类名 3</li>
<li>分类名 4</li>
<li class="foot"></li>
</ul>
</div>
</body>
</html>
```

效果：



6.2.4 常用的 CSS 属性讲解

6.2.4.1 定义边框的四要素：

定义：在 HTML 中，我们使用表格来创建文本周围的边框，但是通过使用 CSS 边框属性，我们可以创建出效果出色的边框，并且可以应用于任何元素。

元素外边距内就是元素的的边框 (border)。元素的边框就是围绕元素内容和内边据的一条或多条线。

每个边框有 3 个方面：宽度、样式，以及颜色。

线条宽度 border-width

线条类型 border-style

线条颜色 border-color

border: width style color

CSS 边框属性

属性	描述
border	简写属性，用于把针对四个边的属性设置在一个声明。
border-style	用于设置元素所有边框的样式，或者单独地为各边设置边框样式。
border-width	简写属性，用于为元素的所有边框设置宽度，或者单独地为各边边框设置宽度。
border-color	简写属性，设置元素的所有边框中可见部分的颜色，或为 4 个边分别设置颜色。
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中。
border-bottom-color	设置元素的下边框的颜色。
border-bottom-style	设置元素的下边框的样式。
border-bottom-width	设置元素的下边框的宽度。
border-left	简写属性，用于把左边框的所有属性设置到一个声明中。
border-left-color	设置元素的左边框的颜色。
border-left-style	设置元素的左边框的样式。
border-left-width	设置元素的左边框的宽度。
border-right	简写属性，用于把右边框的所有属性设置到一个声明中。
border-right-color	设置元素的右边框的颜色。
border-right-style	设置元素的右边框的样式。
border-right-width	设置元素的右边框的宽度。
border-top	简写属性，用于把上边框的所有属性设置到一个声明中。
border-top-color	设置元素的上边框的颜色。
border-top-style	设置元素的上边框的样式。
border-top-width	设置元素的上边框的宽度。

6.2.4.2 文本控制

定义：css 文本属性可定义文本的外观。通过文本属性，您可以改变文本的颜色，字符间距，对齐文本，装饰文本，对文本进行缩进，等等。

文本颜色、字体（大小、字体类型、加粗等）

color 文本颜色

font-family

font-size

font-weight
font: font-weight font-size font-family
line-height
text-decoration
text-align

css 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。
text-decoration	向文本添加修饰。
text-indent	缩进元素中文本的首行。
text-shadow	设置文本阴影。CSS2 包含该属性，但是 CSS2.1 没有保留该属性。
text-transform	控制元素中的字母。
unicode-bidi	设置文本方向。
white-space	设置元素中空白的处理方式。
word-spacing	设置字间距。



关注客户体验

6.2.4.3 边距控制

定义：内边距：元素的内边距在边框和内容区之间。控制该区域最简单的属性是 padding 属性。

CSS padding 属性定义元素边框与元素内容之间的空白区域。

CSS padding 属性定义元素的内边距。padding 属性接受长度值或百分比值，但不允许使用负值。

例如，如果您希望所有 h1 元素的各边都有 10 像素的内边距，只需要这样：

```
h1 {padding: 10px;}
```

您还可以按照上、右、下、左的顺序分别设置各边的内边距，各边均可以使用不同的单位或百分比值：

```
h1 {padding: 10px 0.25em 2ex 20%;}
```

css 内边距属性

属性	描述

padding	简写属性。作用是在一个声明中设置元素的所内边距属性。
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。

围绕在元素边框的空白区域是外边距。设置外边距会在元素外创建额外的“空白”。

设置外边距的最简单的方法就是使用 `margin` 属性，这个属性接受任何长度单位、百分数值甚至负值。

设置外边距的最简单的方法就是使用 `margin` 属性。

`margin` 属性接受任何长度单位，可以是像素、英寸、毫米或 `em`。

`margin` 可以设置为 `auto`。更常见的做法是为外边距设置长度值。下面的声明在 `h1` 元素的各个边上设置了 $1/4$ 英寸宽的空白：

```
h1 {margin : 0.25in;}
```

下面的例子为 `h1` 元素的四个边分别定义了不同的外边距，所使用的长度单位是像素 (px)：

```
h1 {margin : 10px 0px 15px 5px;}
```

与内边距的设置相同，这些值的顺序是从上外边距 (top) 开始围着元素顺时针旋转的：

```
margin: top right bottom left
```

另外，还可以为 `margin` 设置一个百分比数值：

```
p {margin : 10%;}
```

百分数是相对于父元素的 `width` 计算的。上面这个例子为 `p` 元素设置的外边距是其父元素的 `width` 的 10%。

`margin` 的默认值是 0，所以如果没有为 `margin` 声明一个值，就不会出现外边距。但是，在实际中，浏览器对许多元素已经提供了预定的样式，外边距也不例外。例如，在支持 CSS 的浏览器中，外边距会在每个段落元素的上面和下面生成“空行”。因此，如果没有为 `p` 元素声明外边距，浏览器可能会自己应用一个外边距。当然，只要你特别作了声明，就会覆盖默认样式。

css 外边距属性

属性	描述
<code>margin</code>	简写属性。在一个声明中设置所有外边距属性。
<code>margin-bottom</code>	设置元素的下外边距。
<code>margin-left</code>	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。

6.2.4.4 背景控制

定义：Css 背景包括：背景图片、背景颜色、背景图片定位方式、背景图片平铺方式、背景尺寸、背景滚动条

background-image
background-color
background-repeat
background-position
background-position-x
background-position-y

6.2.5 常用的 CSS 属性参考字典

6.2.5.1 字体属性

属性	属性含义	属性值
font-family	使用什么字体	所有字体
font-style	字体是否斜体	normal、italic、oblique
font-variant	字体是否用小体大写	normal、small-caps
font-weight	定义字体的粗细	normal、bold、bolder、lithter 等
font-size	定义字体的大小	Absolute-size、relative-size、length、percentage 等

6.2.5.2 颜色和背景属性

属性	属性含义	属性书写格式	属性值
color	定义前景色	例：p {color: red}	颜色
background-color	定义背景色	例：body { background-color: yellow}	颜色
background-image	定义背景图案	例：body { background-image: (.jpg)}	图片路径
background-repeat	背景图案重复方式	例：body { background-repeat: repeat -y}	repeat -x repeat -y no-repeat
background-attachment	设置滚动	例：body { background-attachment: scroll}	scroll fixed
background-position	背景图案的初始化位	例：body	Percentage length、

	置	{ background: url(.jpg) top center}	top left right bottom 等
--	---	-------------------------------------	-------------------------

属性: background 属性值: <background-color>||<background-image>||<background-repeat>||<background-attachment>||<background-position>

使用 background 属性可以一次定义前面讲到的一切有关背景的属性, 包括背景色、背景图案等等。

例如: "background: url (01038.jpg) no-repeat"

6.2.5.3 文本属性

属性	属性含义	属性值
word-spacing	定义各个单词之间的距离	normal<lenght> (必须以长度为单位)
letter-spacing	定义了每个字母之间的间距	normal<lenght> (必须以长度为单位)
text-decoration	定义文字的“装饰”样式	none underline overline line-through blink
vertical-align	定义了元素在垂直方向上的位置	baseline sub super top text-top middle bottom text-bottom <percentage>
text-transform	使文本转换为其他形式	capitalize uppercase lowercase none
text-align	定义了文字的对齐方式	left right center justify
text-indent	定义文本的首行的缩进方式	<lenght> <percentage>
line-height	定义了文本的行高	normal <number> <lenght> <percentage>

/*定义伪类元素 (a :) , 大括号内定义了前景色属性和文本装饰属性, hover 加上'font-size'属性目的是让鼠标激活链接时改变字体*/

```
a: link{color: green;text-decoration: none}  
/*未访问时的状态, 颜色为绿色(green), 文本装饰属性(text-decoration)值为没有(None)//  
a: visited{color: red;text-decoration: none}  
/*访问过的状态, 颜色为红色(red), 文本装饰属性值为没有*/  
a: hover{color: blue;text-decoration: overline;font-size: 20pt}  
/*鼠标激活的状态, 颜色为蓝色(blue), 文本装饰属性值为上划(overline), 字体大小为 20pt*/
```

6.2.5.4 “容器”属性

什么叫“容器”属性呢？CSS 的容器属性包括边距、填充距、边框和宽度、高度、浮动、清除等属性。

1) 边距属性

属性	属性含义	属性值
margin-top	设置顶端边距	length percentage auto
margin-left	设置左边边距	length percentage auto
margin-right	设置右边边距	length percentage auto
margin-bottom	设置底部边距	length percentage auto

和 font 属性一样，表中的四个属性可以用一个属性一次。边距顺序是上、右、下、左。

body{margin: 1em 2em 3em 4em}

/*定义文本的上、右、下、左的边距分别为 1、2、3、4em*/

2) 填充距属性

属性	属性含义	属性值
padding-top	设置顶端填充距	length percentage
padding-left	设置左边填充距	length percentage
padding-right	设置右边填充距	length percentage
padding-bottom	设置底部填充距	length percentage

padding: 1em 2em 3em 4em

3) 边框属性

属性	属性含义	属性值
border-top-width	设置顶端边框宽度	thin medium thick length
border-left-width	设置左边边框宽度	thin medium thick length
border-right-width	设置右边边框宽度	thin medium thick length
border-bottom-width	设置底部边框宽度	thin medium thick length
border-width	一次定义边框宽度	thin medium thick length
border-color	设置边框颜色	color
border-style	设置边框样式	sone dotted dash solid double groove ridge inset outset
border-top	一次定义顶端各种属性	border-top-width border-style color
border-left	一次定义左边各种属性	border-left-width border-style color
border-right	一次定义右边各种属性	border-right-width border-style color
border-bottom	一次定义底部各种属性	border-boottom-width border-style color

4) 图文混排

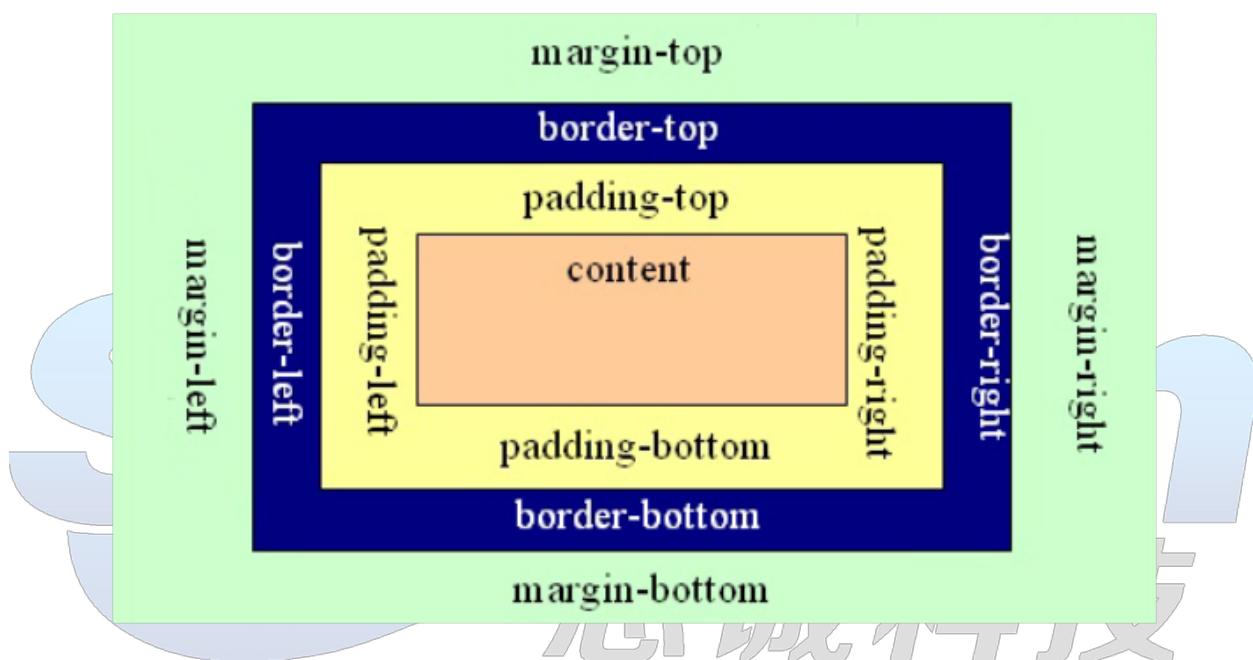
属性	属性含义	属性值
width	定义宽度属性	length percentage auto
height	定义高度属性	length auto

float	使用文字环绕在一个元素的四周	left right none
clear	定义某一边是否有环绕文字	left right none both

6.2.6 盒子模型

CSS 中，Box Model 叫盒子模型（或框模型），Box Model 规定了元素框处理元素内容（element content）、内边距（padding）、边框（border）和外边距（margin）的方式。在 HTML 文档中，每个元素（element）都有盒子模型，所以说在 Web 世界里（特别是页面布局），Box Model 无处不在。

下面是 Box Model 的图示：



说明：

1、上图中，由内而外依次是元素内容(content)、内边距(padding-top、padding-right、padding-bottom、padding-left)、边框(border-top、border-right、border-bottom、border-left)和外边距(margin-top、margin-right、margin-bottom、margin-left)。

2、内边距、边框和外边距可以应用于一个元素的所有边，也可以应用于单独的边。而且，外边距可以是负值，而且在很多情况下都要使用负值的外边距。

3、外边距默认是透明的，因此不会遮挡其后的任何元素（其实元素的 margin 就是其所在父元素的 padding）。元素的背景应用于由内容和内边距、边框组成的区域。

4、内边距、边框和外边距都是可选的，默认值是零。但是，许多元素将由用户代理样式表设置外边距和内边距。可以通过将元素的 margin 和 padding 设置为零来覆盖这些浏览器样式。这可以分别进行，也可以使用通用选择器 (*) 对所有元素进行设置：

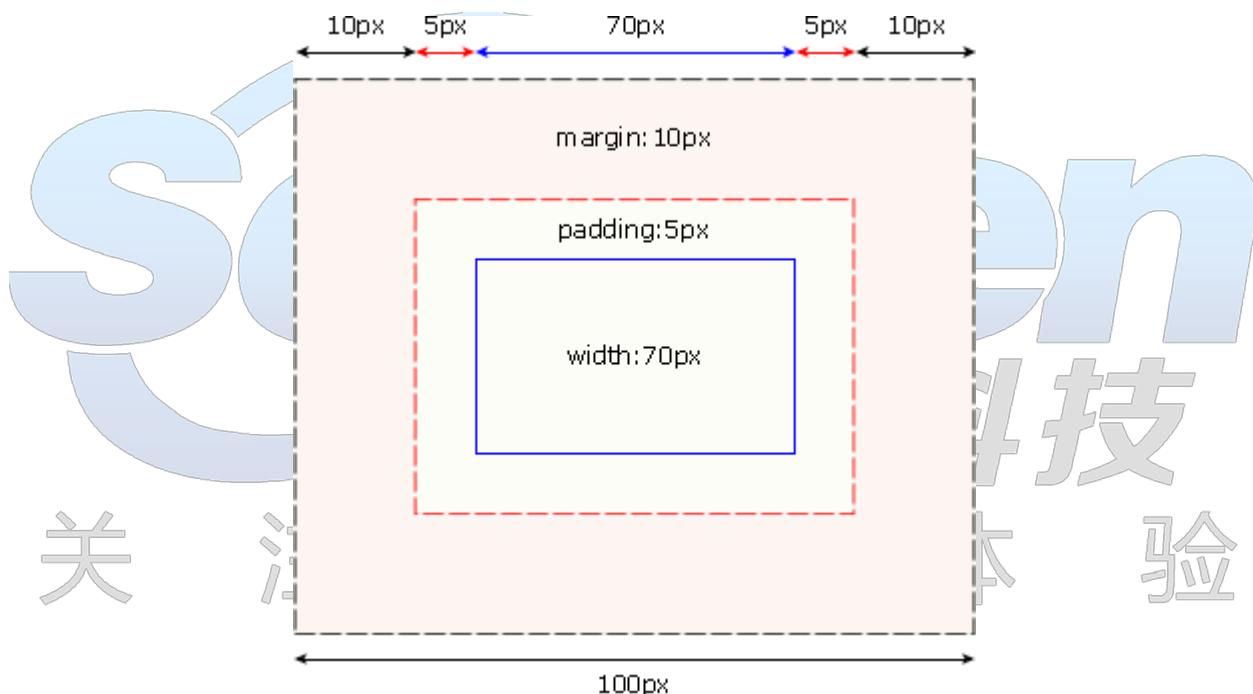
```
/*设置所有元素的外边距和内边距为0*/  
* {  
    margin: 0;
```

```
padding: 0;  
}
```

5、在 CSS 中，width 和 height 指的是内容区域（element）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，但是会增加元素框的总尺寸。假设框的每个边上有 10 个像素的外边距和 5 个像素的内边距。如果希望这个元素框达到 100 个像素，就需要将内容的宽度设置为 70 像素，以下是 CSS 代码：

```
#box {  
    width: 70px;  
    margin: 10px;  
    padding: 5px;  
}
```

下图是对上面 CSS 代码的解释：



<div>标签可以把文档分割为独立的、不同的部分，此元素需要关闭标签；
<div>是一个块级元素，也就是说，浏览器通常会在 div 元素前后放置一个换行符；

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
<title>div 介绍</title>  
<meta http-equiv="content-type" content="text/html ;charset=utf-8">  
</head>  
<body>  
<div style="width: 600px; height: 100px; background-color: red;">
```

```
你好我是第一的 div 块  
</div>  
<div style="width: 700px; height: 200px; background-color: yellow;">  
你好我是第二的 div 块  
</div>  
</body>  
</html>
```

效果：



6.2.7 浮动

语法：

float : none | left | right

取值：

none : 默认值。对象不飘浮

Left : 文本流向对象的右边

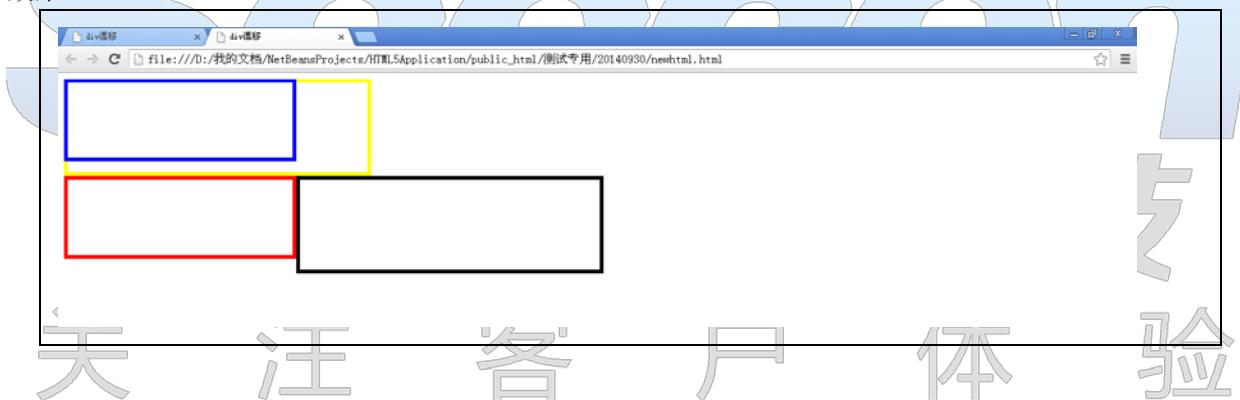
right : 文本流向对象的左边

当该属性不等于 `none` 引起对象浮动时，对象将被视作块对象，跟随浮动对象的对象将移动到浮动对象的位置。浮动对象会向左或向右移动直到遇到边框(`border`、内补丁(`padding`)、外补丁(`margin` 或者另一个块对象(`block-level`)为止

案例：

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>div 浮动</title>
</head>
<body>
<div style="width: 300px; height: 100px; border: 5px solid blue; float: left;">
</div>
<div style="width: 400px; height: 120px; border: 5px solid yellow;">
</div>
<div style="width: 300px; height: 100px; border: 5px solid red; float: left;">
</div>
<div style="width: 400px; height: 120px; border: 5px solid black; float: left;">
</div>
</body>
</html>
```

效果：



6.2.8 定位

position:

- static 默认。无特殊定位，对象遵循 HTML 定位规则
- relative 相对于 static 的位移，但保留原位置的空间占用
- absolute 相对于最近一级设置了 position 的父级元素位移，同时不保留原位置的空间占用
- fixed 相对于浏览器窗口左上角零点坐标的位移，同时不保留原位置的空间占用

案例 1：position: relative absolute

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>day02_html_Demo14_div 定位 position</title>
<style type="text/css">
    .mydiv {
        width: 800px; height: 600px; border: 5px solid black;
        margin: 20px 0 0 100px;
    }
    .div1 {
        width: 200px; height: 80px; border: 5px solid red;
    }
    .div2 {
        width: 240px; height: 120px; border: 5px solid yellow;
        position: relative; left: 50px; top: 50px;
    }
    .div3 {
        width: 280px; height: 200px; border: 5px solid blue;
        position: relative;
    }
    .div4 {
        width: 100px; height: 50px; border: 5px solid #c39025;
    }
    .div3-1 {
        width: 60px; height: 40px; border: 5px solid green;
    }
    .div3-2 {
        width: 80px; height: 50px; border: 5px solid #808080;
        background-color: #808080;
        position: absolute;
        right: -50px; top: -50px;
    }
    /* .div3-3 {
        width: 90px; height: 40px; border: 5px solid red;
    }*/
</style>
</head>
<body>
<div class="mydiv">

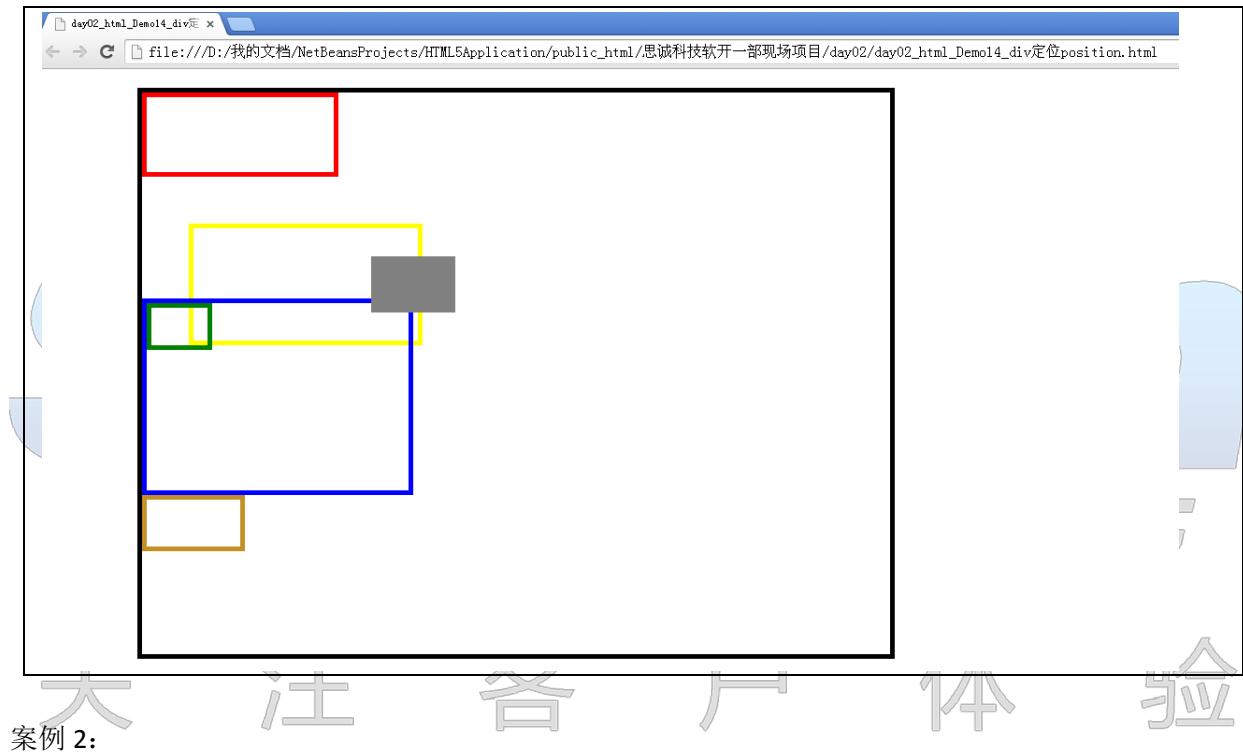
    <div class="div1"></div>
    <div class="div2"></div>

    <div class="div3">
        <div class="div3-1"></div>
        <div class="div3-2"></div>
        <!--<div class="div3-3"></div>-->
    </div>
</div>

```

```
</div>
<div class="div4"></div>
</div>
</body>
</html>
```

效果：



案例 2：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day02_html_Demo15_div 定位 position(fixed)</title>
<style type="text/css">
.mydiv {
    width: 100px; height: 100px;
    border: 5px solid red;
    position: fixed;
    top: 300px; right: 0;
}
</style>
```

```
</head>
<body>
<div class="mydiv">
我是一个静止的 DIV!
</div>

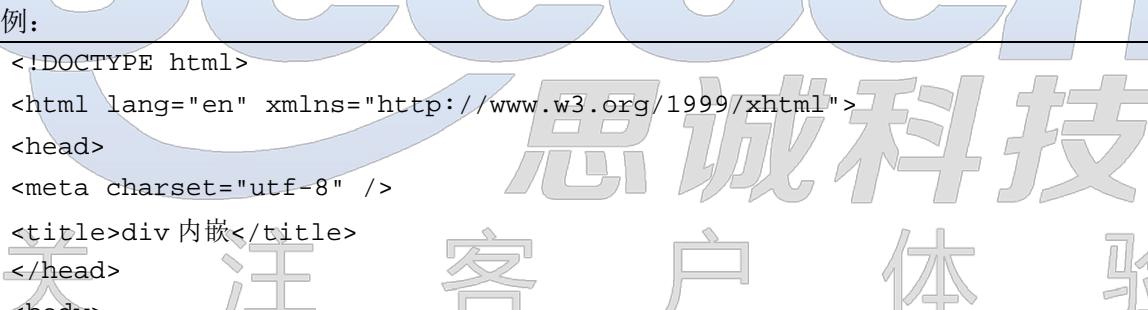
<p>思诚开发者沙龙与本月举行了第五次论坛，期间学员和用人单位进行了交流，并初步达成意向</p>
</body>
</html>
```

效果：



6.2.9 DIV 内嵌

案例：



```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>div 内嵌</title>
</head>
<body>
<div style="width: 660px; height: 360px; border: 5px solid black;">
<div style="width: 660px; height: 30px; border: 5px solid blue; background-color: yellow; padding: 10px;">
    hggjgjh
</div>
<div style="width: 660px; height: 330px; border: 5px solid red;"></div>
</div>
</body>
</html>
```

效果



本章作业

1、熟练掌握课堂中的案例，敲 3 遍

一般现在都使用 div+CSS 布局，以前有很多人用 table。div+CSS 布局的优点是比较方便简洁，代码比较少，制作和维护也比较容易，就是有些地方不同的浏览器兼容性不一样，可能会有不同的显示。table 布局代码比较多，页面嵌套了一层又一层，到后期维护起来是非常麻烦。

关注 客户体验

6.2.10 分级属性

属性	属性含义	属性值
display	定义是否显示	block inline list-item none
white-space	决定怎么处理空白部分	normal pre nowrap
list-style-type	在列表项前加项目编号	disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha none
list-style-image	在列表项前加图案	<url> none
list-style-position	决定列表项中第二行的起始位置	inside outside
list-style	一次定义前面的列表属性	<keyword> <position> <url>

6.2.11 鼠标属性

用 CSS 来改变鼠标的属性，就是当鼠标移动到不同的元素对象上面时，让鼠标以不同的形状、图案显示。

属性值	解释
auto	自动，按照默认状态自行改变
crosshair	精确定位“十”字
default	默认指针
hand	手型
move	移动
e-resize	箭头朝右方
ne-resize	箭头朝右上方
nw-resize	箭头朝左上方
n-resize	箭头朝上方
se-resize	箭头朝右下方
sw-resize	箭头朝左下方
s-resize	箭头朝下方
w-resize	箭头朝左方
text	文本“I”
wait	等待
help	帮助



6.2.12 CSS 应用

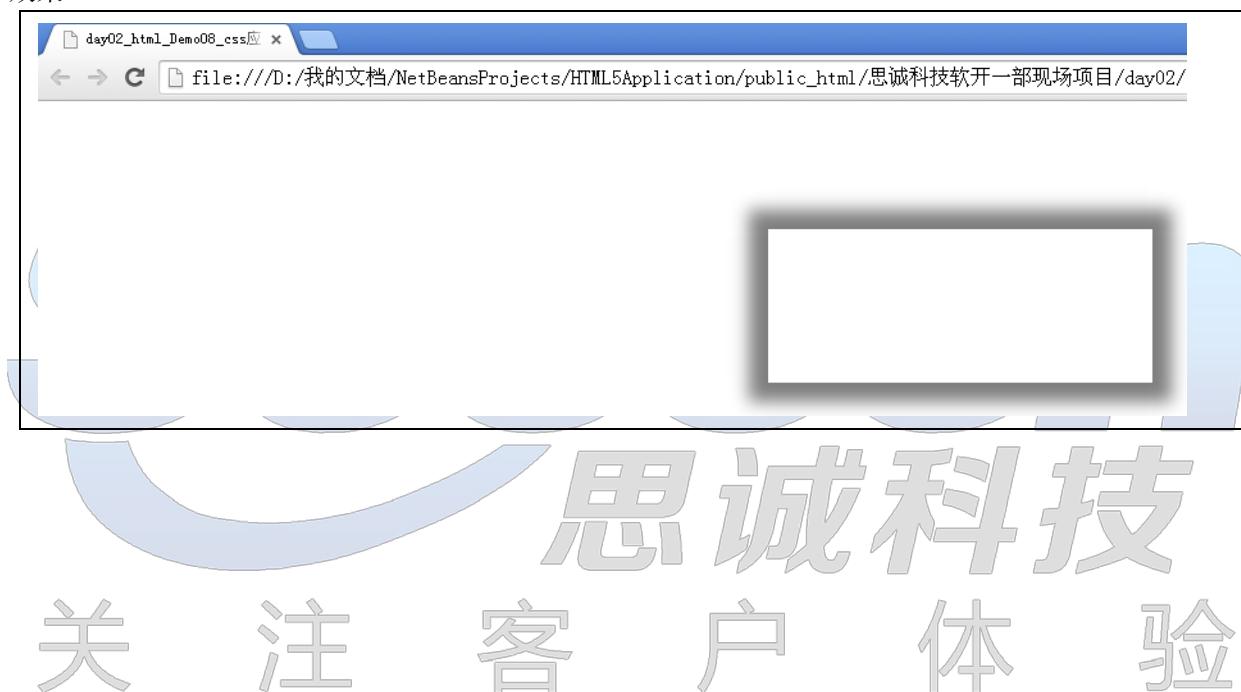
6.2.12.1 案例 1：阴影

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>CSS Box Shadow Demo</title>
<style type="text/css">
```

```
.mydiv {  
    width: 300px; height: 120px;  
    margin: 100px auto 0 auto;  
    box-shadow: #808080 0 0 20px 15px;  
}  
</style>  
</head>  
<body>  
<div class="mydiv"></div>  
</body>  
</html>
```

效果



6.2.12.2 案例 1：应用于表格

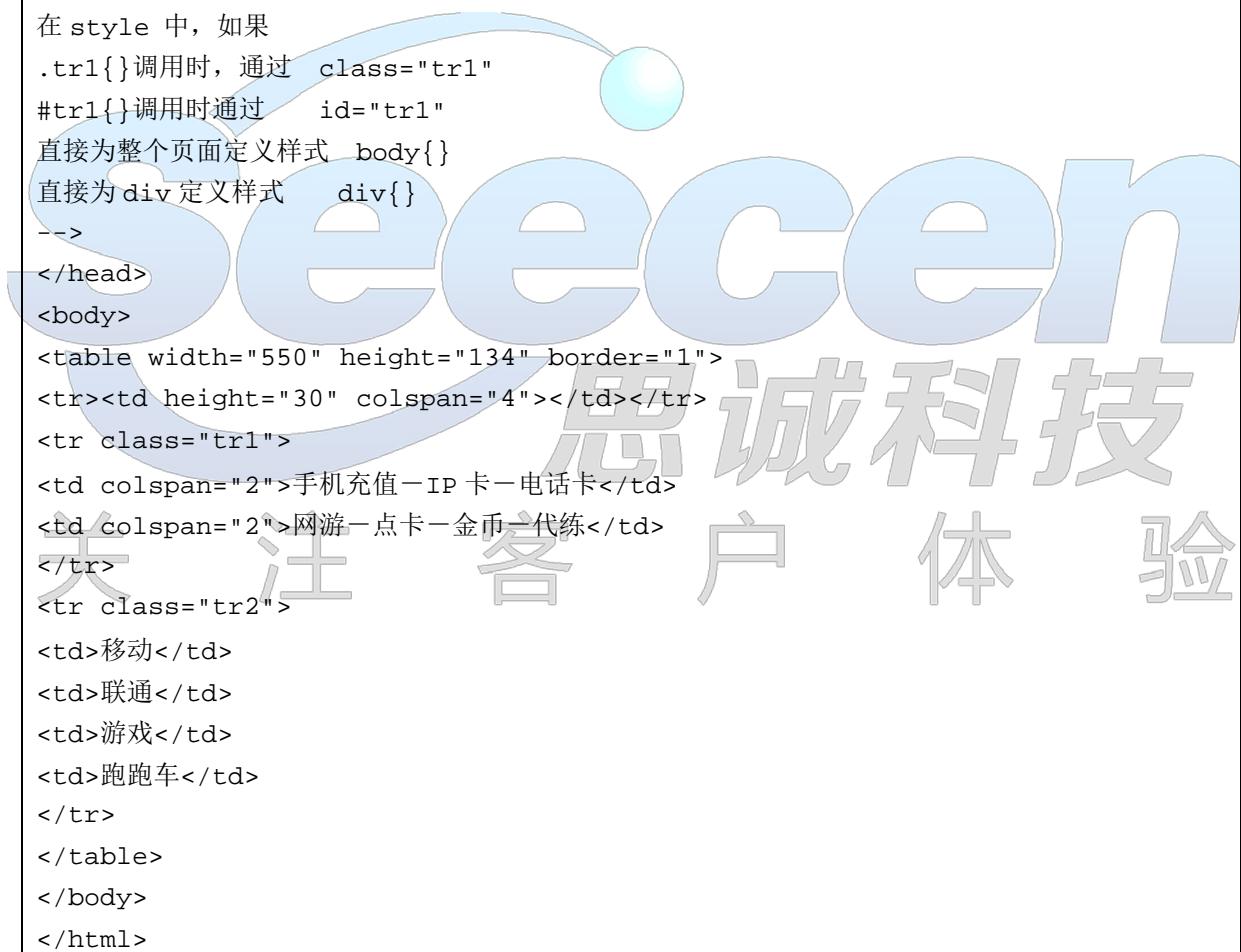
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
<title>游戏卡页面</title>  
  
<meta http-equiv="content-type" content="text/html ;charset=utf-8">  
<meta http-equiv="description" content="this is my page">  
<style>  
    table{background-color: #ecf5fc; background-image:  
url(background.jpg);}  
    .tr1{background-color: #E6F8FF; text-align: center;font-size:  
18px;font-weight: bold; color: blue;}
```

```
.tr2{background-color:#ccffcc;text-align: center;font-size: 14px;color: red}

</style>
<!--

table 表格    width 宽 height 高 border 边框的宽度为 1 像素
样式: 设置内容的位置 大小 颜色
样式写在 style 中, 有些独立的元素, 金额直接放样式 比如 table
background-image:url(a.jpg) 背景图 url 放图的路径
colspan="4" 占 4 列
text-align:center 字体居中
font-size 设置字体大小
font-weight:bold 把字体加粗
color:为字体设置颜色

在 style 中, 如果
.tr1{}调用时, 通过 class="tr1"
#tr1{}调用时通过 id="tr1"
直接为整个页面定义样式 body{}
直接为 div 定义样式 div{}
-->
</head>
<body>
<table width="550" height="134" border="1">
<tr><td height="30" colspan="4"></td></tr>
<tr class="tr1">
<td colspan="2">手机充值—IP 卡—电话卡</td>
<td colspan="2">网游一点卡—金币—代练</td>
</tr>
<tr class="tr2">
<td>移动</td>
<td>联通</td>
<td>游戏</td>
<td>跑跑车</td>
</tr>
</table>
</body>
</html>
```



效果:



6.2.12.3 案例 2：级联菜单

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day02_html_Demo09_css 应用级联菜单</title>
<link type="text/css" rel="stylesheet" href="css/menu-1.css"/>
</head>
<body>
<ul class="menu">
<li>
<a href="#">菜单-1</a>
<ul class="menu-child">
<li><a href="#">子菜单 1-1</a></li>
<li><a href="#">子菜单 1-2</a></li>
<li><a href="#">子菜单 1-3</a></li>
<li><a href="#">子菜单 1-4</a></li>
</ul>
</li>
<li class="separator"></li>
<li>
<a href="#">菜单-2</a>
<ul class="menu-child">
<li><a href="#">子菜单 2-1</a></li>
<li><a href="#">子菜单 2-2</a></li>
<li><a href="#">子菜单 2-3</a></li>
<li><a href="#">子菜单 2-4</a></li>
</ul>
</li>
<li class="separator"></li>
```

```
<li>
<a href="#">菜单-3</a>
<ul class="menu-child">
<li><a href="#">子菜单 3-1</a></li>
<li><a href="#">子菜单 3-2</a></li>
<li><a href="#">子菜单 3-3</a></li>
<li><a href="#">子菜单 3-4</a></li>
</ul>
</li>
<li class="separator"></li>
<li>
<a href="#">菜单-4</a>
<ul class="menu-child">
<li><a href="#">子菜单 4-1</a></li>
<li><a href="#">子菜单 4-2</a></li>
<li><a href="#">子菜单 4-3</a></li>
<li><a href="#">子菜单 4-4</a></li>
</ul>
</li>
<li class="separator"></li>
<li>
<a href="#">菜单-5</a>
<ul class="menu-child">
<li><a href="#">子菜单 5-1</a></li>
<li><a href="#">子菜单 5-2</a></li>
<li><a href="#">子菜单 5-3</a></li>
<li><a href="#">子菜单 5-4</a></li>
</ul>
</li>
<li class="separator"></li>
<li>
<a href="#">菜单-6</a>
<ul class="menu-child">
<li><a href="#">子菜单 6-1</a></li>
<li><a href="#">子菜单 6-2</a></li>
<li><a href="#">子菜单 6-3</a></li>
<li><a href="#">子菜单 6-4</a></li>
</ul>
</li>
</ul>
</body>
</html>
```

效果：



6.2.12.4 案例 3：家电城头部

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<!--<link rel="stylesheet" type="text/css" href="css.css" />-->
<style type="text/css">
    div {
        height: 40px;
        border: 1px solid #125CBB;
        border-width: 1px 0 1px 0;
        background-color: #1369C0;
    }
    li {
        float: left;
        padding: 0 25px 0 25px;
        line-height: 40px;
        list-style-type: none;
    }
    .split {
        margin: 12px 30px 0 30px;
        padding: 0;
        height: 16px;
        border-left: 1px solid #095797;
    }
    .myclass {
        list-style-type: circle;
    }
</style>
```

```
#hot {  
    position: absolute; top: -8px; left: 45px;  
}  
  
</style>  
</head>  
<body>  
<div>  
<ul style="padding: 0; margin: 0;">  
<li><a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">首页</a></li>  
<li class="split myclass"></li>  
<li style="position: relative;">  
  
<a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">家电城</a>  
</li>  
<li><a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">天黑黑</a></li>  
<li><a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">团购</a></li>  
<li class="split"></li>  
<li><a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">发现</a></li>  
<li>  
<a href="#" style="color: white; text-decoration: none; font-size: 14px; font-family: 微软雅黑;">小易说事</a>  
</li>  
</ul>  
</div>  
<hr />  
<div></div>  
</body>  
</html>
```

效果：



6.2.12.5 案例 4: CSS 鼠标事件

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day02_html_Demo11_css 应用鼠标事件</title>
<style type="text/css">
    .myinput {
        width: 300px; height: 40px;
        border: 5px solid red;
    }
    .myinput:hover {
        border-color: green;
    }
    .myinput:focus {
        border-color: blue;
        border-radius: 10px;
    }
}
.myp {
    font-size: 12px; line-height: 20px;
}
.myp:first-line {
    font-weight: bold;
}
.myp:first-letter {
    padding-left: 20px;
}
</style>
</head>
<body>
<input class="myinput" />
<hr />
<p class="myp">
    习近平向马尔代夫人民致以中国人民的诚挚问候和良好祝愿。习近平指出，中马建交 42 年来，两国
</p>
```

关系健康稳定发展，成为大小国家平等相待、和睦相处的典范。前不久，总统先生前往中国南京，出席第二届夏季青年奥林匹克运动会，表达了马尔代夫政府和人民对中国的热情支持。马尔代夫是古代海上丝绸之路的重要一站，同中国历史渊源深厚。中方欢迎马方积极参与 21 世纪海上丝绸之路建设，愿同马方齐心协力，同舟共济，巩固传统友谊，深化互利合作，携手实现共同发展。

```
</p>  
</body>  
</html>
```

效果：



6.2.12.6 案例 5：“设计教程”

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
</head>
<body>
<div style="width: 660px; height: 360px; font-size: 12px; border: 1px solid #808080; padding: 5px; border-radius: 5px;">
<div style="height: 30px; border-bottom: 1px solid #808080; margin-bottom: 10px;">
<h4 style="margin: 0 0 0 5px; height: 30px; line-height: 30px; float: left; background-image: url(images/title.gif); background-repeat: no-repeat; background-position-y: center; padding-left: 15px;">
<!--
设计教程
</h4>
<ul style="float: right; padding: 0; margin: 0; line-height: 30px;">
<li style="float: left; list-style-type: none; padding: 0 5px 0 5px;"><a href="#" style="text-decoration: none;">Photoshop</a></li>
<li style="float: left; list-style-type: none; padding: 0 5px 0 5px;"><a href="#" style="text-decoration: none;">平面设计</a></li>
<li style="float: left; list-style-type: none; padding: 0 5px 0 5px;"><a
```

```
href="#" style="text-decoration: none;">印前技术</a></li>
<li style="float: left; list-style-type: none; padding: 0 5px 0 5px;"><a href="#" style="text-decoration: none;">网页设计</a></li>
<li style="float: left; list-style-type: none; padding: 0 5px 0 5px;"><a href="#" style="text-decoration: none;">多媒体</a></li>
</ul>
</div>
<div style="height: 310px;">
<div style="width: 374px; height: 310px; float: left;">
<ul style="padding: 0; margin: 0;">
<li style="list-style-type: none; height: 90px; margin-bottom: 10px;">

<div style="float: left; width: 235px; margin: 0 0 0 5px;">
<h5 style="font-weight: normal; font-size: 14px; margin: 10px 0 10px 0;">
<a href="#" style="color: black;">用 Photoshop 制作精美光线条</a>
</h5>
<p style="margin: 0;">
使用 Photoshop 最基本的命令，通过简单几步就可以实现精美的光线条。
</p>
</div>
</li>
<li style="list-style-type: none; height: 90px; margin-bottom: 10px;">

<div style="float: left; width: 235px; margin: 0 0 0 5px;">
<h5 style="font-weight: normal; font-size: 14px; margin: 10px 0 10px 0;">
<a href="#" style="color: black;">对面的鸟儿看过来 PS 趣味动画实例</a>
</h5>
<p style="margin: 0;">
最近在网络上看到一些有趣的小鸟动画，真是可爱极了，忍不住用 Photoshop 研究了一下它们的做法，现将过程截...
</p>
</div>
</li>
<li style="list-style-type: none; height: 90px; margin-bottom: 10px;">

<div style="float: left; width: 235px; margin: 0 0 0 5px;">
<h5 style="font-weight: normal; font-size: 14px; margin: 10px 0 10px 0;">
<a href="#" style="color: black;">Photoshop 打造美丽的风中蒲公英</a>
</h5>
<p style="margin: 0;">
虽然她很不起眼，但她又是那么的纯洁，那么的飘逸。当风来的时候，她美丽的身影飘向大地感觉就像天使，那么...
</p>
</div>
</li>
```

效果：



6.2.12.7 案例 6：“六人行”

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title>六人行 - 首页</title>
<style type="text/css">
body {
    padding: 0px;
    margin: 0px;
    font-size: 12px;
}

.wrapper {
    width: 940px;
    /*上右下左*/
    margin: 0 auto 0 auto;
}
.wrapper>div {
    margin: 5px auto 0 auto;
}
.linklist {
    margin: 0px; padding: 0px; display: inline-block;
}
```

```
.linklist li {  
    margin: 0px; padding: 0 8px 0 8px; float: left;  
    list-style-type: none;  
    text-align: center;  
    font-size: 14px; font-weight: bold;  
}  
.linklist li:first-child {  
    padding-left: 0;  
}  
a {  
    text-decoration: none;  
}  
h6 {  
    margin: 3px 0 3px 0; padding: 0; font-size: 12px;  
}  
  
  
.border {  
    border: 1px solid #CCCCCC;  
}  
.infolist {  
    margin: 0px; padding: 0px;  
}  
.infolist li {  
    margin: 10px 0 0 0; padding: 0px;  
    height: 60px;  
    list-style-type: none;  
}  
.infolist li img {  
    width: 50px; height: 50px; float: left; margin-top: 5px;  
}  
.infolist li p {  
    float: left; width: 210px; margin: 0 0 0 10px; text-align: left;  
    line-height: 20px;  
}  
.pline {  
    margin: 6px 0 6px 0; color: #1C6ED1;  
}  
.pline>* {  
    vertical-align: middle;  
}  
.neightlist {  
    margin: 0px; padding: 10px;  
}
```

```
.neightlist li {  
    margin: 0; padding: 0px;  
    height: 70px;  
    list-style: none;  
    float: left;  
}  
.neightlist li img {  
    width: 50px; height: 50px; float: left; margin-top: 5px;  
}  
.neightlist li p {  
    float: left; width: 90px; height: 60px; margin: 0 0 0 10px;  
    text-align: left; line-height: 20px;  
}  
</style>  
</head>  
<body>  
<div class="wrapper">  
<div style="width: 640px; height: 75px;">  
<div style="float: left; width: 180px;">  
  
</div>  
<div style="float: left; padding-top: 5px;">  
<ul class="linklist">  
<li><a href="#">首页</a></li>  
<li>|</li>  
<li><a href="#">活动</a></li>  
<li>|</li>  
<li><a href="#">据点</a></li>  
<li>|</li>  
<li><a href="#">招贴</a></li>  
<li>|</li>  
<li><a href="#">VIP 服务</a></li>  
<li>|</li>  
<li><a href="#">随便逛逛</a></li>  
</ul>  
<div style="padding-top: 5px;">  
<input type="text" style="width: 335px; height: 24px; border: 5px solid blue;  
padding-left: 25px; background-image: url(images/search.png);  
background-repeat: no-repeat; background-position: 5px center;" />  
<input type="button" value="找找!" style="width: 80px; height: 36px;  
background-color: blue; border-style: none; font-size: 16px; font-weight:  
bold; color: white;" />  
</div>  
</div>  
</div>
```

```
<div style="height: 455px;">
<div class="border" style="float: left; width: 578px; height: 433px; padding: 10px;">


<div style="height: 235px; margin-top: 10px;">
<div style="float: left; width: 284px; height: 230px;">
<h6>参加丰富的<span style="color: red;">活动</span></h6>
<ul class="infolist">
<li>

<p>
[培训] <a href="#">11月17号婚恋心理工作坊 (</a><br />
活动介绍婚恋心理工作坊：单身人士进入婚恋关系的十大障碍...
</p>
</li>
<li>

<p>
[演出] <a href="#">天籁之夜---尚雯婕 2009 北(</a><br />
尚雯婕 2009 北京演唱会天籁职业---尚雯婕 2009 巡回演唱会/音乐会
</p>
</li>
<li>

<p>
[休闲户外] <a href="#">爱旅行 11月8周日黄(</a><br />
【行程安排】早 7:50 集合，8:00 出发，集合地：苹果园地铁 D 出口往右物...
</p>
</li>
</ul>
</div>
<div style="float: left; width: 284px; height: 230px;">
<h6>加入身边的<span style="color: red;">据点</span></h6>
<ul class="infolist">
<li>

<p>
剧场<a href="#">梅兰芳大剧场 (</a><br />
拥有者：(价值: 3000) <br />
地址：西城区平安里西大街 32 号
</p>
</li>
<li>

</li>
</ul>
</div>
</div>
```

```
<p>
滑雪场<a href="#">莲花山滑雪场 (</a><br />
拥有者: (价值: 4000) <br />
地址: 顺义区张镇良山东路
</p>
</li>
<li>

<p>
餐厅<a href="#">四川简阳羊肉馆 (</a><br />
拥有者: (价值: 11600) <br />
地址: 朝阳区金台路 9 路总站西 100 米
</p>
</li>
</ul>
</div>
</div>
</div>
<div style="float: left; width: 330px; height: 455px; margin-left: 10px;">
<div class="border" style="height: 178px; padding: 10px; background-color: #ECF5FC;">

<p class="pline">你的手机号: </p>
<p class="pline">
<input type="text" style="width: 240px; height: 15px;" />
</p>
<p class="pline">你的密码: </p>
<p class="pline">
<input type="text" style="width: 240px; height: 15px;" />
</p>
<p class="pline">
<input id="rememberme" type="checkbox" />
<label for="rememberme">在这台电脑上记住我</label>
</p>
<p class="pline">
<button style="margin: 0; padding: 0; border-width: 0;"></button>
<a href="#">忘记密码? </a>
</p>
</div>
<div class="border" style="height: 48px; line-height: 48px; font-size: 16px;
padding-left: 20px; margin-top: 10px; background-color: #ECF5FC;">
还不是六人行会员? <a href="#">马上注册</a>
</div>
<div class="border" style="height: 183px; margin-top: 10px;">
```

```
<h6 style="margin: 0; padding: 0 0 0 10px; height: 25px; line-height: 25px; border-bottom: 1px solid #16B3DC; background: linear-gradient(to bottom, #EFF5FF 0, #E8D6D6 100%);">  
找到身边的<span style="color: red;">邻居</span>  
</h6>  
<ul class="neightlist">  
<li>  
  
<p>  
<a href="#">圣哲西</a><br />  
所在写字楼<br />  
<a href="#">华杰大厦</a>  
</p>  
</li>  
<li>  
  
<p>  
<a href="#">zhangj</a><br />  
所在写字楼<br />  
<a href="#">海龙大厦</a>  
</p>  
</li>  
<li>  
  
<p>  
<a href="#">凤凰花</a><br />  
所在写字楼<br />  
<a href="#">大成国际中心</a>  
</p>  
</li>  
<li>  
  
<p>  
<a href="#">镜花水</a><br />  
所在写字楼<br />  
<a href="#">SOHO 现代城</a>  
</p>  
</li>  
</ul>  
</div>  
</div>  
</div>  
</body>  
</html>
```

效果：



6.3 本章作业

1、网站页面编写

2、课堂中的内容课后练习

第 7 章 JavaScript 语言基础

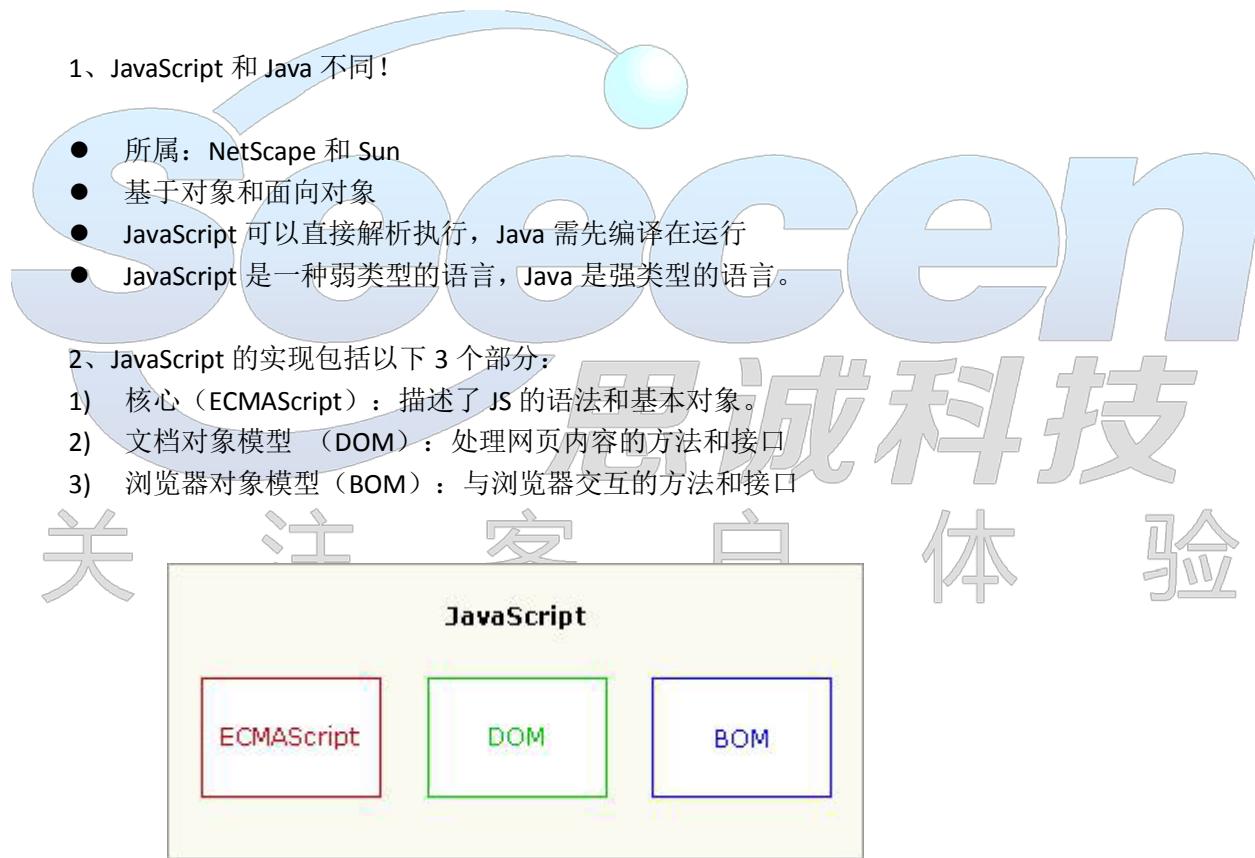
7.1 本章目标

7.2 内容

7.2.1 JavaScript 概述

JavaScript 是网景公司开发的一种在浏览器端执行的脚本语言，是基于对象和事件驱动的客户端脚本语言。有如下特点：

- 交互性
- 安全性（不可以直接访问本地硬盘）
- 跨平台性（只要是可以解析 js 的浏览器都可以执行，和平台无关）



ECMAScript 规范

ECMAScript 是一种由 Ecma 国际（前身为欧洲计算机制造商协会）通过 ECMA-262 标准化的脚本程序设计语言。这种语言在万维网上应用广泛，它往往被称为 JavaScript 或 JScript，但实际上后两者是 ECMA-262 标准的实现和扩展。各个浏览器都严格遵守该规范，没有兼容性问题。ECMAScript 规范由 ECMA 制订。

DOM (document object model 文档对象模型)

主要定义了如何将 HTML 转换成一棵符合 DOM 规范的树，并且如何对这棵树进行相应的操作。

该规范由 W3C 定义，但是，部分浏览器没有严格遵守该规范。写代码时需要考虑兼容性问题。

BOM (browser object model)

浏览器内置的一些对象，用来操作窗口。

这些对象包括 window、screen、location、navigator、document、XmlHttpRequest 等。虽然该部分没有规范，但是，各个浏览器都支持这些对象

3、在 HTML 中使用 JavaScript

- 1) 在<script></script>标签内部直接编写 JS 代码
- 2) 通过 script 标签的 src 属性直接引入外部 js 文件

注意：若以上两种方式同时存在，则标签内部定义的 JS 代码不会被执行。

7.2.2 JavaScript 特点

- 1) javascript 是类 C 的语言
- 2) javascript 脚本可以保存在 .js 文件里，也可以直接写在 html 文件里
- 3) javascript 基于对象，但不是纯粹的面向对象的语言
比如，没有定义类的语法，也没有继承和多态
- 4) javascript 是一种弱类型语言，即变量在声明时，不能明确声明其类型
变量的类型是在运行时确定的，并且可以随时改变

7.2.3 JavaScript 变量

变量在脚本中的第一次出现是在声明中。变量在第一次用到时就设置于内存中，便于后来在脚本中引用。
使用变量之前先进行声明。可以使用 var 关键字来进行变量声明。

JavaScript 是一种区分大小写的语言。因此变量名称 myCounter 和变量名称 mYCounter 是不一样的。变量的名称可以是任意长度。创建合法的变量名称应遵循如下规则：

第一个字符必须是一个 ASCII 字母（大小写均可），或一个下划线(_)。注意第一个字符不能是数字。
后续的字符必须是字母、数字或下划线。

变量名称一定不能是保留字。

变量声明的方法如下：

```
var count; // 单个声明。
```

```
var count, amount, level; // 用单个 var 关键字声明的多个声明。
```

```
var count = 0, amount = 100; // 一条语句中的变量声明和初始化。
```

案例：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>day05_js_Demo10_变量的声明使用</title>
<script>
    var str = "你好，世界";
    document.writeln(str);
</script>
</head>
<body>
</body>
</html>
```

7.2.4 常用的输入/输出

prompt : **prompt** 方法用于显示可提示用户进行输入的对话框。

语法: `prompt(text,defaultText)`

参数	描述
<code>text</code>	可选。要在对话框中显示的纯文本（而不是 HTML 格式的文本）。
<code>defaultText</code>	可选。默认的输入文本。

说明:

如果用户单击提示框的取消按钮，则返回 `null`。如果用户单击确认按钮，则返回输入字段当前显示的文本。在用户点击确定按钮或取消按钮把对话框关闭之前，它将阻止用户对浏览器的所有输入。在调用 `prompt()` 时，将暂停对 JavaScript 代码的执行，在用户作出响应之前，不会执行下一条语句。

alert: 弹出消息对话框(对话框中有一个 **OK** 按钮), **alert**, 中文“提醒”的意思

语法: `alert(" hello word!")`

参数	描述
<code>alert</code>	只有一个参数，可以在里面写纯文本或属性

document.write: 这个命令简单地打印指定的文本内容到页面上。为了逐字打印文本，在打印的文本字符串加上单引号。

语法:`document.write("hello word!");`

案例:

```
<html>
<head>
<title>javaScript 常用的输入输出</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
<script type="text/javascript">
    var name = prompt("请输入您的姓名");
    alert("您的姓名是：" + name);
    document.write("您的姓名是：" + name);
</script>
</head>
<body>
</body>
</html>
```

7.2.5 JavaScript 数据类型

JavaScript 中有 5 种简单数据类型（也称为基本数据类型）：Undefined、Null、Boolean、Number 和 String。还有 1 种复杂数据类型—Object，Object 本质上是由一组无序的名值对组成的。

7.2.5.1 typeof 操作符

介于 JavaScript 是松散类型的，因此需要有一种手段来检测给定变量的数据类型—typeof 就是负责提供者方面信息的操作符。对一个值使用 typeof 操作符可能返回下列某个字符串：

- "undefined"—如果这个值未定义；
- "boolean"—如果这个值是布尔值；
- "string"—如果这个值是字符串；
- "number"—如果这个值是数值；
- "object"—如果这个值是对象或 null；
- "function"—如果这个值是函数；

7.2.5.2 Undefined 类型

Undefined 类型只有一个值，即特殊的 undefined。在使用 var 声明变量但未对其加以初始化时，这个变量的值就是 undefined，例如：

```
var message;
alert(message == undefined) //true
```

7.2.5.3 Null 类型

Null 类型是第二个只有一个值的数据类型，这个特殊的值是 `null`。从逻辑角度来看，`null` 值表示一个空对象指针，而这也正是使用 `typeof` 操作符检测 `null` 时会返回"object"的原因，例如：

```
var car = null;  
alert(typeof car); // "object"
```

如果定义的变量准备在将来用于保存对象，那么最好将该变量初始化为 `null` 而不是其他值。这样一来，只要直接检测 `null` 值就可以知道相应的变量是否已经保存了一个对象的引用了，例如：

```
if(car != null)  
{  
    //对 car 对象执行某些操作  
}
```

实际上，`undefined` 值是派生自 `null` 值的，因此 ECMA-262 规定对它们的相等性测试要返回 `true`。

```
alert(undefined == null); //true
```

尽管 `null` 和 `undefined` 有这样的关系，但它们的用途完全不同。无论在什么情况下都没有必要把一个变量的值显式地设置为 `undefined`，可是同样的规则对 `null` 却不适用。换句话说，只要意在保存对象的变量还没有真正保存对象，就应该明确地让该变量保存 `null` 值。这样做不仅可以体现 `null` 作为空对象指针的惯例，而且也有助于进一步区分 `null` 和 `undefined`。

7.2.5.4 Boolean 类型



该类型只有两个字面值：`true` 和 `false`。这两个值与数字值不是一回事，因此 `true` 不一定等于 1，而 `false` 也不一定等于 0。

虽然 `Boolean` 类型的字面值只有两个，但 JavaScript 中所有类型的值都有与这两个 `Boolean` 值等价的值。要将一个值转换为其对应的 `Boolean` 值，可以调用类型转换函数 `Boolean()`，例如：

```
var message = 'Hello World';  
var messageAsBoolean = Boolean(message);
```

在这个例子中，字符串 `message` 被转换成了一个 `Boolean` 值，该值被保存在 `messageAsBoolean` 变量中。可以对任何数据类型的值调用 `Boolean()` 函数，而且总会返回一个 `Boolean` 值。至于返回的这个值是 `true` 还是 `false`，取决于要转换值的数据类型及其实际值。下表给出了各种数据类型及其对象的转换规则。

这些转换规则对理解流控制语句（如 `if` 语句）自动执行相应的 `Boolean` 转换非常重要，例如：

```
var message = 'Hello World';  
if(message)  
{
```

```
    alert("Value is true");
}
```

运行这个示例，就会显示一个警告框，因为字符串 `message` 被自动转换成了对应的 `Boolean` 值 (`true`)。由于存在这种自动执行的 `Boolean` 转换，因此确切地知道在流控制语句中使用的是什么变量至关重要。

7.2.5.5 Number 类型

这种类型用来表示整数和浮点数值，还有一种特殊的数值，即 `NaN`（非数值 `Not a Number`）。这个数值用于表示一个本来要返回数值的操作数未返回数值的情况（这样就不会抛出错误了）。例如，在其他编程语言中，任何数值除以 0 都会导致错误，从而停止代码执行。但在 `JavaScript` 中，任何数值除以 0 会返回 `NaN`，因此不会影响其他代码的执行。

`NaN` 本身有两个非同寻常的特点。首先，任何涉及 `NaN` 的操作（例如 `NaN/10`）都会返回 `NaN`，这个特点在多步计算中有可能导致问题。其次，`NaN` 与任何值都不相等，包括 `NaN` 本身。例如，下面的代码会返回 `false`。

```
alert(NaN == NaN); //false
```

`JavaScript` 中有一个 `isNaN()` 函数，这个函数接受一个参数，该参数可以使任何类型，而函数会帮我们确定这个参数是否“不是数值”。`isNaN()` 在接收一个值之后，会尝试将这个值转换为数值。某些不是数值的值会直接转换为数值，例如字符串“10”或 `Boolean` 值。而任何不能被转换为数值的值都会导致这个函数返回 `true`。例如：

```
alert(isNaN(NaN)); //true
alert(isNaN(10)); //false(10 是一个数值)
alert(isNaN("10")); //false(可能被转换为数值 10)
alert(isNaN("blue")); //true(不能被转换为数值)
alert(isNaN(true)); //false(可能被转换为数值 1)
```

有 3 个函数可以把非数值转换为数值：`Number()`、`parseInt()` 和 `parseFloat()`。第一个函数，即转型函数 `Number()` 可以用于任何数据类型，而另外两个函数则专门用于把字符串转换成数值。这 3 个函数对于同样的输入会返回不同的结果。

`Number()` 函数的转换规则如下：

- 如果是 `Boolean` 值，`true` 和 `false` 将分别被替换为 1 和 0
- 如果是数字值，只是简单的传入和返回
- 如果是 `null` 值，返回 0
- 如果是 `undefined`，返回 `NaN`
- 如果是字符串，遵循下列规则：

- 如果字符串中只包含数字，则将其转换为十进制数值，即”1“会变成 1， ”123“会变成 123，而”011“会变成 11（前导的 0 被忽略）
 - 如果字符串中包含有效的浮点格式，如”1.1“，则将其转换为对应的浮点数（同样，也会忽略前导 0）
 - 如果字符串中包含有效的十六进制格式，例如”0xf“，则将其转换为相同大小的十进制整数值
 - 如果字符串是空的，则将其转换为 0
 - 如果字符串中包含除了上述格式之外的字符，则将其转换为 NaN
- 如果是对象，则调用对象的 **valueOf()**方法，然后依照前面的规则转换返回的值。如果转换的结果是 NaN，则调用对象的 **toString()**方法，然后再依次按照前面的规则转换返回的字符串值。

```
var num1 = Number("Hello World");      //NaN
var num2 = Number("");                  //0
var num3 = Number("000011");            //11
var num4 = Number(true);                //1
```

由于 **Number()**函数在转换字符串时比较复杂而且不够合理，因此在处理整数的时候更常用的是 **parseInt()**函数。**parseInt()**函数在转换字符串时，更多的是看其是否符合数值模式。它会忽略字符串前面的空格，直至找到第一个非空格字符。如果第一个字符串不是数字字符或者负号，**parseInt()**会返回 NaN；也就是说，用 **parseInt()**转换空字符串会返回 NaN。如果第一个字符是数字字符，**parseInt()**会继续解析第二个字符，知道解析完所有后续字符或者遇到了一个非数字字符。例如，"1234blue"会被转换为 1234，"22.5"会被转换为 22，因为小数点并不是有效的数字字符。

如果字符串中的第一个字符是数字字符，**parseInt()**也能够识别出各种整数格式（即十进制、八进制、十六进制）。为了更好的理解 **parseInt()**函数的转换规则，下面给出一些例子

```
var num1 = parseInt("1234blue");        //1234
var num2 = parseInt("");                 //NaN
var num3 = parseInt("0xA");              //10 (十六进制)
var num4 = parseInt("22.5");             //22
var num5 = parseInt("070");              //56 (八进制)
var num6 = parseInt("70");               //70

var num7 = parseInt("10", 2);            //2 (按二进制解析)
var num8 = parseInt("10", 8);            //8(按八进制解析)
var num9 = parseInt("10", 10);           //10 (按十进制解析)
var num10 = parseInt("10", 16);          //16 (按十六进制解析)
var num11 = parseInt("AF");              //56 (八进制)
var num12 = parseInt("AF", 16);          //175
```

与 **parseInt()**函数类似，**parseFloat()**也是从第一个字符（位置 0）开始解析每个字符。而且也是一直解析到字符串末尾，或者解析到遇见一个无效的浮点数字字符为止。也就是说，字符串中的第一个小

数点是有效的，而第二个小数点就是无效的了，因此它后面的字符串将被忽略。例如，“22.34.5”将被转换成 22.34。

`parseFloat()` 和 `parseInt()` 的第二个区别在于它始终都会忽略前导的零。由于 `parseFloat()` 值解析十进制值，因此它没有用第二个参数指定基数的用法。

```
var num1 = parseFloat("1234blue");      //1234
var num2 = parseFloat("0xA");           //0
var num3 = parseFloat("22. 5");         //22. 5
var num4 = parseFloat("22. 34. 5");     //22. 34
var num5 = parseFloat("0908. 5");       //908. 5
```

7.2.5.6 String 类型

`String` 类型用于表示由零或多个 16 位 Unicode 字符组成的字符序列，即字符串。字符串可以由单引号(')或双引号(")表示。

```
var str1 = "Hello";
var str2 = 'Hello' ;
```

任何字符串的长度都可以通过访问其 `length` 属性取得

```
alert(str1.length);      //输出 5
```

要把一个值转换为一个字符串有两种方式。第一种是使用几乎每个值都有的 `toString()` 方法。

```
var age = 11;
var ageAsString = age.toString();    //字符串"11"
var found = true;
var foundAsString = found.toString(); //字符串"true"
```

数值、布尔值、对象和字符串值都有 `toString()` 方法。但 `null` 和 `undefined` 值没有这个方法。

多数情况下，调用 `toString()` 方法不必传递参数。但是，在调用数值的 `toString()` 方法时，可以传递一个参数：输出数值的基数。

```
var num = 10;
alert(num.toString());      //"10"
alert(num.toString(2));    //"1010"
alert(num.toString(8));    //"12"
alert(num.toString(10));   //"10"
alert(num.toString(16));   //"a"
```

通过这个例子可以看出，通过指定基数，`toString()` 方法会改变输出的值。而数值 10 根据基数的不同，可以在输出时被转换为不同的数值格式。

在不知道要转换的值是不是 `null` 或 `undefined` 的情况下，还可以使用转型函数 `String()`，这个函数能够将任何类型的值转换为字符串。`String()` 函数遵循下列转换规则：

- 如果值有 `toString()` 方法，则调用该方法（没有参数）并返回相应结果
- 如果值是 `null`，则返回“`null`”
- 如果值是 `undefined`，则返回“`undefined`”

```
var value1 = 10;
var value2 = true;
var value3 = null;
var value4;

alert(String(value1));    // "10"
alert(String(value2));    // "true"
alert(String(value3));    // "null"
alert(String(value4));    // "undefined"
```

7.2.5.7 Object 类型

对象其实就是一组数据和功能的集合。对象可以通过执行 `new` 操作符后跟要创建的对象类型的名称来创建。而创建 `Object` 类型的实例并为其添加属性和（或）方法，就可以创建自定义对象。

```
var o = new Object();
```

`Object` 的每个实例都具有下列属性和方法：

- `constructor`—保存着用于创建当前对象的函数
- `hasOwnProperty(propertyName)`—用于检查给定的属性在当前对象实例中（而不是在实例的原型中）是否存在。其中，作为参数的属性名(`propertyName`)必须以字符串形式指定（例如：`o.hasOwnProperty("name")`）
- `isPrototypeOf(object)`—用于检查传入的对象是否是另一个对象的原型
- `propertyIsEnumerable(propertyName)`—用于检查给定的属性是否能够使用 `for-in` 语句来枚举
- `toString()`—返回对象的字符串表示
- `valueOf()`—返回对象的字符串、数值或布尔值表示。通常与 `toString()` 方法的返回值相同。

7.2.6 JavaScript 注释（单行、多行）

7.2.6.1 单行注释语法：

单行的 JScript 注释以一对正斜杠(`//`)开始。下面给出一个单行注释的示例。

```
aGoodIdea = "Comment your code thoroughly."; // 这是一个单行注释。
```

7.2.6.2 多行注释语法:

多行注释以一个正斜杠加一个星号的组合(`/*`)开始,并以其逆向顺序 (`*/`)结束。

```
/*
```

这是一个用来解释前面的代码语句的多行注释。
该语句将一个值赋给 `aGoodIdea` 变量。
用引号包含的这种值称为一个文字。
文字显式并直接包含信息;
而不是简接地引用信息。
(引号不属于该文字的内容。) */

7.2.7 JavaScript 运算符

7.2.7.1 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 **x=10** 和 **y=5**, 下面的表格解释了赋值运算符:

运算符	例子	等价于	结果
=	<code>x=y</code>		<code>x=5</code>
+=	<code>x+=y</code>	<code>x=x+y</code>	<code>x=15</code>
-=	<code>x-=y</code>	<code>x=x-y</code>	<code>x=5</code>
=	<code>x=y</code>	<code>x=x*y</code>	<code>x=50</code>
/=	<code>x/=y</code>	<code>x=x/y</code>	<code>x=2</code>
%=	<code>x%=y</code>	<code>x=x%y</code>	<code>x=0</code>

7.2.7.2 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

给定 **y=5**, 下面的表格解释了这些算术运算符:

运算符	描述	例子	结果
+	加	<code>x=y+2</code>	<code>x=7</code>
-	减	<code>x=y-2</code>	<code>x=3</code>
*	乘	<code>x=y*2</code>	<code>x=10</code>
/	除	<code>x=y/2</code>	<code>x=2.5</code>

%	求余数 (保留整数)	$x=y \% 2$	$x=1$
++	累加	$x=++y$	$x=6$
--	递减	$x=--y$	$x=4$

7.2.7.3 比较运算符

比较运算符用于给 JavaScript 变量做比较。

给定 $x=5$ 下面的表格解释了比较运算符：

运算符	描述	例子	结果
$==$	等于	$x==8$	false
$==$	全等 (值和类型)	$x==5$ $x=="5"$	true false
$!=$	不等于	$x!=8$	false
$>$	大于	$x>8$	false
$<$	小于	$x<8$	true
\geq	大于或等于	$x\geq 8$	false
\leq	小于或等于	$x\leq 8$	true

7.2.7.4 逻辑运算符

关注 客户体验

逻辑运算符用于测定变量或值之间的逻辑。

给定 $x=6$ 以及 $y=3$, 下表解释了逻辑运算符：

运算符	描述	例子
$\&\&$	and	$(x < 10 \&\& y > 1)$ 为 true
$ $	or	$(x==5 y==5)$ 为 false
!	not	$!(x==y)$ 为 true

7.2.7.5 条件运算符(三元运算符)

`variableName=(condition)?value1:value2`

例如: `greeting=(visitor=="PRES")?"Dear President ":"Dear ";`

如果变量 `visitor` 中的值是"PRES", 则向变量 `greeting` 赋值"Dear President ", 否则赋值"Dear"。

7.2.8 JavaScript 逻辑控制语句

通常在写代码时, 您总是需要为不同的决定来执行不同的动作。您可以在代码中使用条件语句来完成该任务。

在 JavaScript 中, 我们可使用以下条件语句:

- **if 语句** - 只有当指定条件为 `true` 时, 使用该语句来执行代码
- **if...else 语句** - 当条件为 `true` 时执行代码, 当条件为 `false` 时执行其他代码
- **if...else if....else 语句** - 使用该语句来选择多个代码块之一来执行
- **switch 语句** - 使用该语句来选择多个代码块之一来执行

7.2.8.1 If...else 条件控制语句

请使用 `if...else` 语句在条件为 `true` 时执行代码, 在条件为 `false` 时执行其他代码。

语法:

```
if (condition)
    statement1
[else
    statement2]
```

如果指定条件为真的时候执行一组语句。如果条件为假的话, 执行另外一组语句。

参数:

`condition` 可以是计算值为真或假的 JavaScript 表达式。条件必须用圆括号括起来。如果 `condition` 的值计算为真的时候, 就会执行 `if` 中 `statement1` 的语句, 否则执行 `else` 中 `statement2` 的语句。

`statement1 statement2` 可以是任意的 JavaScript 语句, 包括更深层次的嵌套 `if` 语句。多个语句必需用大括号括起来。

7.2.8.1.1 案例:

```
<html>
<head>
<title>出租车-if 条件控制语句</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script language="javascript">
    //3:某市出租车3公里的起租价为10元, 3公里以外, 按1.8元/公里计费。现编程
    输入行车里程数, 输出应付车费。
    var b = 2;
    if (b <= 3 && b > 0) {
```

```
        document.write("10 元");
    } else {
        document.write((b - 3) * 1.8 + 10);
    }
</script>
</head>
<body style="text-align:center">
</body>
</html>
```

效果:

7.2.8.1.2 案例 2:

```
<html>
<head>
<title>判断早上中午下午-if 条件控制语句</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script language="javascript">
    var time = 8;
    if (time >= 6 && time < 12) {
        document.write("上午好");
    } else {
        if (time >= 12 && time < 16) {
            document.write("下午好");
        } else {
            if (time >= 16 && time < 24) {
                document.writeln("晚上好");
            }
        }
    }
</script>
</head>
<body style="text-align:left">
</body>
</html>
```

7.2.8.2 if{} else if{} ...else{} 条件控制语句

使用 if...else if...else 语句来选择多个代码块之一来执行。

7.2.8.2.1 语法:

```
if (condition1) {  
    statement1  
} else if (condition2) {  
    Statement2  
} else if (condition3) {  
    Statement3  
} else {  
    Statement4  
}
```

上面代码中,condition1 条件为 false 的话就走下一个 else if 判断 condition2, 假如 condition2 条件也为 false 就走下一个 else if 进条件 condition3, condition3 如果条件为 true 就执行 Statement3 的语句, 否则进 else 走 statement4 的语句。

注意: 如果进了某个条件执行语句则下面的 else if 不再执行。

7.2.8.2.2 案例:

```
<!DOCTYPE html>  
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta charset="utf-8" />  
<title>if 语句 if{}else if{} else</title>  
<script type="text/javascript">  
    var type = prompt("请输入您的类型");  
    var text;  
    if (type == 1) {  
        text = "小正太";  
    } else if (type == 2) {  
        text = "魅力少年";  
    } else if (type == 3) {  
        text = "成熟男生";  
    } else {  
        text = "风趣大叔";  
    }  
    document.write(text);  
</script>  
</head>  
<body>  
</body>  
</html>
```

7.2.8.3 switch 多分支语句

switch 语句用于基于不同的条件来执行不同的动作。

7.2.8.3.1 语法：

```
switch (expression) {  
    case label :  
        statementlist  
        break;  
    case label :  
        statementlist  
        break;  
    default :  
        statementlist  
}
```

工作原理：首先设置表达式 expression（通常是一个变量）。随后表达式的值会与结构中的每个 case 的值做比较。如果存在匹配，则与该 case 关联的代码块会被执行。请使用 break 来阻止代码自动地向下一个 case 运行。default 关键字，如果在没有匹配到的时候就执行 default 中的代码块。

案例 1：

```
<html>  
<head>  
<title>switch 控制语句</title>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8">  
<script language="javascript" type="text/javascript">  
    var hour = prompt("请输入当前的小时数：" , " ");  
    switch (hour) {  
        case "12":  
        case "13":  
        case "14":  
        case "15":  
        case "16":  
        case "17":  
        case "18":  
            document.write("<h2>下午好！欢迎来到思诚</h2>");  
            break;  
        case "19":  
        case "20":  
        case "21":  
        case "22":  
        case "23":  
            document.write("<h2>晚上好！欢迎来到思诚</h2>");  
            break;  
    }  
</script>  
</head>  
<body>  
</body>  
</html>
```

```
default:  
    document.write( "<h2>上午好!欢迎来到思诚</h2>" );  
}  
</script>  
</head>  
<body style="text-align:left">  
</body>  
</html>
```

7.2.8.3.2 案例 2：数据计算

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.d  
td">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
<title>computer</title>  
<script>  
    var op1 = prompt("请输入第一个数字");  
    var op2 = prompt("请输入第二个数字");  
    var sign = prompt("请输入运算符");  
    op1 = parseInt(op1);  
    op2 = parseInt(op2);  
    var result;  
    switch (sign) {  
        case "+":  
            result = op1 + op2;  
            break;  
        case "-":  
            result = op1 - op2;  
            break;  
        case "*":  
            result = op1 * op2;  
            break;  
        case "/":  
            result = op1 / op2;  
            break;  
        default:  
            result = op1 + op2;  
    }  
    document.writeln(result);
```

```
</script>
</head>
<body>
</body>
</html>
```

7.2.8.3.3 案例 3：车速

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>switch</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<script>
    var s = prompt("请输入您的车档速");
    var ret;
    s = parseInt(s);
    switch (s) {
        case 1:
            ret = "您的车速是 10 公里";
            break;
        case 2:
            ret = "您的车速是 20 公里";
            break;
        case 3:
            ret = "您的车速是 30 公里";
            break;
        case 4:
            ret = "您的车速是 40 公里";
            break;
        case 5:
            ret = "您的车速是 50 公里";
            break;
        default:
            ret = "请输入正确的档速";
    }
    document.write(ret);
</script>
</head>
<body>
</body>
</html>
```

7.2.8.4 for 循环

循环可以将代码块执行指定的次数。

7.2.8.4.1 语法:

```
for(初始化; 条件; 增量) {  
    javascript 代码  
}
```

只要指定条件为 true 都执行语句块。

for (initialization; test; increment)

statements

参数

initialization

必选项。一个表达式。该表达式只在执行循环前被执行一次。

test

必选项。一个 Boolean 表达式。如果 test 是 true，则 statement 被执行。如果 test 是 false，则循环结束。

increment

必选项。一个表达式。在每次经过循环的最后执行该递增表达式。

statements

可选项。test 是 true 时，要执行的一个或多个语句。可以是复合语句。

说明

循环要执行确定的次数，通常使用 for 循环。

示例

下面的例子示范了一个 for 循环。

```
/* 在开始时 i 被设为 0，并在每次重复的最后被增加 1。
```

在循环重复前，

如果 i 不小于 10，则循环结束。 */

```
for (var i = 0; i < 10; i++) {  
    document.write(i);  
}
```

会打印在页面 0 1 2 3 4 5 6 7 8 9

7.2.8.4.2 案例：for 循环

```
<html>  
<head>  
<title>for 循环语句</title>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
<script language="javascript">
    //4:以标题 1 至标题 7 的格式输出“欢迎访问我的网站”（for 语句）
    var a;
    for (a = 1; a < 7; a++) {
        document.write("欢迎访问我的网站</h" + a + ">");
    }
</script>
</head>
<body style="text-align:left">
</body>
</html>
```

7.2.8.4.3 案例：打印三角形

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>for 循环应用(双重循环)</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<style type="text/css"></style>
<!--<link rel="stylesheet" type="text/css" href="../styles.css">-->
<script type="text/javascript">
    document.write("<body style='text-align:center;'>")
    for (var i = 1; i < 10; i++) { //控制行
        for (var j = 0; j < i; j++) { //控制列
            document.write(" *&nbsp;&nbsp;&nbsp; ");
        }
        document.write("<br/> ");
    }
    document.write("</body> ")
</script>
</head>
<body style=" text-align: center;">
</body>
</html>
```

7.2.8.4.4 案例：for 循环 99 乘法表

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>for 循环应用(双重循环)</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<style type="text/css"></style>
<!--<link rel="stylesheet" type="text/css" href=".//styles.css">-->
<script type="text/javascript">
    for (var i=1;i<10;i++){
        for (var j=1;j<=i;j++){
            document.write(j+" * "+i+" = "+(i*j)+" &ampnbsp ");
        }
        document.write("<br/>");
    }
</script>
</head>
<body>
</body>
</html>
```

7.2.8.4.5 案例：for 循环百元买百鸡

共 100 元钱，公鸡 5 元/个，小鸡 1 元钱 3 个，请问 100 元钱可以买多少只鸡，并且不找钱。

```
<html>
<head>
<title>百元买百鸡</title>
<meta charset="GBK">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script type="text/javascript">
    var x, y, z;
    for (x = 1; x <= 20; x++) {
        for (y = 1; y <= 33; y++)
        {
            z = 100 - x - y;
            if (5 * x + 3 * y + z / 3 == 100)
            {
                document.write("100 元可以买公鸡" + x + "只, 母鸡" + y + "只, 小鸡" + z + "只" + "<br>")
            }
        }
    }
</script>
</head>
<body>
</body>
</html>
```

```
        }
    }
</script>
</head>
<body>
</body>
</html>
```

7.2.8.5 while 循环

While 循环会在指定条件为真时循环执行代码块。

7.2.8.5.1 语法:

```
while(条件){
    javascript 代码
}
```

执行一个语句，执行的条件为真就一直可以循环，直到指定的条件为 false 才会终止循环。

`while (expression)`

statements

参数
expression

注

客

户

体

验

必选项。Boolean 表达式，在循环前被检查。如果 expression 是 true，则执行循环。如果 expression 是 false，则结束循环。

statements

可选项。expression 是 true 时要执行 javascript 语句。

说明

while 语句在循环第一次被执行前检查 expression。如果 expression 此次是 false，则该循环一次都不执行。

7.2.8.5.2 案例:

```
<html>
<head>
<title>while 循环语句</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
```

```
<script language="javascript">
    //4:以标题 1 至标题 7 的格式输出“欢迎访问我的网站”（for 语句）
    var a = 0;
    while (a < 10) {
        document.write("欢迎访问我的网站" + a + "<br/>");
        a++;
    }
</script>
</head>
<body style="text-align:left">
</body>
</html>
```

7.2.8.6 案例：do-while 循环

do/while 循环是 while 循环的变体。该循环会执行一次代码块，在检查条件是否为真之前，然后如果条件为真的话，就会重复这个循环。

7.2.8.6.1 语法：

```
Do{
    statement
}while (expression);
```

第一次执行一个语句块，然后看 while 里表达式条件是否为 true，是就重复循环的执行该语句块，直到条件表达式等于 false 就不再循环。

参数

statement

可选项。expression 是 true 时要执行的语句。可以是复合语句。

expression

可选项。一个可以强制转换为 Boolean true 或 false 的表达式。如果 expression 是 true，则再执行一次循环。如果 expression 是 false，则结束循环。

说明

在循环的第一次重复执行完成前，不检查 expression 的值，保证至少执行循环一次。此后，循环每成功重复一次后都要检查表达式。

7.2.8.6.2 案例

```
<html>
<head>
<title>while 循环语句 1 加到 100</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
```

```
<meta http-equiv="description" content="this is my page">
<script language="javascript">
    var i = 1;
    var sum = 0;
    do {
        sum += i;
        i++;
    } while (i <= 100);
    alert(sum);
</script>
</head>
<body>
</body>
</html>
```

7.2.8.7 循环中断 break continue

break 语句可用于跳出循环。

continue 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：

7.2.8.7.1 语法：

```
break [label];
```

说明：
中断当前循环，或和 label 一起使用，中断相关联的语句。
可选的 label 参数指定断点处语句的标签。

通常在 switch 语句和 while、for、for...in、或 do...while 循环中使用 break 语句。最一般的是在 switch 语句中使用 label 参数，但它可在任何语句中使用，无论是简单语句还是复合语句。

执行 break 语句会退出当前循环或语句，并开始脚本执行紧接着的语句。

案例:break

```
<html>
<head>
<title>while 循环语句</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript">
    var i = 0;
    for (i = 0; i <= 5; i++) {
        if (i == 3) {
```

```
        break;
    }
    document.write("这个数字是: " + i + "<br/>");
}
</script>
</head>
<body style="text-align:left">
</body>
</html>
```

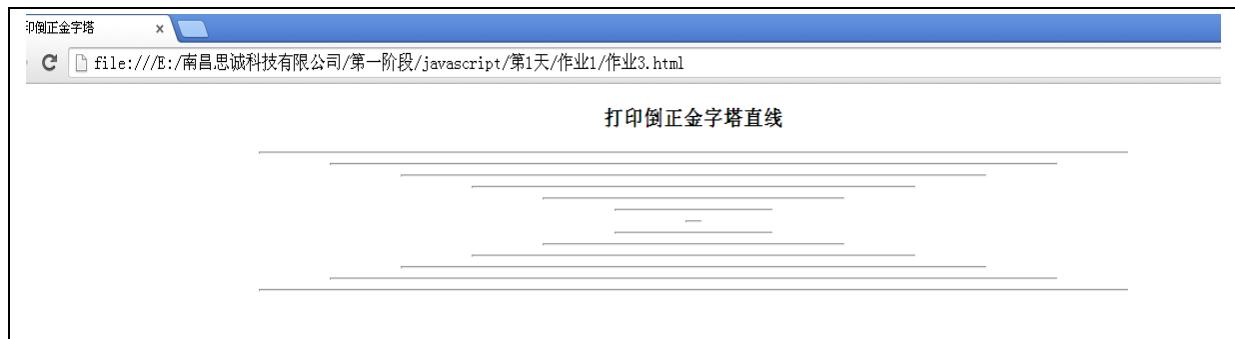
7.2.8.7.2 案例: continue

continue 用于跳过循环中的一个迭代, continue 语句中断循环中的迭代, 如果出现了指定的条件, 然后继续循环中的下一个迭代。

```
<html>
<head>
<title>while 循环语句</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
<script type="text/javascript">
    var i = 0;
    for (i = 0; i <= 5; i++) {
        if (i == 3) {
            continue;
        }
        document.write("这个数字是: " + i + "<br/>");
    }
</script>
</head>
<body style="text-align:left">
</body>
</html>
```

7.2.8.8 循环中断 break continue

1、用循环实现打印金字塔直线



2、根据页面提供的输入数字打印正方形，使用循环实现该功能

例如：prompt 输入了 9，打印出下面的效果



3、用 switch 实现页面输入星期几，然后每天都 alert 一个消息；

例如输入星期一那么提示努力工作，输入星期二那么提示

4、使用 setTimeout 函数设置定时器每隔 1 秒（1000 毫秒），刷新时钟显示

7.2.9 常用的系统对象

7.2.9.1 Date 对象

思诚科技

Date 对象用于处理日期和时间。用如下语句实例化对象：

```
var dateObj = new Date()
```

Date 对象属性

属性	描述
constructor	返回对创建此对象的 Date 函数的引用。
prototype	使您有能力向对象添加属性和方法。

Date 对象方法

方法	描述
Date()	返回当日的日期和时间。

getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。
getMonth()	从 Date 对象返回月份 (0 ~ 11)。
getFullYear()	从 Date 对象以四位数字返回年份。
getYear()	请使用 getFullYear() 方法代替。
getHours()	返回 Date 对象的小时 (0 ~ 23)。
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。
getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。
getTimezoneOffset()	返回本地时间与格林威治标准时间 (GMT) 的分钟差。
getUTCDate()	根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。
getUTCDay()	根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。
getUTCMonth()	根据世界时从 Date 对象返回月份 (0 ~ 11)。
getUTCFullYear()	根据世界时从 Date 对象返回四位数的年份。
getUTCHours()	根据世界时返回 Date 对象的小时 (0 ~ 23)。
getUTCMinutes()	根据世界时返回 Date 对象的分钟 (0 ~ 59)。
getUTCSeconds()	根据世界时返回 Date 对象的秒钟 (0 ~ 59)。
getUTCMilliseconds()	根据世界时返回 Date 对象的毫秒(0 ~ 999)。
parse()	返回 1970 年 1 月 1 日午夜到指定日期 (字符串) 的毫秒数。
setDate()	设置 Date 对象中月的某一天 (1 ~ 31)。
setMonth()	设置 Date 对象中月份 (0 ~ 11)。
setFullYear()	设置 Date 对象中的年份 (四位数字)。
setYear()	请使用 setFullYear() 方法代替。
setHours()	设置 Date 对象中的小时 (0 ~ 23)。
setMinutes()	设置 Date 对象中的分钟 (0 ~ 59)。
setSeconds()	设置 Date 对象中的秒钟 (0 ~ 59)。
setMilliseconds()	设置 Date 对象中的毫秒 (0 ~ 999)。
setTime()	以毫秒设置 Date 对象。
setUTCDate()	根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。

setUTCMonth()	根据世界时设置 Date 对象中的月份 (0 ~ 11)。
setUTCFullYear()	根据世界时设置 Date 对象中的年份 (四位数字)。
setUTCHours()	根据世界时设置 Date 对象中的小时 (0 ~ 23)。
setUTCMinutes()	根据世界时设置 Date 对象中的分钟 (0 ~ 59)。
setUTCSeconds()	根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。
setUTCMilliseconds()	根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。
toSource()	返回该对象的源代码。
toString()	把 Date 对象转换为字符串。
toTimeString()	把 Date 对象的时间部分转换为字符串。
toDateString()	把 Date 对象的日期部分转换为字符串。
toGMTString()	请使用 toUTCString() 方法代替。
toUTCString()	根据世界时，把 Date 对象转换为字符串。
toLocaleString()	根据本地时间格式，把 Date 对象转换为字符串。
toLocaleTimeString()	根据本地时间格式，把 Date 对象的时间部分转换为字符串。
toLocaleDateString()	根据本地时间格式，把 Date 对象的日期部分转换为字符串。
UTC()	根据世界时返回 1970 年 1 月 1 日到指定日期的毫秒数。
valueOf()	返回 Date 对象的原始值。

下面介绍 Date() 对象的主要方法：

关注客户体验

返回 Date 对象中用本地时间表示的年份值。如：返回值 1976

调用方法：

dateObj.getFullYear()

下面这个例子说明了 GetFullYear 方法的用法。

```
var d, s = "今天 UTC 日期是: ";
d = new Date();
s += (d.getMonth() + 1) + "/";
s += d.getDate() + "/";
s += d.getFullYear();
document.write(s);
```

7.2.9.1.2 getMonth 方法

返回 `Date` 对象中用本地时间表示的月份值。

`dateObj.getMonth()`

`getMonth` 方法返回一个处于 0 到 11 之间的整数，它代表 `Date` 对象中的月份值。这个整数并不等于按照惯例来表示月份的数字，而是要比按惯例表示的值小 1。如果一个 `Date` 对象中保存的时间值是 "Jan 5, 1996 08:47:00"，那么 `getMonth` 方法就会返回 0。

示例

下面这个例子说明了 `getMonth` 方法的用法：

```
var d, s = "今天日期是: ";
d = new Date();
s += (d.getMonth() + 1) + "/";
s += d.getDate() + "/";
s += d.getFullYear();
document.write(s);
```

7.2.9.1.3 getDate 方法

返回 `Date` 对象中用本地时间表示的一个月中的日期值。返回值是一个处于 1 到 31 之间的整数，它代表了相应的 `Date` 对象中的日期值。

示例

下面这个例子说明了 `getDate` 方法的用法：

```
var d, s = "今天日期是: ";
d = new Date();
s += (d.getMonth() + 1) + "/";
s += d.getDate() + "/";
s += d.getFullYear();
document.write(s);
```

7.2.9.1.4 getHours 方法

`getHours` 方法

返回 `Date` 对象中用本地时间表示的小时值。

`dateObj.getHours()`

`getHours` 方法返回一个处于 0 到 23 之间的整数，这个值表示从午夜开始计算的小时数。在下面两种情况下此方法的返回值是 0：时间在 1:00:00 am 之前，或者在创建 `Date` 对象的时候没有将时间保存在该对象中。而要确定究竟是哪种情况，唯一的方法就是进一步检查分钟和秒钟值是否也是 0。如果这

两个值都是 0，那就几乎可以确定是没有将时间值保存到 Date 对象中。

下面这个例子说明了 `getHours` 方法的用法。

```
var d, s = "当前本地时间为:";  
var c = ":";  
d = new Date();  
s += d.getHours() + c;  
s += d.getMinutes() + c;  
s += d.getSeconds() + c;  
s += d.getMilliseconds();  
document.write(s);
```

7.2.9.1.5 `getMinutes` 方法

返回 Date 对象中用本地时间表示的分钟值。

`dateObj.getMinutes()`

`getMinutes` 方法返回一个处于 0 到 59 之间的整数，返回值就等于保存在 Date 对象中的分钟值。在下面两种情况下返回值为 0：在时钟整点之后经过的时间少于一分钟，或者是在创建 Date 对象的时候没有将时间值保存到该对象中。而要确定究竟是哪种情况，唯一的方法是同时检查小时和秒钟值是否也是 0。如果这两个值都是 0，那就几乎可以确定是没有将时间值保存到该 Date 对象中。

示例：

下面这个例子说明了 `getMinutes` 方法的用法：

```
var d, s = "当前本地时间为:";  
var c = ":";  
d = new Date();  
s += d.getHours() + c;  
s += d.getMinutes() + c;  
s += d.getSeconds() + c;  
s += d.getMilliseconds();  
document.write(s);
```

7.2.9.1.6 `getSeconds` 方法

返回 Date 对象中用本地时间表示的秒钟值。

`dateObj.getSeconds()`

`getSeconds` 方法返回一个处于 0 到 59 之间的整数，它表示了相应的 Date 对象中的秒钟值。在下面两种情况下，返回值为 0。第一种情况是在当前的一分钟中所经过的时间少于一秒。另外一种情况是在创建 Date 对象时没有将时间值保存到该对象中。而为了确定究竟属于哪种情况，唯一的方法是同时检查小时和分钟值是否也都是 0。如果这两个值都是 0，那就几乎可以确定是没有将时间值保存到 Date

对象中。

示例

下面这个例子说明了 `getSeconds` 方法的用法:

```
var d, s = "当前本地时间为: ";
var c = ":";

d = new Date();
s += d.getHours() + c;
s += d.getMinutes() + c;
s += d.getSeconds() + c;
s += d.getMilliseconds();

document.write(s);
```

7.2.9.1.7 `getDay` 方法

返回 `Date` 对象中用本地时间表示的一周中的日期值。

`dateObj.getDay()`

`getDay` 方法所返回的值是一个处于 **0 到 6** 之间的整数，它代表了一周中的某一天，返回值与一周中日期的对应关系如下：

值	星期
0	星期天
1	星期一
2	星期二
3	星期三
4	星期四
5	星期五
6	星期六

下面这个例子说明了 `getDay` 方法的用法。

```
var d, day, x, s = "今天是: ";
var x = new Array("星期日", "星期一", "星期二");
var x = x.concat("星期三", "星期四", "星期五");
var x = x.concat("星期六");
d = new Date();
day = d.getDay();
document.write(s += x[day]);
```

案例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day05_js_Demo03_Date 对象</title>
<script type="text/javascript">
    var date = new Date();
    var year = date.getFullYear(); //得到 2014 年不是 14
    var month = date.getMonth(); //得到 9
    var day = date.getDate(); //得到 8
    var hour = date.getHours(); //得到 17
    var minute = date.getMinutes(); //得到 44
    var second = date.getSeconds(); //得到 44
    var week = date.getDay(); //得到 3
    document.writeln(year + " - " + month + " - " + day + " " + hour + ":" +
+ minute + ":" + second + " " + "星期" + week);
//          2014-9-8 17:44:44 星期 3
</script>
</head>
<body>
    <div id="myclock"></div>
</body>
</html>
```

7.2.9.2 String 对象

可用于处理或格式化文本字符串以及确定和定位字符串中的子字符串。

语法

```
new String(s);
String(s);
```

参数

参数 s 是要存储在 String 对象中或转换成原始字符串的值。

String 对象属性

属性	描述
constructor	对创建该对象的函数的引用
length	字符串的长度
prototype	允许您向对象添加属性和方法

String 对象方法

方法	描述
anchor()	创建 HTML 锚。
big()	用大号字体显示字符串。
blink()	显示闪动字符串。
bold()	使用粗体显示字符串。
charAt()	返回在指定位置的字符。
charCodeAt()	返回在指定的位置的字符的 Unicode 编码。
concat()	连接字符串。
fixed()	以打字机文本显示字符串。
fontcolor()	使用指定的颜色来显示字符串。
fontsize()	使用指定的尺寸来显示字符串。
fromCharCode()	从字符编码创建一个字符串。
indexOf()	检索字符串。
italics()	使用斜体显示字符串。
lastIndexOf()	从后向前搜索字符串。
link()	将字符串显示为链接。
localeCompare()	用本地特定的顺序来比较两个字符串。
match()	找到一个或多个正则表达式的匹配。
replace()	替换与正则表达式匹配的子串。
search()	检索与正则表达式相匹配的值。
slice()	提取字符串的片断，并在新的字符串中返回被提取的部分。
small()	使用小字号来显示字符串。
split()	把字符串分割为字符串数组。
strike()	使用删除线来显示字符串。
sub()	把字符串显示为下标。
substr()	从起始索引号提取字符串中指定数目的字符。

substring()	提取字符串中两个指定的索引号之间的字符。
sup()	把字符串显示为上标。
toLocaleLowerCase()	把字符串转换为小写。
toLocaleUpperCase()	把字符串转换为大写。
toLowerCase()	把字符串转换为小写。
toUpperCase()	把字符串转换为大写。
toSource()	代表对象的源代码。
toString()	返回字符串。
valueOf()	返回某个字符串对象的原始值。

String 对象的主要方法:

(1) charAt 方法:

返回指定索引位置处的字符。

strObj.charAt(index)

参数

strObj

必选项。任意 String 对象或文字。

index

必选项。想得到的字符的基于零的索引。有效值是 0 与字符串长度减 1 之间的值。

说明

charAt 方法返回一个字符值，该字符位于指定索引位置。字符串中的第一个字符的索引为 0，第二个的索引为 1，等等。超出有效范围的索引值返回空字符串。

示例

下面的示例说明了 charAt 方法的用法:

```
var str = "ABCDEFGHIJKLMNPQRSTUVWXYZ"; // 初始化变量。  
var s; // 声名变量。  
s = str.charAt(5); // 从索引为 n - 1 的位置处  
// 获取正确的字符。  
document.write(s); // 返回字符。
```

(2) search 方法

返回与正则表达式查找内容匹配的第一个子字符串的位置。

stringObj.search(rgExp)

参数

stringObj

必选项。要在其上进行查找的 String 对象或字符串文字。

rgExp

必选项。包含正则表达式模式和可用标志的正则表达式对象。

说明

search 方法指明是否存在相应的匹配。如果找到一个匹配，search 方法将返回一个整数值，指明这个匹配距离字符串开始的偏移位置。如果没有找到匹配，则返回 -1。

示例

下面的示例演示了 search 方法的用法。

```
var r, re; // 声明变量。  
var s = "The rain in Spain falls mainly in the plain.";  
re = /falls/i; // 创建正则表达式模式。  
r = s.search(re); // 查找字符串。  
document.write(r); // 返回 Boolean 结果。
```

(3) substr 方法

返回一个从指定位置开始的指定长度的子字符串。

stringvar.substr(start [,length])

参数

stringvar

必选项。要提取子字符串的字符串文字或 String 对象。

start

必选项。所需的子字符串的起始位置。字符串中的第一个字符的索引为 0。

length

可选项。在返回的子字符串中应包括的字符个数。

说明

如果 length 为 0 或负数，将返回一个空字符串。如果没有指定该参数，则子字符串将延续到 stringvar 的最后。

示例

下面的示例演示了 substr 方法的用法。

```
var s, ss; // 声明变量。  
var s = "The rain in Spain falls mainly in the plain.";  
ss = s.substr(12, 5); // 获取子字符串。  
document.write(ss); // 返回"Spain".
```

(4) substring 方法

返回位于 String 对象中指定位置的子字符串。

strVariable.substring(start, end)

"String Literal".substring(start, end)

参数

start

指明子字符串的起始位置，该索引从 0 开始起算。

end

指明子字符串的结束位置，该索引从 0 开始起算。

说明

substring 方法将返回一个包含从 start 到最后（不包含 end）的子字符串的字符串。

substring 方法使用 start 和 end 两者中的较小值作为子字符串的起始点。例如， strvar.substring(0, 3) 和 strvar.substring(3, 0) 将返回相同的子字符串。

如果 start 或 end 为 NaN 或者负数，那么将其替换为 0。

子字符串的长度等于 start 和 end 之差的绝对值。例如，在 strvar.substring(0, 3) 和 strvar.substring(3, 0) 返回的子字符串的的长度是 3。

示例

下面的示例演示了 substring 方法的用法。

```
var ss; // 声明变量。  
var s = "The rain in Spain falls mainly in the plain...";  
ss = s.substring(12, 17); // 取子字符串。  
document.write(ss); // 返回子字符串。
```

(5) indexOf 方法

返回 String 对象内第一次出现子字符串的字符位置。

strObj.indexOf(subString[, startIndex])

参数

strObj

必选项。String 对象或文字。

subString

必选项。要在 String 对象中查找的子字符串。

startIndex

可选项。该整数值指出在 String 对象内开始查找的索引。如果省略，则从字符串的开始处查找。

说明

indexOf 方法返回一个整数值，指出 String 对象内子字符串的开始位置。如果没有找到子字符串，则返回 -1。

如果 startIndex 是负数，则 startIndex 被当作零。如果它比最大的字符位置索引还大，则它被当作最大的可能索引。

从左向右执行查找。否则，该方法与 `lastIndexOf` 相同。

示例

下面的示例说明了 `indexOf` 方法的用法。

```
var str1 = "BABEBIBOBUBABEBIBOBU"  
var s = str1.indexOf(2);  
document.write(s);
```

(6) `lastIndexOf` 方法

返回 `String` 对象中子字符串最后出现的位置。

`strObj.lastIndexOf(substring[, startIndex])`

参数

`strObj`

必选项。`String` 对象或文字。

`substring`

必选项。要在 `String` 对象内查找的子字符串。

`startIndex`

可选项。该整数值指出在 `String` 对象内进行查找的开始索引位置。如果省略，则查找从字符串的末尾开始。

说明

`lastIndexOf` 方法返回一个整数值，指出 `String` 对象内子字符串的开始位置。如果没有找到子字符串，则返回 `-1`。

如果 `startIndex` 是负数，则 `startIndex` 被当作零。如果它比最大字符位置索引还大，则它被当作最大的可能索引。

从右向左执行查找。否则，该方法和 `indexOf` 相同。

下面的示例说明了 `lastIndexOf` 方法的用法：

```
var str1 = "BABEBIBOBUBABEBIBOBU"  
var s = str1.lastIndexOf(4);  
document.write(s);
```

7.2.9.2.1 `charAt` `trim()` 案例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>  
<head>  
<title>String 对象的方法演示</title>  
<meta http-equiv="content-type" content="text/html ;charset=utf-8">  
<meta http-equiv="description" content="this is my page">  
<!--<link rel="stylesheet" type="text/css" href=".//styles.css">-->  
<script language="javascript">  
    var str = " abcde fg ";
```

```
        alert(str.length);
        str = str.trim();      // 去前后空格
        alert(str.length);
        var s = str.charAt(3); //取第三位的字符串，字符串的长度是从 0 开始的
        alert(s);
        for (var i = 0; i < str.length; i++) {
            alert(str.charAt(i));
        }
        var n = str.indexOf("a");//返回第一次出现的位置查找不到返回-1
        var m = str.lastIndexOf("a");//返回最后一次出现的位置查找不到返回-1
        alert(n);
        alert(m);
        var substr = str.substring(1, 3); // 从第 1 位（含）到第三位(不含)之
前
        alert(substr);
    </script>
</head>
<body>
</body>
</html>
```

7.2.9.2.2 Indexof() 函数 search() 函数案例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional
//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>indexOf</title>
<script>
    var s="abcdefghijkl";
    // 须从第 0 个位置开始找，找第一个位置出现的位置
    var a = s.indexOf("d",0);
    document.writeln(a);
    // 须找出 d 出现的最后一次位置，从右开始找
    var b=s.lastIndexOf("e",9);
    document.writeln(b);
    // 不须定位位置，直接找
    var c= s.search("d");
    document.writeln(c);
</script>
</head>
<body>
</body>
```

```
</html>
```

7.2.9.2.3 substr() 函数 search() 函数案例

substr：返回一个从指定位置开始的指定长度的子字符串

语法：stringvar.substr(start [, length])

substring：返回位于 String 对象中指定位置的子字符串。

语法：strVariable.substring(start, end)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>substr substring</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<script type="text/javascript">
    var str = "aoeiuuybpmfdtnl";
    var str1 = str.substr(4, 5); //返回一个从指定位置开始的指定长度的子字符串
    var str2 = str.substring(6, 8); //返回位于 String 对象中指定位置的子字符串
    document.writeln(str1);
    document.writeln(str2);
</script>
</head>
<body>
</body>
</html>
```

7.2.9.3 parseInt ("字符串")

将字符串转换为整型数字

如: parseInt ("86") 将字符串 “86” 转换为整型值 86

7.2.9.4 parseFloat("字符串")

将字符串转换为浮点型数字

如: parseFloat("34.45") 将字符串 “34.45” 转换为浮点值 34.45

7.2.9.5 Math 对象方法演示

是一个固有对象，提供基本数学函数和常数。

使用 Math 的属性和方法的语法：

```
var pi_value=Math.PI;  
var sqrt_value=Math.sqrt(15);
```

注释：Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。

Math 对象属性

属性	描述
E	返回算术常量 e，即自然对数的底数（约等于 2.718）。
LN2	返回 2 的自然对数（约等于 0.693）。
LN10	返回 10 的自然对数（约等于 2.302）。
LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。
LOG10E	返回以 10 为底的 e 的对数（约等于 0.434）。
PI	返回圆周率（约等于 3.14159）。
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。
SQRT2	返回 2 的平方根（约等于 1.414）。

Math 对象方法

方法	描述
abs(x)	返回数的绝对值。
acos(x)	返回数的反余弦值。
asin(x)	返回数的反正弦值。
atan(x)	以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。
atan2(y,x)	返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。
ceil(x)	对数进行上舍入。
cos(x)	返回数的余弦。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。

log(x)	返回数的自然对数（底为 e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0~1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。
toSource()	返回该对象的源代码。
valueOf()	返回 Math 对象的原始值。

案例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Math 对象方法演示</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<!--<link rel="stylesheet" type="text/css" href=". /styles.css">-->
<script type="text/javascript" language="javascript">
    var num1 = -20;
    var num2 = 20;
    var num3 = Math.abs(num1);
    num2 = Math.abs(num2);
    alert(Math.abs(num1) == Math.abs(num2));
    var num3 = 2.4;
    alert(Math.ceil(num3));
    alert(Math.floor(num3));
    alert(Math.round(num3));
    var a = prompt("请输入你的幸运数字", " ");
    var userNum = parseInt(a);
    var i = Math.round(Math.random() * 10);
    alert(i);
    if (userNum == i) {
        document.write("<h1>恭喜你中奖啦! </h1>");
    } else {
        document.write("<h3>谢谢惠顾! </h3>");
    }
</script>

```

```
        }
    </script>
</head>
<body>
</body>
</html>
```

7.2.9.6 isNaN()

用于检查其参数是否是非数字

返回一个 Boolean 值，指明提供的值是否是保留值 NaN（不是数字）。

isNaN(numValue)

必选项 numvalue 参数为要检查是否为 NAN 的值。

说明

如果值是 NaN，那么 isNaN 函数返回 true，否则返回 false。使用这个函数的典型情况是检查 parseInt 和 parseFloat 方法的返回值。

还有一种办法，变量可以与它自身进行比较。如果比较的结果不等，那么它就是 NaN。这是因为 NaN 是唯一与自身不等的值。

7.2.9.7 typeOf()

用途：用于检验数据的类型；

案例：

```
<html>
<head>
<title>函数的调用触发演示</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript">
    var num1 = 1.2;
    var str = "125.66";
    document.writeln("num1 之前的数据类型是" + typeof (num1) + "值是" +
num1 + "<br/>");
    document.writeln("num1 转换为浮点之后的数据类型是" + typeof
(parseFloat(num1)) + "值是：" + parseFloat(num1) + "<br/>");
    document.writeln("str 之前的数据类型是" + typeof (str) + "值是：" +
str + "<br/>");
    document.writeln("str 转换为整形之后的数据类型是" + typeof
(parseInt(str)) + "值是：" + parseInt(str) + "<br/>");
    document.writeln("str 转换为浮点之后的数据类型是" + typeof
(parseFloat(str)) + "值是：" + parseFloat(str) + "<br/>");
```

```
var str2 = "nanchang";
var str3 = "125";
alert("str2=" + str2 + " str2 是非数字? " + isNaN(str2));
alert("str3=" + str3 + " str3 是非数字? " + isNaN(str3));
// alert(parseInt(num1));
// alert(typeof(parseInt(num1)));
</script>
</head>
<body>
</body>
</html>
```

7.2.9.8 作业

- 1、练习数据类型的检测（`typeof`）
- 2、熟练掌握输入输出函数的用法

7.2.10 函数

函数定义：是用于完成特定任务的代码语句块；

使用更简单：不用定义属于某个类，直接使用；
函数分为：系统函数、自定义函数

根据函数的返回值分为：无参函数、有参函数

函数是 JavaScript 的一种基本数据类型

7.2.10.1.1 案例 1：直接触发

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day06_js_Demo01_函数直接触发</title>
</head>
<body>
<input type="button" name="butn1" value="触发函数"
onclick="javascript:alert('Hello World1!');" ></input>
<input type="button" name="butn2" value="触发函数" onclick="alert('Hello
World2!');" ></input>
</body>
</html>
```

7.2.10.1.2 案例 2：函数的调用触发

```
<html>
<head>
<title>函数的调用触发演示</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript">
    function func1() {
        alert("Hello World!!!!");
    }
</script>
</head>
<body>
<input type="button" name="butn1" value="触发函数" onclick="func1();"
<!-- <input type="button" name="butn1" value="触发函数"
onclick="alert('Hello World!');" -->
</body>
</html>
```

定时函数

setTimeout()用法

setTimeout ("调用的函数", "指定的时间后")

setInterval()方法

setInterval ("调用的函数", "指定的时间间隔")

关注客户体验

```
var myTime=setTimeout("disptime()", 1000);
```

```
var myTime=setInterval("disptime()", 1000);
```

setTimeout()只执行 disptime()一次，如果要多次调用使用 setInterval()者者

让 disptime()自身再次调用 setTimeout()

案例 1：函数内实现时间循环

```
<html>
<head>
<title>Date 时间对象的方法演示</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script language="javascript">
    function f() {
        var date = new Date();
```

```
var month = date.getMonth() + 1;
var day = date.getDate();
var hours = date.getHours();
var minutes = date.getMinutes();
var seconds = date.getSeconds();

var innerStr = month + " 月 " + day + " 日 " + hours + ":" + minutes
+ ":" + seconds;

document.getElementById("innerdiv").innerHTML = innerStr;
setTimeout("f()", 5000);
}

</script>
</head>
<body>
<div id="innerdiv"></div>
<input type="button" value="开始计时" onclick="f();"/>
</body>
</html>
```

案例 2：时间循环 函数外实现

```
<html>
<head>
<title>时间循环</title>
<script language="javascript">
    function showTime() {
        var today = new Date();
        var hh = today.getHours();
        var mm = today.getMinutes();
        var ss = today.getSeconds();
        document.getElementById("dt").innerHTML = ss;
    }
    setInterval("showTime()", 1000);
</script>
</head>
<body>
<div id="dt"></div>
</body>
</html>
```

7.2.11 自定义函数

创建函数

无参函数

有参函数

调用函数

function 函数名()

```
{  
    JavaScript 代码;  
}
```

function 函数名(参数 1, 参数 2, ...)

```
{  
    JavaScript 代码;  
}
```

函数调用一般和表单元素的事件一起使用，调用格式：

事件名 = “函数名()”；

7.2.11.1.1 案例1 无参函数：调用无参函数，输出 10 次“HelloWorld”

```
<html>  
<head>  
<title>无参函数的演示</title>  
<meta http-equiv="content-type" content="text/html; charset=UTF-8">  
<script type="text/javascript">  
    var num = prompt("请输入 Hello World 需要显示的次数");  
    function func1() {  
        for (var i = 0; i < num; i++) {  
            // alert("Hello World!!!!");  
            document.writeln("Hello World!!!!" + "<br/>");  
        }  
    }  
</script>  
</head>  
<body>  
<input type="button" name="butn1" value="触发函数" onclick="func1(); " >
```

```
<!-- <input type="button" name="butn1" value="触发函数"
onclick="alert('Hello World!');" > -->
</body>
</html>
```

7.2.11.1.2 案例 2 有参函数：调用无参函数，输出 10 次“HelloWorld”

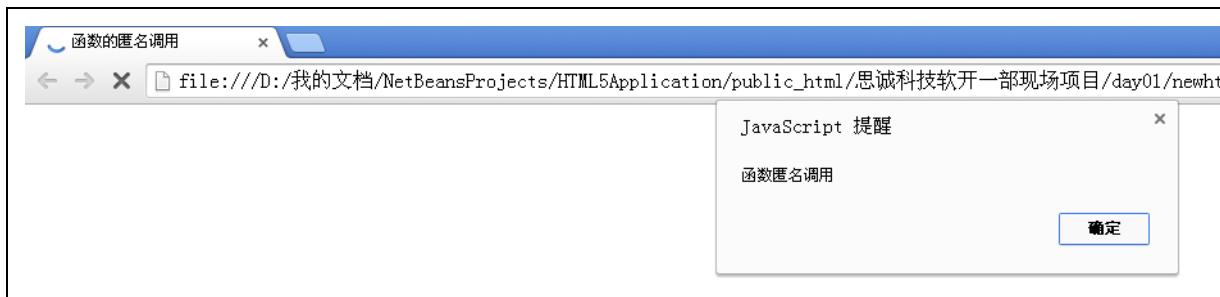
```
<html>
<head>
<title>有参函数的演示</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript">
    function func1(count) {
        for (var i = 0; i < count; i++) {
            // alert("Hello World!!!!");
            document.writeln("Hello World!!!!" + "<br/>");
        }
    }
</script>
</head>
<body>
<input type="button" name="butn1" value="触发函数" onclick="func1(prompt('请输入 Hello World 的显示次数', '10'));">
<!-- <input type="button" name="butn1" value="触发函数"
onclick="alert('Hello World!');" > -->
</body>
</html>
```

7.2.12 函数的匿名调用

案例：

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>函数的匿名调用</title>
<meta charset="utf-8" />
</head>
<body onload=(function(){alert("函数匿名调用")})();>
</body>
</html>
```

效果：



7.2.13 变量的作用域

JScript 有两种变量范围：全局和局部。如果在任何函数定义之外声明了一个变量，则该变量为全局变量，且该变量的值在整个持续范围内都可以访问和修改。如果在函数定义内声明了一个变量，则该变量为局部变量。每次执行该函数时都会创建和破坏该变量；且它不能被该函数外的任何事物访问。

一个局部变量的名称可以与某个全局变量的名称相同，但这是完全不同和独立的两个变量。因此，更改一个变量的值不会影响另一个变量的值。在声明局部变量的函数内，只有该局部变量有意义。

```
var aCentaur = "a horse with rider,"; // aCentaur 的全局定义。
```

```
// JScript 代码，为简洁起见有省略。
```

```
function antiquities() // 在这个函数中声明了一个局部 aCentaur 变量。
```

```
{
```

```
// JScript 代码，为简洁起见有省略。
```

```
var aCentaur = "A centaur is probably a mounted Scythian warrior";
```

```
// JScript 代码，为简洁起见有省略。
```

```
} // 函数结束。
```

```
var nothinginparticular = antiquities();
```

```
aCentaur += " as seen from a distance by a naive innocent.";
```

```
/*
```

在函数内，该变量的值为 "A centaur is probably a mounted Scythian warrior,

misreported; that is, "；在函数外，该变量的值为这句话的其余部分：

"a horse with rider, as seen from a distance by a naive innocent."

*/

JScript 在运行代码前处理变量声明，所以声明是位于一个条件块中还是其他某些结构中无关紧要。JScript 找到所有的变量后立即运行函数中的代码。如果变量是在函数中显式声明的 ----- 也就是说，如果它出现于赋值表达式的左边但没有用 var 声明 ----- 那么将把它创建为全局变量。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>day05_js_Demo03_变量的作用域</title>
<script language="javascript" type="text/javascript">
    var num; // 全局变量的演示
    function showHello() {
        var num2 = num;
        document.write("<h2>Hello World</h2>");
    }
    function counts() {
        num = prompt("请输入显示 HelloWorld 的次数: ", "");
        showHello();
    }
</script>
</head>
<body>
<input name="btn" type="button" value="请输入显示 HelloWorld 的次数"
onclick="counts()"/>
</body>
</html>
```

7.2.14 数组

数组对象的作用是：使用单独的变量名来存储一系列的值。

JavaScript 数组和 Java 的数组不同，JS 数组的大小是不定长的，而 Java 的数组长度是固定不可变化的；

Js 数组的创建有两种方式：

1、直接赋值

Var arr=[12, 23, 44, 55, 66];

2、使用 Array 对象创建

var arr = new Array();

```
var arr = new Array(3) ;
```

7.2.14.1 案例 1：数组直接赋值

```
<!DOCTYPE html>

<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript">
    var array = [1, 2, true, "abc", null, undefined, 321, NaN];
    var length = array.length;
    for (var i = 0; i < array.length; i++) {
        var item = array[i],
            type = typeof (item);
        document.write(type + ":" + item + "<br />");
    }
    var i = 0;
    while (i++ < array.length) {
        var item = array[i - 1],
            type = typeof (item);
        document.write(type + ":" + item + "<br />");
    }
    var i = 0;
    do {
        var item = array[i],
            type = typeof (item);
        document.write(type + ":" + item + "<br />");
    } while (++i < array.length);
    document.write("循环调用完毕！");
</script>
</head>
<body>
</body>
</html>
```

7.2.14.2 案例 2：使用 Array 创建且直接赋值

```
<html>
<!--数组-->
<head>
<script>
```

```
/* 创建数组 */
function f1(){
    /* 创建数组的第一种方式
     * 使用 Array。 */
    var arr1 = new Array() ;
    arr1[0] = 1 ;
    arr1[7] = 12 ;
    alert("arr1[2]="+arr1[2]) ;
    alert("arr1.length:"+arr1.length) ;

    /* 创建数组的第二种方式
     * 使用[]。
     */
    var arr2 = [] ;
    arr2[3] = 12 ;
    alert("arr2.length:"+arr2.length) ;
    var arr3 = [1,2,'abc'] ;
    arr3[4] = 12 ;
    alert("arr3[2]:"+arr3[2]) ;
}
</script>
<body onload="f1();">
</body>
</html>
```

7.2.14.3 案例 3：Array 数组实现从小到大排序的演示

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Array 数组实现从小到大排序的演示</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<!--<link rel="stylesheet" type="text/css" href=".//styles.css">-->
<script language="javascript">
    var array2 = new Array(10, 20, 30, 50, 40, 5, 6, 70);
    for (var i = 0; i < array2.length; i++) {
        for (var j = i + 1; j < array2.length; j++) {
            if (array2[j] < array2[i]) {
                var temp = array2[i];
                array2[i] = array2[j];
                array2[j] = temp;
            }
        }
    }
</script>
</head>
<body>
<h1>思诚科技</h1>
<p>关注客户体验</p>
</body>
</html>
```

```
        }
        for (var i = 0; i < array2.length; i++) {
            alert(array2[i]);
        }
    </script>
</head>
<body>
</body>
</html>
```

7.2.14.4 案例 4：Array 数组求最大值

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>最大值</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<script type="text/javascript">
    function setE() {
        var a = parseInt(document.getElementById("a").value);
        var b = parseInt(document.getElementById("b").value);
        var c = parseInt(document.getElementById("c").value);
        var d = parseInt(document.getElementById("d").value);
        document.getElementById("e").innerHTML = fun1(a, b, c, d);
    }
    function fun1(a, b, c, d) {
        var max
        if (a >= b && a >= c && a >= d) {
            max = a;
        } else if (b >= a && b >= c && b >= d) {
            max = b;
        } else if (c >= a && c >= b && c >= d) {
            max = c;
        } else if (d >= a && d >= b && d >= c) {
            max = d;
        }
        return max;
    }
    // 下面这些是实现的第二种方法    function fun2(a,b,c,d){
    //         var arr = [a,b,c,d];
    //     }
```

```
//          //最大值的下标, 先假定为第一个元素的下标
//          var index = 0;
//          for(var x = 0; x < arr.length; x++){
//
//              if(arr[index] < arr[x]){
//
//                  index = x;
//
//              }
//
//          }
//
//          document.write("索引为" + index + "中的" + arr[index] + "最大");
//          return arr[index];
//
//
//      }
//
//      function maxs() {
//          var ary = [document.getElementById("a").value,
//          document.getElementById("b").value,
//          document.getElementById("c").value,
//          document.getElementById("d").value];
//          var temp = 0;
//          for (var i = 0; i < ary.length; i++) {
//              if (temp < ary[i]) {
//                  temp = ary[i];
//              }
//          }
//          document.write("最大数是:" + temp);
//      }
//      function setE2() {
//          var a = parseInt(document.getElementById("a").value);
//          var b = parseInt(document.getElementById("b").value);
//          var c = parseInt(document.getElementById("c").value);
//          var d = parseInt(document.getElementById("d").value);
//          document.getElementById("e").innerHTML = fun2(a, b, c, d);
//      }
//</script>
</head>

<body>
<!-- 1、在界面中有 4 个输入框 还有一个按钮
2、在四个输入框中分别输入 4 个整数, 通过按钮点击可在界面上输出最大值
3、用 2 中方法分支结构 和循环结构--&gt;</pre>
```

第一个数: <input type="text" id="a">

第二个数: <input type="text" id="b">


```
第三个数: <input type="text" id="c"><br/>
第四个数: <input type="text" id="d"><br/>
最大值是:<span id="e"></span>
<input type="button" value = "最大值" onclick="maxs( );">
</body>
</html>
```

7.2.14.5 案例 5：遍历思维

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>遍历思维</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<script type="text/javascript">
for (var i = 1; i < 45; i++) {
var a = i - 2;
var b = i + 2;
var c = i * 2;
var d = i / 2;
if (a + b + c + d == 45) {
document.write("这四个数是" + a + "&ampnbsp" + b + "&ampnbsp" + c + "&ampnbsp" + d +
"<br/> ");
}
}
</script>
</head>
<body>
<!-- 钱柜中有 4 5 元，须输入四个数的密码，每个数位数不限，<br/>
四个数之和是 4 5，第一个数 + 2 和第二个数 - 2 第三个× 2 第四个除于 2 结果相等，  

这四个数就是密码，请输出密码-->
</body>
</html>
```

7.2.15 BOM 对象 (Window 对象)

7.2.15.1 概述

浏览器对象模型 (BOM) 使 JavaScript 能够与浏览器进行“对话”，获取浏览器信息，操作浏览器。虽然 W3C 并没有对 BOM 作出规范，但是所有浏览器都支持 BOM，有一些事实上的标准。

Window 对象表示当前浏览器的窗口，是 JavaScript 的顶级对象，我们创建的所有对象、函数、变量都是 Window 对象的成员。

不过，一般情况下我们的代码中省略了 window 对象，浏览器默认会作为 window 对象的成员来调用。

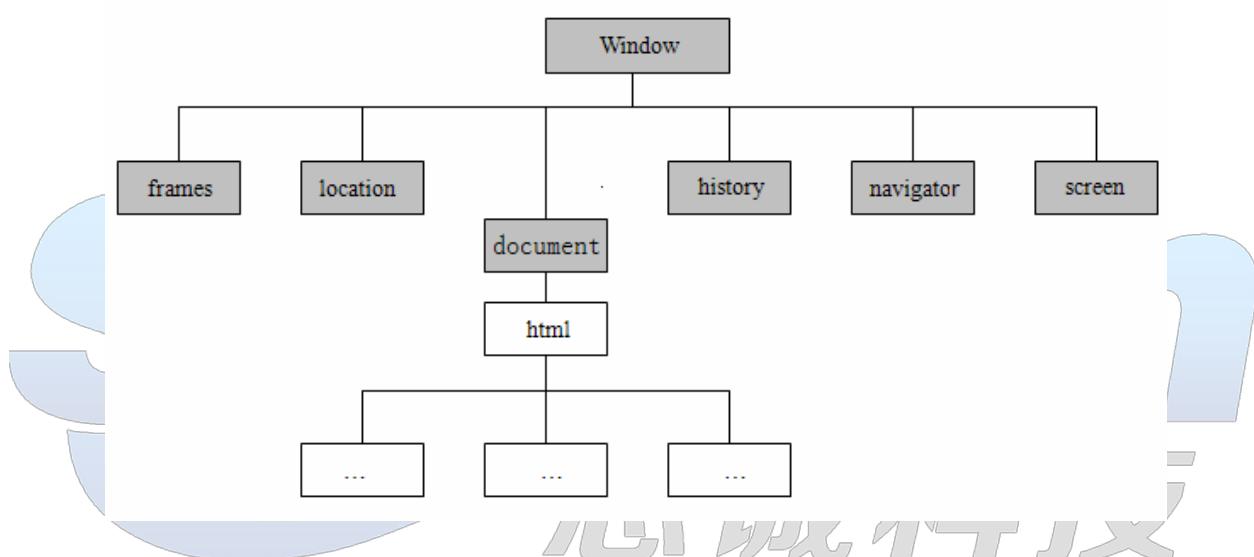
例如，调用一个全局变量 myName 的完整写法是：

```
window.myName;
```

但是我们完全可以这样写：

```
myName;
```

JavaScript 顶级对象参考模型



关注客户体验

7.2.15.2 常用的属性

screen: 包含了描述显示屏幕和颜色的属性

history: 包含了一组客户已经在窗口中浏览过的 URL 的信息

location: 包含了当前 URL 的信息

案例：location

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript">
    function jump() {
```

```
        window.location.href = "http://www.baidu.com";
    };
    function refresh() {
        window.location.reload();
    };
</script>
</head>
<body>
<input type="button" value="跳转" onclick="javascript: jump();" />
<input type="button" value="刷新" onclick="javascript: refresh();" />
</body>
</html>
```

案例: history

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript">
    function back() {
        //window.history.back();
        window.history.go(-1);
    };
    function forward() {
        //window.history.forward();
        window.history.go(1);
    };
</script>
</head>
<body>
<input type="button" value="返回" onclick="javascript: back();" />
<input type="button" value="前进" onclick="javascript: forward();" />
</body>
</html>
```

案例: screen

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta charset="utf-8" />
<title></title>
<script type="text/javascript">
    alert(window.screen.height);
</script>
</head>
<body>
</body>
</html>
```

效果：



7.2.15.3 常用的方法

思诚科技

名称	说明
prompt	显示可提示用户输入的对话框
alert	显示带有一个提示信息和一个确定按钮的警报框
confirm	显示一个带有提示信息、确定和取消按钮的对话框
close	关闭浏览器窗口
open	打开一个新的浏览器窗口，加载给定 URL 所指定的文档
setTimeout	在指定的毫秒数后调用函数或计算表达式
setInterval	按照指定的周期（以毫秒计）来调用函数或表达式

主要的方法讲解

1) confirm() 方法

confirm 方法返回值是 boolean 值

confirm("对话框中显示的纯文本")

```
<script language="javascript" type="text/javascript">  
var flag=confirm("确认要删除此条信息吗？");  
  
if(flag==true){  
  
    alert("删除成功！");  
}  
else{  
    alert("你取消了删除");  
}  
</script>
```

confirm()与 alert ()、 prompt()区别

2) open() 方法

窗口特征

名称	说明
height、width	窗口文档显示区的高度、宽度。以像素计。
left、top	窗口的x坐标、y坐标。以像素计
toolbar=yes no 1 0	是否显示浏览器的工具栏。默认是yes。
scrollbars=yes no 1 0	是否显示滚动条。默认是yes。
location=yes no 1 0	是否显示地址地段。默认是yes。
status=yes no 1 0	是否添加状态栏。默认是yes。
menubar=yes no 1 0	是否显示菜单栏。默认是yes。
resizable=yes no 1 0	窗口是否可调节尺寸。默认是yes。
titlebar=yes no 1 0	是否显示标题栏。默认是yes。
fullscreen=yes no 1 0	是否使用全屏模式显示浏览器。默认是no。处于全屏模式的窗口必须同时处于剧院模式。

7.2.15.3.1 案例：open

```
<html>  
<head>
```

```
<title>计算两个数的运算结果演示</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script type="text/javascript">
    function opens() {
        window.open("baidu.html");
    }
</script>
</head>
<body>
<form action="" method="post">
<input name="btn" type="button" value="弹出窗口" onclick="opens()" />
</form>
</body>
</html>
```

baidu.htm 子文件 1

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>百度搜索页面</title>
<script type="text/javascript">
window.open("adv.html","","",
width=350,height=229,toolbar=0,scrollbars=0,location=0,status=0,menubar=0,resizable=0");
</script>
</head>
<body>

</body>
</html>
```

adv.html 子文件 2

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>广告页面</title>
```

```
</head>

<body style="margin:0">

</body>
</html>
```

7.2.15.3.2 案例：confirm

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<script language="javascript" type="text/javascript">
    var flag = confirm("确认要删除此条信息吗? ");
    if (flag == true) {
        alert("删除成功!");
    } else {
        alert("你取消了删除");
    }
</script>
</head>
<body>
</body>
</html>
```

案例代码：

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<script type="text/javascript">
    function open_adv() {
        window.open("adv.html");
    }
    function open_fix_adv() {
        window.open("adv.html", "", "height=380,width=320,toolbar=0,
scrollbars = 0, location = 0, status = 0, menubar = 0, resizable = 0");
    }
</script>
</head>
<body>
</body>
</html>
```

```
function fullscreen() {
    window.open("plan.html", "", "fullscreen=yes");
}
function confirm_msg() {
    if (confirm("你相信自己是最棒的吗? "))
    {
        alert("有信心必定会赢, 没信心一定会输!");
    }
}
</script>
</head>
<body>
<input name="open1" type="button" value="弹出窗口" onclick="open_adv()" />
<input name="open2" type="button" value="弹出固定大小窗口, 且无菜单栏等"
onclick="open_fix_adv()" />
<input name="full" type="button" value="全屏显示" onclick="fullscreen()" />
<input name="con" type="button" value="打开确认窗口"
onclick="confirm_msg()" />
</body>
</html>
```

7.2.15.4 常用的事件

Onload: 在浏览器完成对象的装载后立即触发;

Onchange: 内容发生改变后触发

Onmouseover: 鼠标移动到页面元素上触发

Onmouseout: 鼠标离开页面元素后触发

Onfocus: 得到焦点后触发

Onblur: 失去焦点后触发

Onkeyup: 当用户释放键盘上的一个键时触发 KeyUp 事件

onKeyDown: 当用户按下键盘的一个键时触发事件

7.2.15.4.1 Onload 事件

onload 事件是 Event 的对象

onload 事件会在页面或图像加载完成后立即发生。

语法

onload="SomeJavaScriptCode"参数描述

SomeJavaScriptCode 必需。规定该事件发生时执行的 JavaScript。

支持该事件的 HTML 标签:

<body>, <frame>, <frameset>, <iframe>, , <link>, <script>

支持该事件的 JavaScript 对象:

image, layer, window 实例

在本例中, 文本 "Page is loaded" 会被显示在状态栏中:

```
<html>
<head>
<script type="text/javascript">
function load()
{
}
window.status="Page is loaded"
}
</script>
</head>
<body onload="load()">
</body>
</html>
```

案例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>onload事件演示</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<!--<link rel="stylesheet" type="text/css" href=".//styles.css">-->
<script type='text/javascript'>
    function func1() {
        alert("你好 Hello World!");
    }
</script>
</head>
<body onload="func1();">
</body>
</html>
```

7.2.15.4.2 onchange 事件

当 change 事件发生时执行的 JavaScript 代码。当一个 Select、Text、或 Textarea 域失去焦点并更改值时触发 change 事件。

事件适用对象 FileUpload, Select, Text, Textarea

实现版本 Navigator 2.0:适用于 Select, Text, 和 Textarea

Navigator 3.0: 添加了对 FileUpload 的适用

语法

onChange="handlerText"

参数

handlerText JavaScript 代码或对一个 JavaScript 函数的调用。

描述

使用 onChange 在用户修改数据后进行校验。

使用的事件属性

type 标明了事件的类型。

target 标明了事件原来发送的对象。

示例

在下面的例子中, userName 是一个文本域。当用户更改文本并离开该域时, onChange 事件句柄会调用 checkValue 函数确认 userName 中是否存放了合法值。

```
<INPUT TYPE="text" VALUE="" NAME="userName" onChange="checkValue(this.value)">
```

案例

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>onchange的用法</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<meta http-equiv="description" content="this is my page">
<!--<link rel="stylesheet" type="text/css" href=". /styles.css">-->
<script>

    function cityChange(){
        var province=document.getElementById("province").value;
        var str;
        if(province=="1"){
            str+="<option>南昌</option>" ;
            str+="<option>九江</option>" ;
            str+="<option>上饶</option>" ;
        }else{
            str+="<option>广州</option>" ;
            str+="<option>深圳</option>" ;
            str+="<option>中山</option>" ;
        }
        document.getElementById("city").innerHTML=str;
    }
</script>
```

```
</head>
<body>
省份<select name="province" id="province" onchange="cityChange()">
<option value="1" >江西省</option>
<option value="2" selected="selected">广东省</option>
</select>
地市<select name="city" id="city">
</select>
</body>
</html>
```

7.2.15.4.3 onMouseOver 事件

事件会在鼠标指针移动到指定对象上时触发



```
<html>
<head>
<title>onchange</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<a href="http://www.w3school.com.cn"
onmouseover="alert('鼠标移到超链接元素上的反馈 An onMouseOver event');
return false">

</a>
</body>
</html>
```

7.2.15.4.4 onMouseOut 事件

当 MouseOut 事件发生时执行的 JavaScript 代码。当每次鼠标指针离开区域(客户端图像地图)或链接时触发 MouseOut 事件。

语法

onMouseOut="handlerText"

参数

handlerText JavaScript 代码或对一个 JavaScript 函数的调用。

描述

如果鼠标指针从客户端图像地图的一个区域移到了另外一个区域，则第一个区域会收到 onMouseOut 事件，而第二个区域会收到 onMouseOver 事件。

案例：

```
<html>
<head>
<title>onchange</title>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
</head>
<body>
<a href="http://www.w3school.com.cn"
    onMouseOut="alert('鼠标离开超链接元素后的反馈 An onMouseOut event'); return
false">

</a>
</body>
</html>
```

7.2.15.4.5 onfocus 事件

当输入框获得焦点时触发



```
<html>
<head>
<title>onfocus</title>
<script type="text/javascript">
    function setStyle(x)
    {
        document.getElementById(x).style.background = "yellow"
    }
</script>
</head>
<body>
    First name: <input type="text" onfocus="setStyle(this.id)" id="fname" />
    <br />
    Last name: <input type="text" onfocus="setStyle(this.id)" id="lname" />
</body>
</html>
```

7.2.15.4.6 onblur 事件

失去了焦点后触发

```
<HTML>
<HEAD>
<meta http-equiv="content-type" content="text/html ;charset=utf-8">
<title>onblur</title>
```

```
</HEAD>
<BODY>
<INPUT TYPE=text NAME=txtFName VALUE="First Name"
       onblur="alert('失去了焦点')">
</BODY>
</HTML>
```

7.2.15.4.7 Onkeydown、onkeyup 事件

当键盘被按下和松开时分别触发 onkeydown 和 onkeyup 事件

```
<!DOCTYPE html>
<html>
<head>
<title>TODO supply a title</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<SCRIPT>
function fun1() {
    id3.innerText += "鼠标按下触发";
}
function fun2() {
    id3.innerText += "鼠标弹起触发";
}
</SCRIPT>
</head>
<body>

<input id="id1" type="text" onkeydown="fun1()" onkeyup="fun2()">

<TEXTAREA ID="id3" >
</TEXTAREA>

</body>
</html>
```

7.2.16 DOM 对象

7.2.16.1 概述

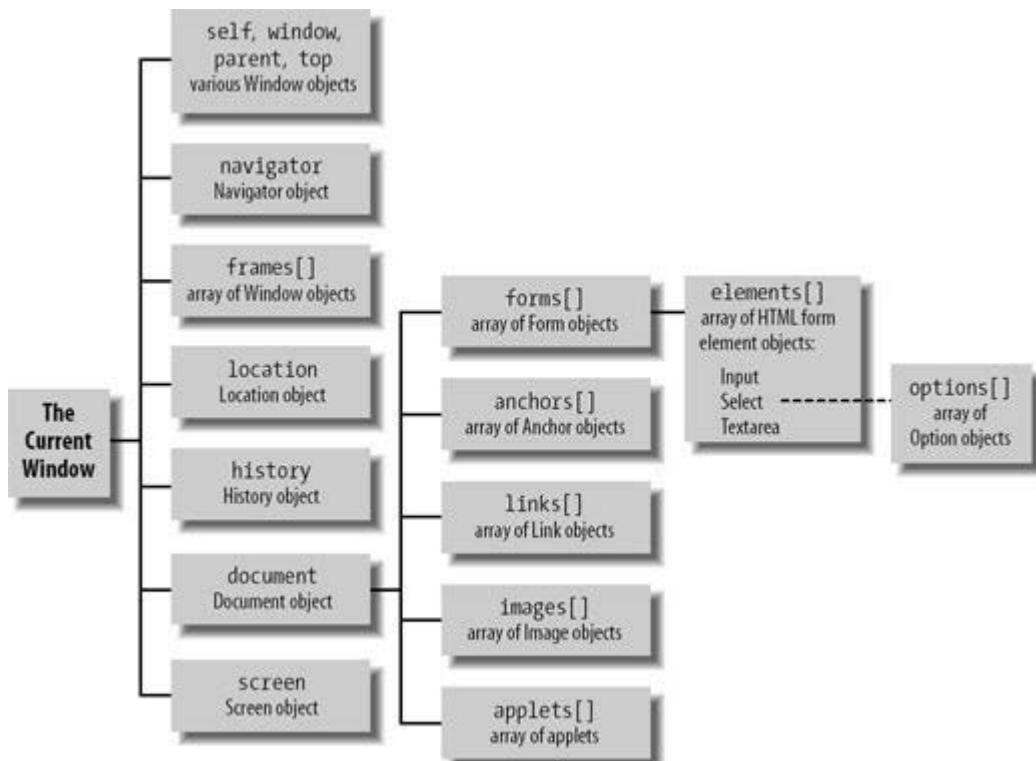
DOM 是“ Document Object Model ”的缩写，简称“ 文档对象模型 ”，由 W3C 制定规范。DOM 定义了 JavaScript 操作 HTML 文档的接口，提供了访问 HTML 文档（如 body、form、div、textarea 等）的途径以及操作方法。

DOM 以“ 树结构 ”来表达 HTML 文档。通过 JavaScript，可以重构整个 HTML 文档，可以添加、移除、改变或重排页面上的内容。要改变 HTML 文档的某项内容，JavaScript 就需要获得访问 HTML 文档的入口。这个入口，连同 HTML 元素以及对 HTML 元素的添加、移动、改变或移除的操作，都是通过 DOM 来获得的。

对于 JavaScript，HTML 文档的每一项内容（HTML 标签、标签属性、标签内的文字、空格或 tab 等）都是一个对象。HTML 文档的标签是一层层嵌套的，最外面的一层是<html>，文档对象模型（DOM）也这样一层层嵌套着，但是通常被理解成一棵树的形状。树根是 document 对象，可以理解为整个 HTML 文档，也是 HTML 文档的入口。树根之下（这棵树的图通常是倒着画，就好像遗传谱系或者家谱那样，树根就是唯一的共同祖先）是子一级的对象，子对象也有它自己的子对象，除了根对象以外，所有的对象都有自己的父对象，同一对象的子对象之间就是兄弟的关系。

在这种由“父子兄弟”组成的“单性繁殖家族图谱树”框架结构中，每项 HTML 文档的内容都可以被确切地定位。文档对象模型（DOM）把整个 HTML 文档组织成这样的一个树状的结构，树结构中的每一项内容都被视为一个节点（node）。包括 JavaScript 在内的各种编程语言都可以通过文档对象模型来访问和改变网页的各种细节。

W3C 已经给文档对象模型（DOM）制定了一系列标准，并且正在制定更多的相关标准。现代浏览器除支持其中的一部分标准之外，还支持某些早在 W3C 标准制定以前就流行了的历史既成的编程接口。也就是说，现代浏览器使用的技术历史由来纷繁复杂，有些人们普遍使用的 DOM 技术并无标准可依。



7.2.16.2 Document 对象的常用方法

getElementById() 返回对拥有指定 id 的第一个对象的引用

getElementsByName() 返回带有指定名称的对象的集合

getElementsByTagName() 返回带有指定标签名的对象的集合

write() 向文档写文本、HTML 表达式或 JavaScript 代码

7.2.16.2.1 getElementById 制作树形菜单

案例 1：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
<head>  
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />  
<title>制作树形菜单</title>  
<style type="text/css">  
    body{font-size:13px;  
         line-height:20px;  
    }  
    a{font-size: 13px;  
      color: #000000;  
      text-decoration: none;
```

```
        }
    a:hover{font-size:13px;
        color: #ff0000;
    }
    img {
        vertical-align: middle;
        border:0;
    }
    .no_circle{list-style:none;
        display:none;
    }

```

</style>

```
<script type="text/javascript">
    function show(d1) {
        if (document.getElementById(d1).style.display == 'block') {
            document.getElementById(d1).style.display = 'none'; //触
动的层如果处于显示状态，即隐藏
        }
        else {
            document.getElementById(d1).style.display = 'block'; //触
动的层如果处于隐藏状态，即显示
        }
    }

```

</script>

```
</head>

<body>
<b>树形菜单: </b>
<ul><a href="#" javascript:onclick=show('art')>
文学艺术</a></ul>

<ul id="art" class="no_circle">
<li>先锋写作</li>
<li>小说散文</li>
<li>诗风词韵</li>
</ul>

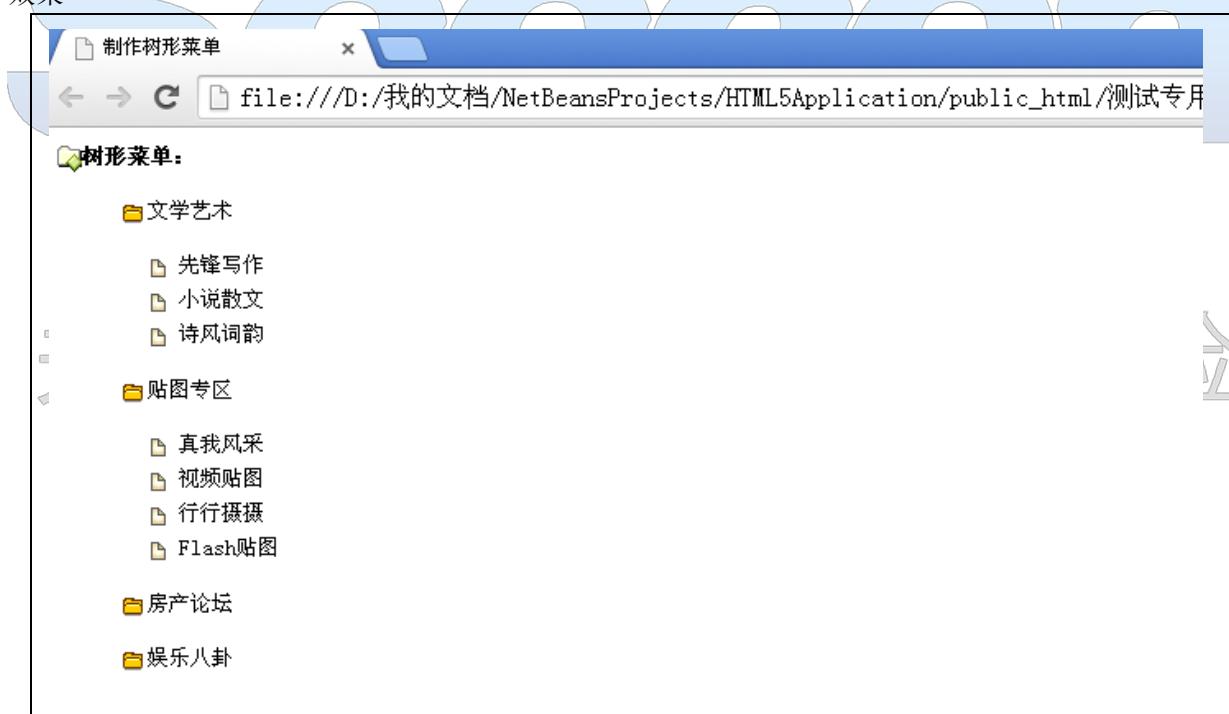
<ul><a href="#" javascript:onclick=show('photo') ">贴图专区</a></ul>
<ul id="photo" class="no_circle">
<li>真我风采</li>
<li>视频贴图</li>
<li>行行摄摄</li>
<li>Flash贴图</li>
</ul>

<ul><a href="#" javascript:onclick=show('home') ">房产论坛</a></ul>

```

```
<ul id="home" class="no_circle">
<li>我要买房</li>
<li>楼市话题</li>
<li>我要卖房</li>
<li>租房心语</li>
<li>二手市场</li>
</ul>
<ul><a href="javascript:onclick=show('game') ">娱乐八卦</a></ul>
<ul id="game" class="no_circle">
<li>红楼一梦</li>
<li>笑话论坛</li>
<li>休闲生活</li>
<li>大话春秋</li>
</ul>
</body>
</html>
```

效果



7.2.16.2.2 getElementById 制作 TAB 切换

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>TAB切换效果</title>
<script type="text/javascript">
    function change(ss) {
        if (ss == "top1") {
            document.getElementById("left1").style.display =
"block";
            document.getElementById("left2").style.display = "none";
            document.getElementById("right1").style.display =
"block";
            document.getElementById("right2").style.display =
"none";
            document.getElementById("end1").style.display = "block";
            document.getElementById("end2").style.display = "none";
        }
        else {
            document.getElementById("left1").style.display = "none";
            document.getElementById("left2").style.display =
"block";
            document.getElementById("right1").style.display =
"block";
            document.getElementById("right2").style.display =
"none";
            document.getElementById("end1").style.display = "none";
            document.getElementById("end2").style.display = "block";
        }
    }
</script>
<style type="text/css">
    img{border:0;}
</style>
</head>
<body>
<table border="0" cellspacing="0" cellpadding="0">
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td colspan="2"></td>
```

```
</tr>
</table>
</body>
</html>
```

效果：



7.2.16.2.3 图片轮播

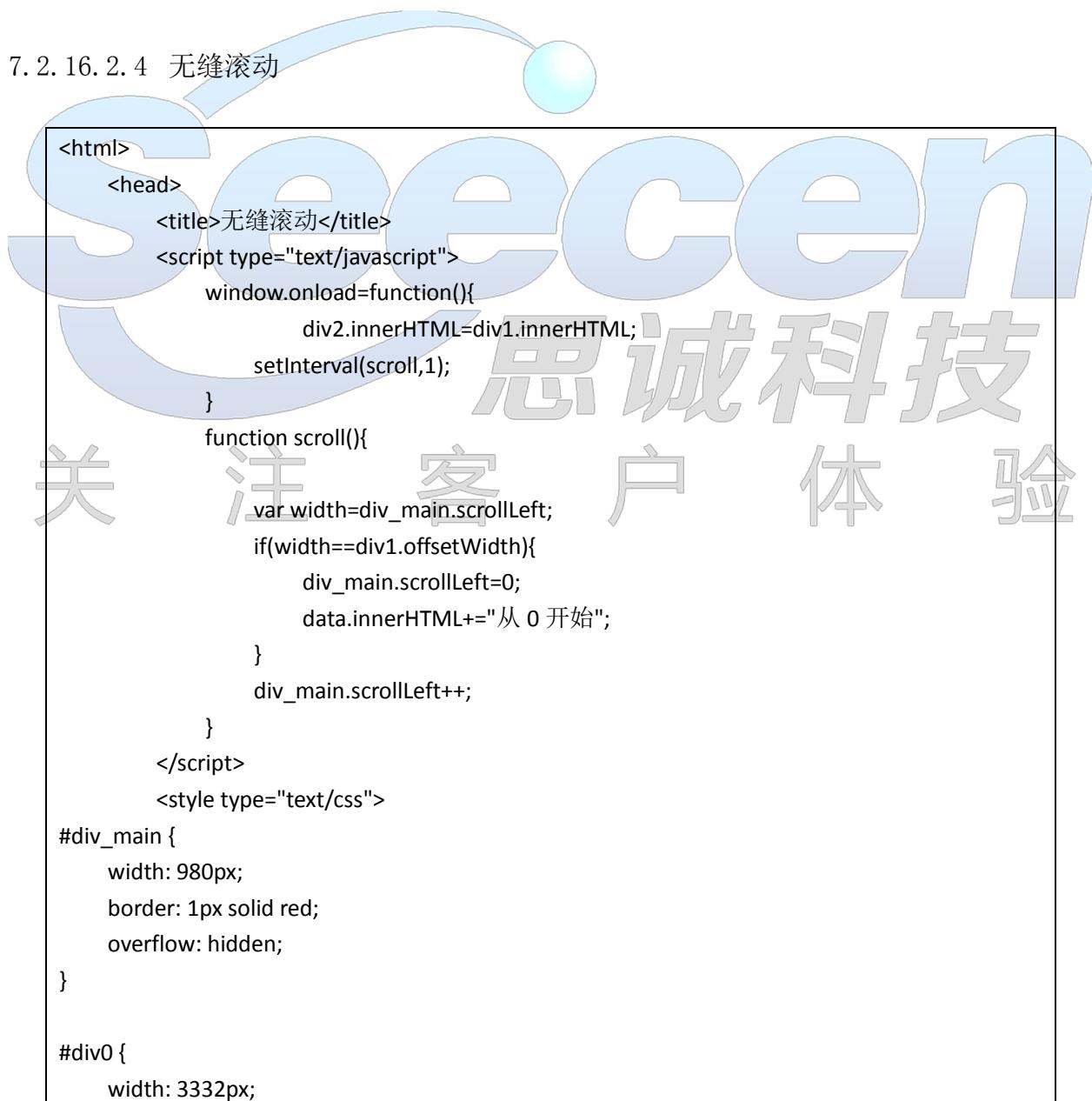
```
<html>
<head>
<title>html02 图片轮播</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<script>
    var NowFrame = 1; // 设置显示
    var MaxFrame = 4; // 设置显示的图片个数
    function show(){
        for(var i=1;i <= MaxFrame;i++){
            if(NowFrame == i){
                document.getElementById("adv" + NowFrame).style.display ='block';
            }else{
                document.getElementById("adv" + i).style.display ='none';
            }
        }
        if(NowFrame == MaxFrame){
            NowFrame =1;
        }else{
            NowFrame++;
        }
    }
    Time = setInterval("show()",1000);
</script>
```

```
</head>
<body onload="show()">




</body>
</html>
```

7.2.16.2.4 无缝滚动



```
display: table;
}

#div1,#div2 {
    width: 1176px;
    float: left;
    display: table;
}

img {
    width: 196px;
    height: 160px;
}

```

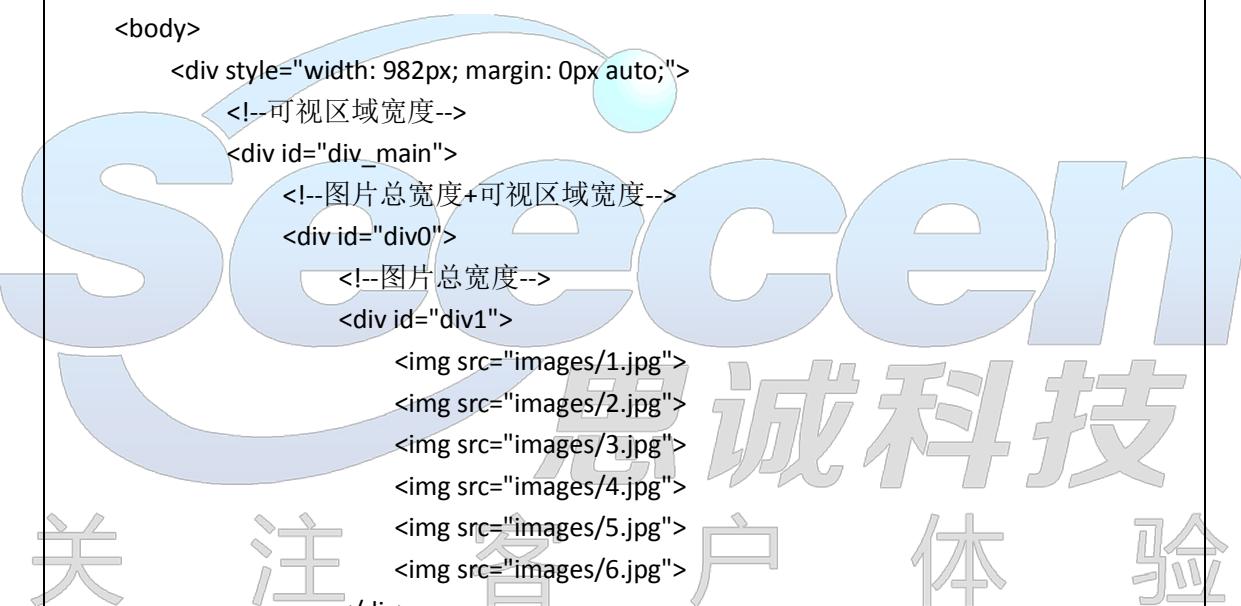
</style>

</head>

<body>

```
<div style="width: 982px; margin: 0px auto;">
    <!--可视区域宽度-->
    <div id="div_main">
        <!--图片总宽度+可视区域宽度-->
        <div id="div0">
            <!--图片总宽度-->
            <div id="div1">
                
                
                
                
                
                
            </div>
            <div id="div2">
                </div>
            </div>
        </div>
        <!--<ul>
            <li>
                
            </li>
            <li>
                
            </li>
            <li>
                
            </li>
        </ul>>
    </div>

```



```
<li>
    
</li>
<li>
    
</li>
<li>
    
</li>
</ul>
-->
</div>
<div id="data">

</div>

</body>
</html>
```

效果：



7.2.16.2.5 广告滚动

```
<html>
    <head>
        <meta content="content-type" http-equiv="charset=utf-8">
        <title>广告滚动</title>
        <script type="text/javascript">
            //隔行改变背景颜色
        </script>
        <style type="text/css">
#slide_adv {
    border: 0px solid red;
    margin: 0px auto;
    width: 586px;
```

```
/*多余的隐藏掉*/
overflow: hidden;
position: relative;
height: 245px;
}

ul {
/*width: 4102px;*/
width:1758px;
height: 245px;
padding: 0px;
margin: 0px;
left: 0px;
top: 0px;
position: absolute;
}

li {
float: left;
border: 0px solid blue;
width: 586px;
height: 243px;
padding: 0px;
margin: 0px;
}

img {
width: 586px;
height: 243px;
}

</style>

<script type="text/javascript">
var imgCount = 3;// 图片数量
var imgWidth = 586; // 每个图片宽度
/**
 * 动画滚动函数
 *
 * @adv 要滚动的层
 * @start 滚动的起始位置
 * @end 滚动的结束位置
 * @speed 滚动速度
 */
function slideadv(adv, start, end, speed) {
if (start < end) {
start += speed;
}
}
```



```
if (start > end) {
    start = end;
}
document.getElementById("data1").innerHTML=start;
adv.scrollLeft = start;
setTimeout(function() {
    slideadv(adv, start, end, speed);
}, 1);
}

if (start > end) {
    start -= speed;
    if (start < end) {
        start = end;
    }
    adv.scrollLeft = start;
    setTimeout(function() {
        slideadv(adv, start, end, speed);
    }, 1);
}

}

function adv() {
    var adv = document.getElementById("slide_adv");
    // 广告图片的统一宽度
    var index = 0; // 自动轮询的图片标示数，标示当前显示到哪一张
    function topSwitch() {
        // alert("a");
        var start = imgWidth * index;
        var end = imgWidth * (++index);
        //如果当前图片张数等于总图片数则用总张数*每张的宽度等于总宽度赋给开始变量
        if (index == imgCount) {
            start = imgWidth * imgCount;
            end = 0;
            index = 0;
        }
        //计算最后一张速递，是前几张的 N(图片数量)倍
        var speed = (Math.abs(end - start) / imgWidth) * 10;
        //var speed =10;
        slideadv(adv, start, end, speed);
        setTimeout(function() {
            topSwitch();
        }, 2000);
    }
    topSwitch();
    //setInterval(topSwitch,3000);
}
```

```
}

//var d=setInterval("adv()",1000);
</script>
</head>
<body style="text-align: center;" onLoad="adv()">
    <div id="slide_adv">
        <ul id="ul1">
            <li>
                <a></a>
            </li>
            <li>
                
            </li>
            <li>
                
            </li>
        </ul>
    </div>
    <div id="data1">
    </div>
    <div id="data2">
    </div>
</body>
</html>
```

关注客户体验

效果：



1172

7.2.16.2.6 图片漂浮

```
<html>
    <head>
        <meta content="Content-type" http-equiv="text/html;charset=utf-8">
        <title>广告图片</title>
        <script type="text/javascript">
            function adv(){
                $("kaola").onmousemove=function(event){
                    //var
                    x=parseInt(event.clientX)+parseInt(document.documentElement.scrollLeft);
                    //var
                    y=parseInt(event.clientY)+parseInt(document.documentElement.scrollTop);
                    $("adv").style.left=event.clientX+"px";
                    $("adv").style.top=event.clientY+"px";
                    $("adv").style.display="block";
                }
                $("kaola").onmouseout=function(event){
                    $("adv").style.display="none";
                }
            }
            function adv1(){
                document.body.onmousemove=function (){
                    var x=event.clientX+parseInt(document.documentElement.scrollLeft);
                    var y=event.clientY+parseInt(document.documentElement.scrollTop);
                    $("adv").style.left=x;
                    $("adv").style.top=y;
                    $("adv").style.display="block";
                }
            }
            function adv2(){
                //alert(window.screen.width);
                //alert(screen.height);
                $("adv2").style.left=Math.floor(Math.random()*(screen.width-200));
                $("adv2").style.top=Math.floor(Math.random()*(screen.height-295));
            }
            setInterval(adv2,1000);

            /*漂浮广告*/
            //定义全局变量
            var moveX = 0;          //X 轴方向移动的距离
            var moveY = 0;          //Y 轴方向移动的距离
            var step = 5;           //图片移动的速度
            var directionY = "下";   //设置图片在 Y 轴的移动方向
        </script>
    </head>
    <body>
        
        <div id="adv" style="position: absolute; left: 0; top: 0; width: 100%; height: 100%;">
```

```
var directionX = "右"; //设置图片在 X 轴的移动方向
function adv3(){
    var img = $("adv3"); //图片所在层 ID
    var width = document.documentElement.clientWidth; //浏览器宽度
    // alert(width+"=="+screen.width);
    var height = document.documentElement.clientHeight; //浏览器高度
    // alert(height+"=="+screen.height);
    var imgHeight=$("#lh").height(); //漂浮图片高度
    var imgWidth=$("#lh").width(); //漂浮图片宽度
    img.style.left =parseInt(moveX + document.documentElement.scrollLeft)+"px";
    //漂浮图片距浏览器左侧位置
    img.style.top = parseInt(moveY + document.documentElement.scrollTop)+"px";
    //漂浮图片距浏览器顶端位置
    //alert(img.style.left);
    if (directionY=="下"){
        moveY = moveY + step; //漂浮图片在 Y 轴方向上向下移动
    }else{
        moveY = moveY - step; //漂浮图片在 Y 轴方向上向上移动
    }
    if (moveY < 0){ //如果漂浮图片漂到浏览器顶端时, 设置图片在 Y 轴方向上向下
        directionY = "下";
        moveY = 0;
    }
    //如果漂浮图片漂到浏览器底端时, 设置图片在 Y 轴方向上向上移动
    if (moveY >= (height - imgHeight)) {
        directionY = "上";//改变方向, 让图片往上走
        //同时高度赋给 Y 轴移动的距离
        moveY = (height - imgHeight);
    }
}

if (directionX=="右"){
    //漂浮图片在 X 轴方向上向右移动
    moveX = moveX + step;
}else {
    //漂浮图片在 X 轴方向上向左移动
    moveX = moveX - step;
}
//如果漂浮图片漂到浏览器左侧时, 设置图片在 X 轴方向上向右移动
if (moveX < 0) {
    directionX = "右";
    moveX = 0;
}
//如果漂浮图片漂到浏览器右侧时, 设置图片在 X 轴方向上向左移动
if (moveX >= (width - imgWidth)) {
```

关注

客户体验

客户

客户

客户

```
        directionX = "左";
        moveX = (width - imgWidth);
    }
}
setInterval("adv3()",30);

function $(id){
    return document.getElementById(id);
}
</script>
<style type="text/css">
body{
    margin: 0px;
    padding: 0px;
}
#adv {
    display: none;
    position: absolute;
}
#adv2,#adv3 {
    position: absolute;
}
</style>
</head>
<body onload="adv();">
    
    <div id="adv">
        
    </div>

    <div id="adv2" style="z-index: 100">
        
    </div>
    <div id="adv3" style="z-index: 200">
        
    </div>

</body>
</html>
```

关注

客户

体验

7.2.16.2.7 动态表格 1

insertCell() 方法用于在 HTML 表的一行的指定位置插入一个空的 <td> 元素。

deleteCell() 方法用于删除表格行中的单元格 (<td> 元素)。

insertRow() 方法用于在表格中的指定位置插入一个新行。

deleteRow() 方法用于从表格删除指定位置的行。

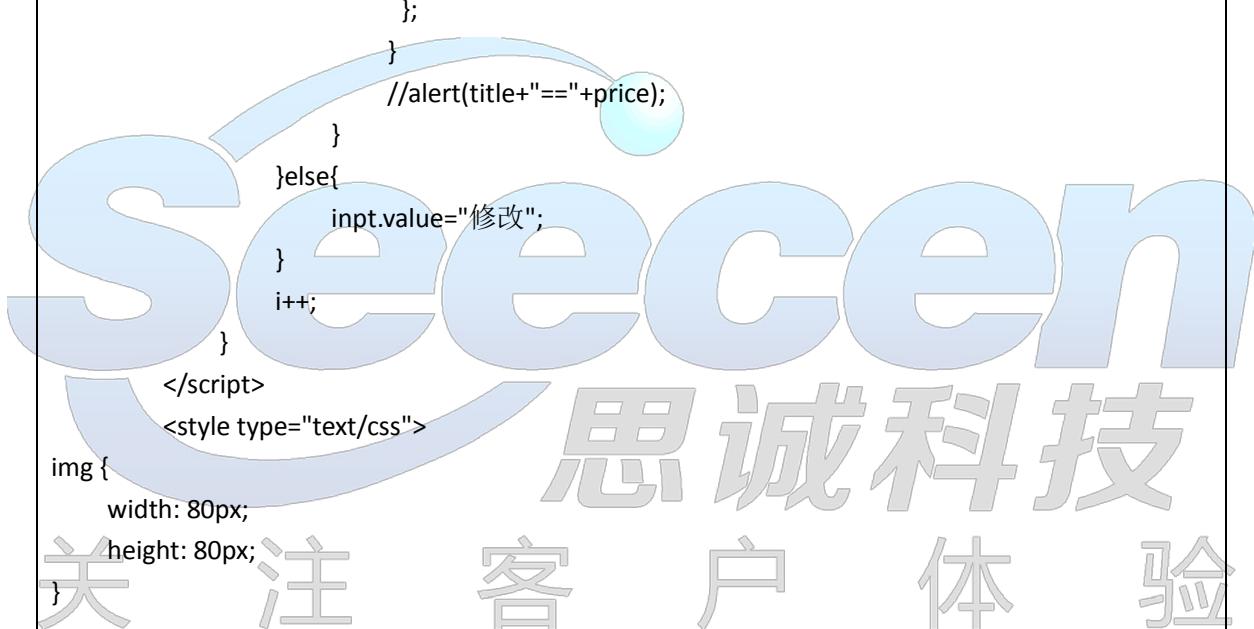
rowIndex 属性返回某一行在表格的行集合中的位置 (row index)。

父节点 (parentNode)。

```
<html>
  <head>
    <title>订单页面</title>
    <script type="text/javascript">
      var count=0;
      function addRow(){
        var tb1=document.getElementById("tb1");
        var tr=tb1.insertRow(tb1.rows.length-1);
        var td=tr.insertCell(0);
        td.innerHTML=<img src='dn.jpg' />;
        td=tr.insertCell(1);
        td.innerHTML="Mac Book Air"+(count++);
        td=tr.insertCell(2);
        td.innerHTML="9800";
        td=tr.insertCell(3);
        td.innerHTML=<input type='button' value='删除' onclick='delRow(this);' />;
        td.innerHTML+="<input type='button' value='修改' onclick='upRow(this);' >";
        //alert(tb1);
      }
      function delRow(obj){
        var index=obj.parentNode.parentNode.rowIndex;
        document.getElementById("tb1").deleteRow(index);
      }
      var i=0;
      function upRow(obj){
        var tr=obj.parentNode.parentNode;
        //alert(tr.cells.length);
        var v=tr.cells[1].innerHTML;
        tr.cells[1].innerHTML=<input id='title"+i+"' value='"+v.trim()+"' />;
        v=tr.cells[2].innerHTML;
        tr.cells[2].innerHTML=<input id='price"+i+"' value='"+v.trim()+"' />;
        v=tr.cells[3].innerHTML;
        var inpt=tr.cells[3].children[1];
        if(inpt.value=="修改"){
          inpt.value="删除";
        } else {
          inpt.value="修改";
        }
      }
    </script>
  </head>
  <body>
    <table border="1" id="tb1">
      <tr>
        <td><img src='dn.jpg' /></td>
        <td>Mac Book Air</td>
        <td>9800</td>
        <td><input type='button' value='删除' onclick='delRow(this);' /></td>
        <td><input type='button' value='修改' onclick='upRow(this);' ></td>
      </tr>
    </table>
  </body>
</html>
```

```
inpt.value="确认";
inpt.onclick=function(){
    var cel1=this.parentNode.parentNode.cells[1];
    var cel2=this.parentNode.parentNode.cells[2];
    var cel3=this.parentNode.parentNode.cells[3];
    var title=cel1.children[0].value;
    var price=cel2.children[0].value;
    cel1.innerHTML=title;
    cel2.innerHTML=price;
    alert('asdfdsaf');
    if(inpt.value=="确认"){
        cel3.children[1].value="修改";
        cel3.children[1].onclick=function(){
            upRow(this)
        };
    }
    //alert(title+"=="+price);
}
}else{
    inpt.value="修改";
    i++;
}
}
</script>
<style type="text/css">
img {
    width: 80px;
    height: 80px;
}

```



```
thead,tbody,tfoot {
    background-color: white;
}
</style>
</head>
<body>
<table style="background-color: #ccc; cellspacing="1" id="tb1">
    <thead>
        <tr>
            <td>
                商品图片
            </td>
            <td>
                商品名称
            </td>

```

```
<td>
    商品价格
</td>
<td>
    操作
</td>
</tr>
</thead>
<tbody>
<tr id="row0">
    <td>
        
    </td>
    <td>
        Mac Book Pro
    </td>
    <td>
        12900
    </td>
    <td>
        <input type="button" value="删除" onclick="delRow(this);"/>
        <input type="button" value="修改" onclick="upRow(this)"/>
    </td>
</tr>
</tbody>
<tfoot>
<tr>
    <td colspan="4" align="center">
        <input type="button" value="添加一行" onclick="addRow()"/>
    </td>
</tr>
</tfoot>
</table>
</body>
</html>
```

效果：

商品图片	商品名称	商品价格	操作
	Mac Book Pro	12900	删除 修改
添加一行			

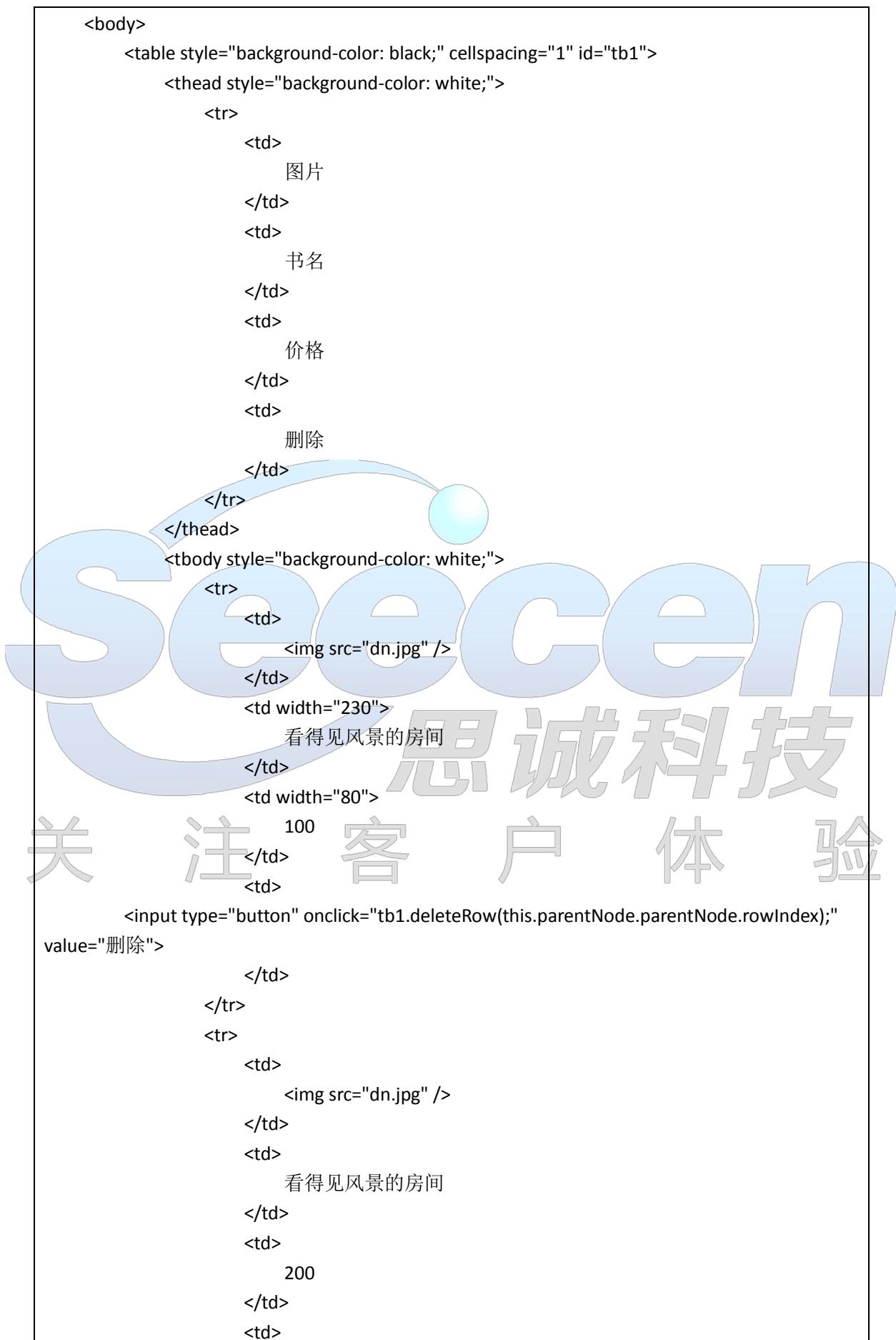
7.2.16.2.8 动态表格 2

```
<html>
  <head>
    <title>DOM 操作表格</title>
    <script type="text/javascript">
      function chgTitle(){
        var tb=document.getElementById("tb1");
        //alert(tb.rows[0].cells[0].innerHTML);
        //alert(tb.rows[0].cells.length);
        //alert(tb.firstChild.firstChild.firstChild.innerHTML);//IE
        //alert(tb.children[0].children[0].children[0].innerHTML);//Chrome IE
        var tds=tb.children[0].children[0].children;
        for(var i=0;i<tds.length;i++){
          /*tds[i].style.textAlign="center";
          tds[i].style.fontWeight="bold";
          tds[i].style.backgroundColor="#ccc";*/
          tds[i].className="title";
        }
      }

      function chgTitle1(){
        var cells1=tb1.rows[0].cells;
        for(var i=0;i<cells1.length;i++){
          if(cells1[i].className=='title'){
            cells1[i].className="";
          }else{
            cells1[i].className='title';
          }
        }
        //获取表格的所有行
        var rows1=tb1.rows;
```

```
for(var i=1;i<rows1.length;i++){
    //获取从第二行开始的每行的第一列
    var cell1=rows1[i].cells[0];
    //获取第一列的第一个子元素
    var img=cell1.children[0];
    //设置图片的 src 属性
    //img.setAttribute("src","Koala.jpg");
    //img.src="Koala.jpg";
    if(img.src.indexOf('Koala.jpg')>-1){
        img.src="dn.jpg";
    }else{
        img.src='Koala.jpg';
    }
}
var count=0;
function addRow(){
    var tr=tb1.insertRow(2);
    var td=tr.insertCell(0);
    td.innerHTML=<img src='dn.jpg' />;
    td=tr.insertCell(1);
    td.innerHTML="Mac Book Pro"+count++;
    td=tr.insertCell(2);
    td.innerHTML="12900";
    td=tr.insertCell(3);
    td.innerHTML=<input type='button' value='删除
    onclick='delRow(this.parentNode.parentNode.rowIndex)'/>;
}
function delRow(index){
    tb1.deleteRow(index);
}
</script>
<style type="text/css">
.title {
    text-align: center;
    font-weight: bold;
    background-color: #ccc;
}

img {
    width: 80px;
    height: 80px;
}
</style>
</head>
```



```
<input type="button" onclick="tb1.deleteRow(this.parentNode.parentNode.rowIndex);"
value="删除">
</td>
</tr>
<tr>
<td>

</td>
<td>
60 个瞬间
</td>
<td>
300
</td>
<td>
<input type="button" onclick="tb1.deleteRow(this.parentNode.parentNode.rowIndex);"
value="删除">
</td>
</tr>
<tr>
<td>

</td>
<td>
60 个瞬间
</td>
<td>
400
</td>
<td>
<input type="button" onclick="tb1.deleteRow(this.parentNode.parentNode.rowIndex);"
value="删除">
</td>
</tr>
</tbody>
</table>
<input type="button" value="修改标题" onclick="chgTitle1()">
<input type="button" value="添加一行" onclick="addRow()">
<input type="button" value="删除一行" onclick="delRow()">
</body>
</html>
```

效果：

图片	书名	价格	删除
	看得见风景的房间	100	<input type="button" value="删除"/>
	看得见风景的房间	200	<input type="button" value="删除"/>
	60个瞬间	300	<input type="button" value="删除"/>
	60个瞬间	400	<input type="button" value="删除"/>

7.2.16.2.9 getElementById 变幻颜色

```
<html>
<head>
<title>onchange</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<style type="text/css">
    #a{width:100%;height:520px;}
</style>
<script language="javascript">
    function f(ccc) {
        document.getElementById("a").style.background = ccc;
        setTimeout("fff('#00ff00')", 700)
    }
    function fff(sdfsdf) {
        document.getElementById("a").style.background = sdfsdf;
        setTimeout("f('#ff0000')", 700)
    }
</script>
</head>
<body>
```

```
<div id="a"><input type="button" value="点击变幻颜色" onclick="f('#000000');"></div>
</body>
</html>
```

效果：



7.2.16.2.10 全选简版：

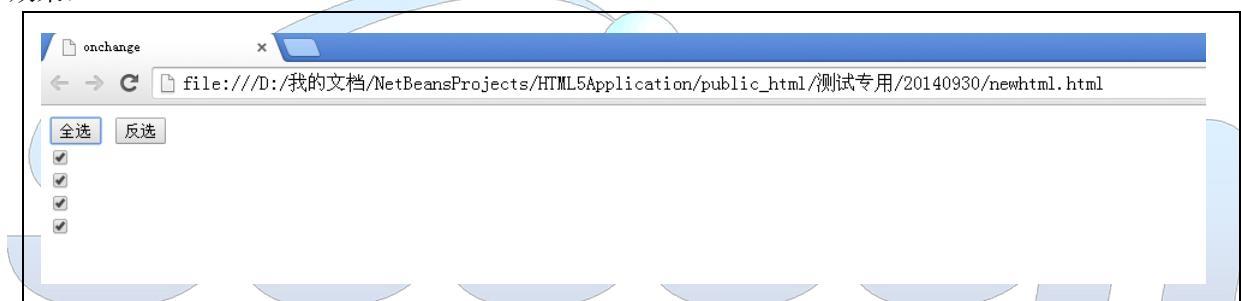
思诚科技

```
<html>
<head>
<title>onchange</title>
<meta http-equiv="content-type" content="text/html; charset=UTF-8">
<script language="javascript">
    function quanxuan() {
        var aaaaa = document.getElementsByName("aihao");
        for (var i = 0; i < aaaaa.length; i++) {
            aaaaa[i].checked = "checked";
        }
    }
    function quanbuxuan() {
        var aaaaa = document.getElementsByName("aihao");
        for (var i = 0; i < aaaaa.length; i++) {
            if (aaaaa[i].checked == true) {
                aaaaa[i].checked = "";
            } else {
                aaaaa[i].checked = "checked";
            }
        }
    }
</script>

```

```
        }
    }
</script>
</head>
<body>
<input type="button" value="全选" onclick="quanxuan()" >
<input type="button" value="反选" onclick="quanbuxuan()" ><br />
<input type="checkbox" value="1" name="aihao"><br />
<input type="checkbox" value="2" name="aihao"><br />
<input type="checkbox" value="3" name="aihao"><br />
<input type="checkbox" value="4" name="aihao"><br />
</body>
</html>
```

效果：



7.2.16.2.11 getElementByName 制作全选/全不选

案例 1:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>全选/全不选效果</title>
<style type="text/css">
.bg{
    background-image: url(images/list_bg.gif);
    background-repeat: no-repeat;
    width: 730px;
}
td{text-align:center;
font-size:13px;
line-height:25px;
}
body{margin:0}
```

```
</style>
<script type="text/JavaScript">
    function check(){
        var oInput=document.getElementsByName( "product" );
        for ( var i=0;i<oInput.length;i++){
            if ( document.getElementById("all").checked==true){
                oInput[i].checked=true;
            }
            else {
                oInput[i].checked=false;
            }
        }
    }
</script>
</head>
<body><table border="0" cellspacing="0" cellpadding="0" class="bg">
<form action="" method="post">
<tr>
<td style="height:40px;">&ampnbsp</td>
<td>&ampnbsp</td>
<td>&ampnbsp</td>
<td>&ampnbsp</td>
</tr>
<tr style="font-weight:bold;">
<td><input id="all" type="checkbox" value="全选" onclick="check();" />全选
</td>
<td>商品图片</td>
<td>商品名称/出售者/联系方式</td>
<td>价格</td>
</tr>
<tr>
<td colspan="4"><hr style="border:1px #CCCCCC dashed" /></td>
</tr>
<tr>
<td><input name="product" type="checkbox" value="1" /></td>
<td></td>
<td>杜比环绕，家庭影院必备，超真实享受<br />
出售者: ling112233<br />
&ampnbsp&ampnbsp
收藏</td>
<td>一口价<br />
2833.0 </td>
</tr>
<tr>
<td colspan="4"><hr style="border:1px #CCCCCC dashed" /></td>
```

```
</tr>
<tr>
<td><input name="product" type="checkbox" value="2" /></td>
<td></td>
<td>NVIDIA 9999GT 512MB 256bit 极品显卡, 不容错过<br />
出售者: aipiaopiao110 <br />
&nbsp;&nbsp;
    收藏</td>
<td>一口价<br />
    6464.0 </td>
</tr>
<tr>
    <td colspan="4"><hr style="border:1px #CCCCCC dashed" /></td>
</tr>
<tr>
<td><input name="product" type="checkbox" value="3" /></td>
<td></td>
<td>精品热卖: 高清晰, 30寸等离子电视<br />
出售者: 阳光的挣扎<br />
&nbsp;&nbsp;
    收藏</td>
<td>一口价<br />
    18888.0 </td>
</tr>
<tr>
    <td colspan="4"><hr style="border:1px #CCCCCC dashed" /></td>
</tr>
<tr>
<td><input name="product" type="checkbox" value="4" /></td>
<td></td>
<td>Sony 索尼家用最新款笔记本<br />
出售者: 疯狂的镜无<br />
&nbsp;&nbsp;
    收藏</td>
<td>一口价<br />
    5889.0 </td>
</tr>
<tr>
    <td colspan="4"><hr style="border:1px #CCCCCC dashed" /></td>
</tr>
</form>
</table>
</body>
</html>
```

效果:

The screenshot shows a web browser window displaying a product listing table. The table has columns for '商品图片' (Product Picture), '商品名称/出售者/联系方式' (Product Name/Seller/Contact Information), and '价格' (Price). There are four rows of data:

商品图片	商品名称/出售者/联系方式	价格
	杜比环绕, 家庭影院必备, 超真实享受 出售者: ling112233 和我联系	一口价 2833.0
	NVIDIA 9999GT 512MB 256bit极品显卡, 不容错过 出售者: aipiaopiao110 和我联系	一口价 6464.0
	精品热卖: 高清晰, 30寸等离子电视 出售者: 阳光的挣扎 和我联系	一口价 18888.0
	Sony索尼家用最新款笔记本 出售者: 疯狂的镜无 和我联系	一口价 5889.0

7.2.16.3 元素的隐藏和显示

7.2.16.3.1 visibility 属性的值 隐藏后位置仍占用

visible 表示元素是可见的

hidden 表示元素是不可见的

语法:

```
object.style.visibility="值"
```

7.2.16.3.2 display 属性的值 隐藏后位置不占用

none 表示此元素不会被显示

block 表示此元素将显示为块级元素, 此元素前后会带有换行符

语法:

```
object.style.display="值"
```

7.2.17 表单基本验证技术

7.2.17.1 案例 1：函数验证

```
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="utf-8" />
<title></title>
<style type="text/css">
</style>
<script type="text/javascript">
    function formReset() {
        document.getElementById("name").focus();
    };
    function isNumber(str) {
        return String(str) == window.parseInt(str).toString();
    };
    function formSubmit() {
        var name = document.getElementById("name");
        if (!name.value) {
            alert("请输入姓名");
            name.focus();
            return false;
        }
        var age = document.getElementById("age");
        if (!isNumber(age.value)) {
            alert("请输入年龄(年龄必须是一个整数)");
            age.focus();
            return false;
        }
        var sex = document.getElementsByName("sex"),
            flag = false;
        for (var i = 0; i < sex.length; i++) {
            if (sex[i].checked) {
                flag = true;
                break;
            }
        }
        if (!flag) {
            alert("请选择性别");
            sex[0].focus();
            return false;
        }
    }
</script>

```

关注

客户

体验

思诚科技

```
var likes = document.getElementsByName("likes"),
    min = 2, counts = 0;
for (var i = 0; i < likes.length; i++) {
    if (likes[i].checked) {
        counts++;
    }
}
if (counts < min) {
    alert("请选择爱好(最少选择 " + min + " 项)");
    likes[0].focus();
    return false;
}
};

</script>
</head>
<body>
<form action="javascript-form-validate.html" method="get"
      onreset="javascript: formReset();"
      onsubmit="javascript: return formSubmit();">
<table>
<tr>
<td>姓名: </td>
<td><input id="name" name="name" type="text" /></td>
</tr>
<tr>
<td>年龄: </td>
<td><input id="age" name="age" type="text" /></td>
</tr>
<tr>
<td>性别: </td>
<td>
<input id="radio0" name="sex" type="radio" value="0" />
<label for="radio0">女</label>
<input id="radio1" name="sex" type="radio" value="1" />
<label for="radio1">男</label>
</td>
</tr>
<tr>
<td>爱好: </td>
<td>
<input id="chk0" name="likes" type="checkbox" value="0" />
<label for="chk0">女</label>
<input id="chk1" name="likes" type="checkbox" value="1" />
<label for="chk1">男</label>
<input id="chk2" name="likes" type="checkbox" value="2" />

```

```
<label for="chk2">唱歌</label>
<input id="chk3" name="likes" type="checkbox" value="3" />
<label for="chk3">跳舞</label>
</td>
</tr>
<tr>
<td colspan="2">
<input type="submit" value="提交" />
<input type="reset" value="重置" />
</td>
</tr>
</table>
</form>
</body>
</html>
```

效果:



文本框对象:

○ 文本框对象的属性、方法和事件		
类别	名称	描述
事件	onblur	失去焦点, 当光标离开某个文本框时触发
	onfocus	获得焦点, 当光标进入某个文本框时触发
	onkeypress	某个键盘按键被按下并松开
方法	blur()	从文本域中移开焦点
	focus()	在文本域中设置焦点, 即获得鼠标光标
	select()	选取文本域中的内容
属性	id	设置或返回文本域的id
	value	设置或返回文本域的value属性的值

7.2.17.2 案例 2：简单的方法验证

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>休闲网登录页面</title>
<link href="login.css" rel="stylesheet" type="text/css">
<script type="text/javascript">
    function check() {
        var mail = document.getElementById("email").value;
        if (mail == "") {//检测Email是否为空
            alert("Email不能为空");
            return false;
        }
        if (mail.indexOf "@" == -1) {
            alert("Email格式不正确\n必须包含@");
            return false;
        }
        if (mail.indexOf "." == -1) {
            alert("Email格式不正确\n必须包含 .");
            return false;
        }
        return true;
    }
</script>
</head>
<body>
<div id="header" class="main">
<div id="headerRight">注册 | 登录 | 帮助</div>
</div>
<div class="main">
<table id="center" border="0" cellspacing="0" cellpadding="0">
<tr>
<td></td></tr>
<tr>
<td class="bg"><table width="100%" border="0" cellspacing="0" cellpadding="0">
<tr>
<td class="bold">登录休闲网</td>
</tr>
<form action="success.html" method="post" name="myform" onsubmit="return
check()"><tr>
```

```
<td>Email: <input id="email" type="text" class="inputs"/></td>
</tr>
<tr>
<td>&nbsp;密码: <input id="pwd" type="password" class="inputs"/></td>
</tr>
<tr>
<td style="height:35px; padding-left:30px;"><input name="btn" type="submit" value="登录" class="rb1" /></td>
</tr></form>
</table>
</td>
</tr>
<tr>
<td></td></tr>
</table>
</div>
<div id="footer" class="main"><a href="#">关于我们</a> | <a href="#">诚聘英才</a> | <a href="#">联系方式</a> | <a href="#">帮助中心</a></div>
</body>
</html>
```



效果:



7.2.17.3 案例 3: onfocus 事件校验:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>休闲网注册页面</title>
<script type="text/javascript">
    function $(ElementID) {
        return document.getElementById(ElementID);
    }
    function checkEmail() {
        var mail = $("email");
        var divID = $("DivEmail");
        divID.innerHTML = "";
        if (mail.value == "") {
            divID.innerHTML = "Email不能为空";
            //mail.focus();
            return false;
        }
        if (mail.value.indexOf("@") == -1) {
            divID.innerHTML = "Email格式不正确，必须包含@";
            //mail.focus();
            return false;
        }
        if (mail.value.indexOf(".") == -1) {
            divID.innerHTML = "Email格式不正确，必须包含。";
            //mail.focus();
            return false;
        }
        //return true;
    }
    function checkPass() {
        var pwd = $("pwd");
        var divID = $("DivPwd");
        divID.innerHTML = "";
        if (pwd.value == "") {
            divID.innerHTML = "密码不能为空";
            //pwd.focus();
            return false;
        }
        if (pwd.value.length < 6) {
            divID.innerHTML = "密码必须等于或大于6个字符";
            //pwd.focus();
            return false;
        }
    }
</script>
</head>
<body>
    <div id="DivEmail"><input type="text" id="email" name="email" value="" /></div>
    <div id="DivPwd"><input type="password" id="pwd" name="pwd" value="" /></div>
    <div style="text-align: center; margin-top: 20px;">
        <input type="button" value="提交" onclick="checkEmail(); checkPass();"/>
    </div>
</body>

```

```
        }
        //return true;
    }
}

function checkRePass() {
    var pwd = $("pwd"); //输入密码
    var repwd = $("repwd"); //再次输入密码
    var divID = $("DivRepwd");
    divID.innerHTML = "";
    if (pwd.value != repwd.value) {
        divID.innerHTML = "两次输入的密码不一致";
        return false;
    }
    //return true;
}
function checkUser() {
    var user = $("user");
    var divId = $("DivUser");
    divId.innerHTML = "";
    if (user.value == "") {
        divId.innerHTML = "姓名不能为空";
        //user.focus();
        return false;
    }
    for (var i = 0; i < user.value.length; i++) {
        var j = user.value.substring(i, i + 1)
        if (j >= 0) {
            divId.innerHTML = "姓名中不能包含数字";
            //user.focus();
            return false;
        }
    }
    //return true;
}

</script>
</head>
<body>
<div id="header" class="main">
<div id="headerRight">注册 | 登录 | 帮助</div>
</div>
<div class="main">
<table id="center" border="0" cellspacing="0" cellpadding="0">
<tr>
<td class="bold" colspan="2">注册休闲网</td>
</tr>
<form action="" method="post" name="myform"><tr>
```

```
<td class="left">您的Email: </td>
<td><input id="email" type="text" class="inputs" onblur="checkEmail()" />
<div class="red" id="DivEmail"></div></td>
</tr>
<tr>
<td class="left">输入密码: </td>
<td><input id="pwd" type="password" class="inputs" onblur="checkPass()" />
<div class="red" id="DivPwd"></div></td>
</tr>
<tr>
<td class="left">再输入一遍密码: </td>
<td><input id="repwd" type="password" class="inputs"
onblur="checkRePass()" />
<div class="red" id="DivRepwd"></div></td>
</tr>
<tr>
<td class="left">您的姓名: </td>
<td><input id="user" type="text" class="inputs" onblur="checkUser()" />
<div class="red" id="DivUser"></div></td>
</tr>
<tr>
<td class="left">性别: </td>
<td><input name="sex" type="radio" value="1" />男
<input name="sex" type="radio" value="0" />女</td>
</tr>
<tr>
<td class="left">出生日期: </td>
<td><select name="year">
<script type="text/javascript">
for (var i = 1900; i <= 2009; i++) {
    document.write(" <option value=" + i + ">" + i + "</option> ");
}
</script>
</select>年
<select name="month">
<script type="text/javascript">
for (var i = 1; i <= 12; i++) {
    document.write(" <option value=" + i + ">" + i + "</option> ");
}
</script>
</select>月
<select name="day">
<script type="text/javascript">
for (var i = 1; i <= 31; i++) {
    document.write(" <option value=" + i + ">" + i + "</option> ");
}
```

```
}

</script>
</select>日</td>
</tr>
<tr><td>&nbsp;</td>
<td ><input name="btn" type="submit" value="注册" class="rb1" /></td>
</tr></form>
</table>
</div>
<div id="footer" class="main"><a href="#">关于我们</a> | <a href="#">诚聘  
英才</a> | <a href="#">联系方式</a> | <a href="#">帮助中心</a></div>
</body>
</html>
```

效果：



作业：

- 1、设置没 5 秒回弹出小框页面（模拟病毒）



2、使用函数实现关闭左上角按钮



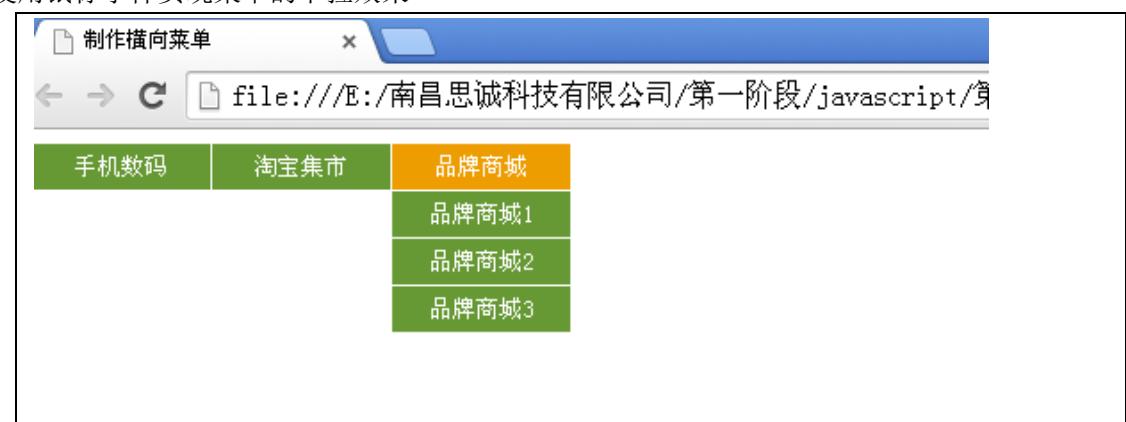
3、使用函数实现菜单的闭合



4、设置定时器，实现 4 张图片轮换显示的广告效果



5、使用鼠标事件实现菜单的下拉效果



6、全选、全不选、单选的效果

A screenshot of a web browser window titled "框选中效果". The address bar shows "file:///E:/南昌思诚科技有限公司/第一阶段/javascript/第3天/作业5/list.html". The page displays a table with a header row containing checkboxes for "全选" (Select All), "产品名称" (Product Name), "价格(元)" (Price), and "数量" (Quantity). Below the header are seven data rows, each with a checkbox. A button at the bottom of the table says "删除选中的产品" (Delete Selected Product).

<input checked="" type="checkbox"/> 全选	产品名称	价格(元)	数量
<input checked="" type="checkbox"/>	诺基亚N85手机	2589	6
<input checked="" type="checkbox"/>	佳能IXUS95ISV数码相机	1865	5
<input checked="" type="checkbox"/>	戴尔新版SK8115键盘	60	56
<input checked="" type="checkbox"/>	联想折叠式笔记本电脑桌	59	10
<input checked="" type="checkbox"/>	康佳32英寸液晶电视	2945	3
<input checked="" type="checkbox"/>	九阳JYDX-78D五谷系列豆浆机	299	8

7、js 实现企业邮箱登陆验证



关注客户体验

8、注册校验，结果显示在栏目右侧，显示红色字体

A screenshot of a web browser window titled "泡泡网 PCPOP.COM". The address bar shows "file:///E:/南昌思诚科技有限公司/第一阶段/javascript/第4天/作业3/register.html". The page has a "用户注册" (User Registration) section with a "填写信息" (Fill Information) button. It includes fields for "用户名" (Username), "密码" (Password), "性别" (Gender), "出生日期" (Date of Birth), and "电子邮箱地址" (Email Address). Error messages are displayed in red: "密码长度为在6-12字符" (Password length is 6-12 characters) next to the password field, and "请选择性别" (Please select gender) next to the gender selection buttons. At the bottom, there are "注册" (Register) and "清除" (Clear) buttons.

9、下侧 5 个图片鼠标事件触发上侧图片变更 (onmouseover onmouseout)



10、使用表格及 DOM 实现 TAB 切换效果

图书周排行榜 TOP 100
近7天销量，每日更新

小说 非小说 少儿

1. 斯凯瑞金色童书 · ...
2. 哈利·波特与“混...”
3. 不一样的卡梅拉（...）
4. 它们是怎么来的
5. 五·三班的坏小子...
6. 男生日记
7. 哈利·波特与魔法石
8. 嘴里啪啦丛书(全7册)

大 江 各 户 体 独

第 8 章 JAVA 语言基础

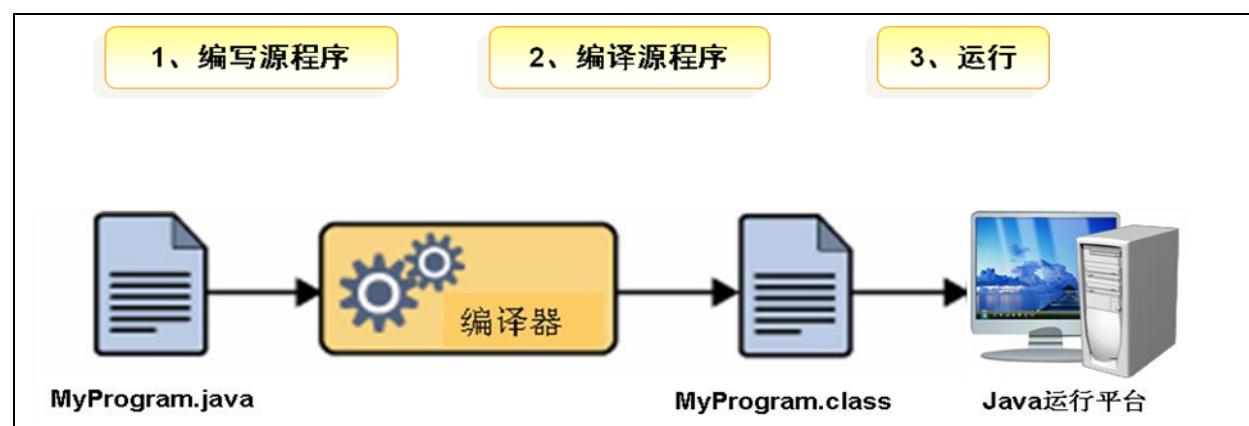
8.1 本章目标

- 了解什么是程序
- 了解 Java 的历史、现状和发展
- 了解 Java 的优点及特性
- 掌握 Myeclipse 基本菜单的使用
- 掌握 JDK 的安装和环境变量的配置（重点）
- 掌握 Java 程序的结构及编写规范
- 掌握 Java 中的注释方法以及关键字、标示符、变量、常量等概念
- 掌握 Java 中的数据类型及其转换方式
- 掌握 Java 的各种运算符的含义、优先级和结合性
- 掌握判断、循环语句的使用方法，并可以编写简单的 Java 程序
- 掌握数组的声明、初始化、元素引用并会解决简单的实际问题

8.2 内容

8.2.1 程序的定义

- Java 是 Sun Microsystems 于 1995 年推出的高级编程语言
 - Java 领域的 JavaSE、JavaEE 技术已发展成为同 C# 和 .NET 平分天下的应用软件开发平台和技术
 - JavaSE: Java Platform, Standard Edition 标准版比如：数据库连接、接口定义、输入/输出、网络编程；
 - JavaEE: Java Platform, Enterprise Edition 企业版比如：EJB、servlet、JSP、XML、事务控
 - JavaME: Java Platform, Micro Edition 微缩版比如：手机、智能卡、PDA、机顶盒
- 程序一词来自生活，通常指完成某些事务的一种既定方式和过程；
在日常生活中，可以将程序看成对一系列动作的执行过程的描述；
开发 Java 程序的三步走：



8.2.2 开发环境配置

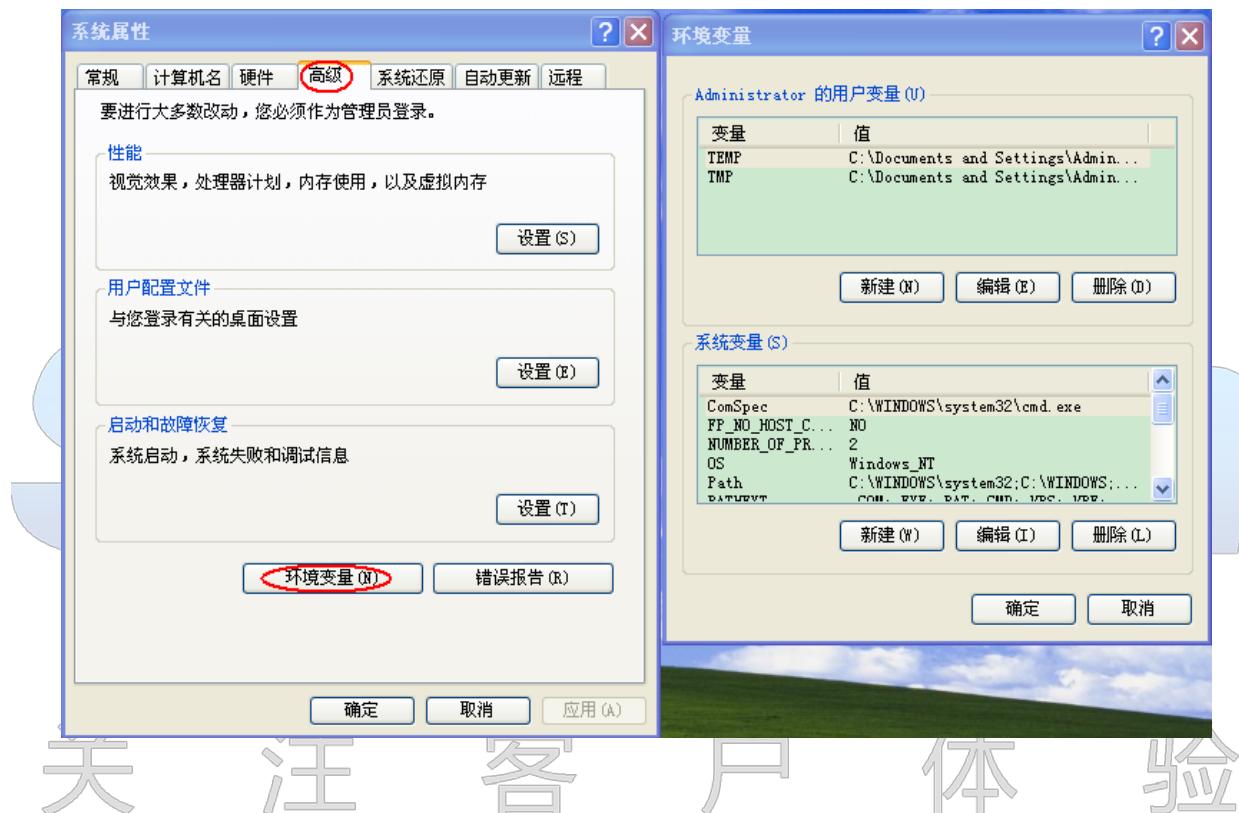
安装 JDK

下载地址: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

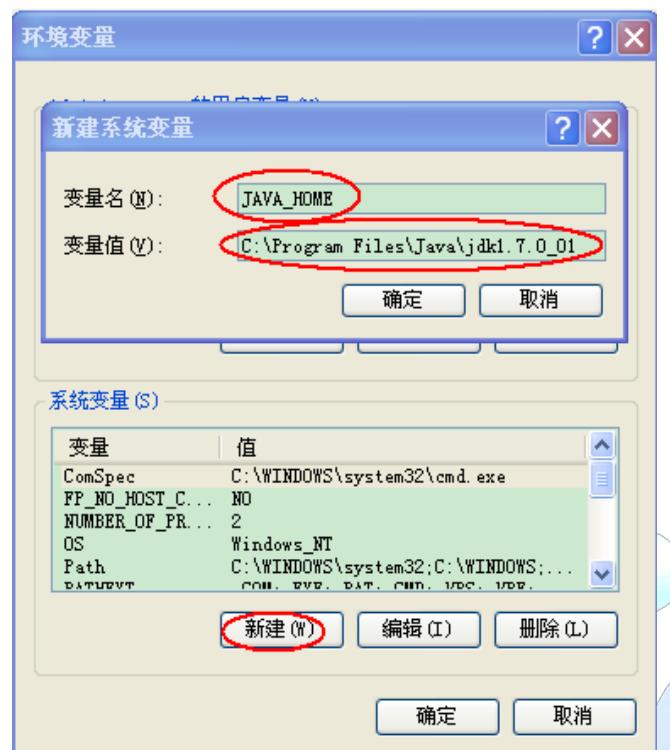
安装过程直接下一步即可, 安装完成会发现 jdk 的安装路径在 C:\Program Files\Java 目录下。

下面开始配置环境变量:

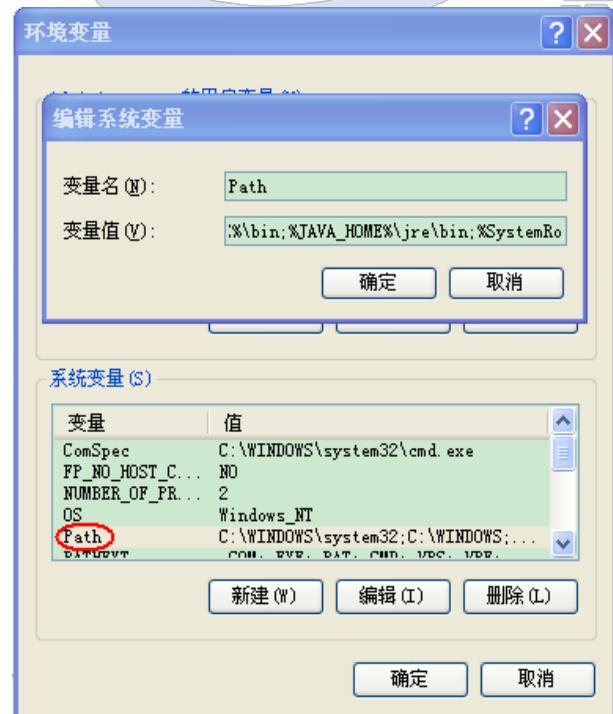
1、右击【我的电脑】---【属性】----【高级】---【环境变量】如图:



2、选择【新建系统变量】--弹出“新建系统变量”对话框，在“变量名”文本框输入“JAVA_HOME”，在“变量值”文本框输入 JDK 的安装路径（也就是 C:\Program Files\Java\java 版本的文件夹路径），单击“确定”按钮，如图：



3、在“系统变量”选项区域中查看 PATH 变量，如果不存在，则新建变量 PATH，否则选中该变量，单击“编辑”按钮，在“变量值”文本框的起始位置添加“%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;%SystemRoot%”，单击确定按钮，如图：



4、在“系统变量”选项区域中查看 CLASSPATH 变量，如果不存在，则新建变量 CLASSPATH，否则选中该变量，单击“编辑”按钮，在“变量值”文本框的起始位置添加“.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;”。如图：



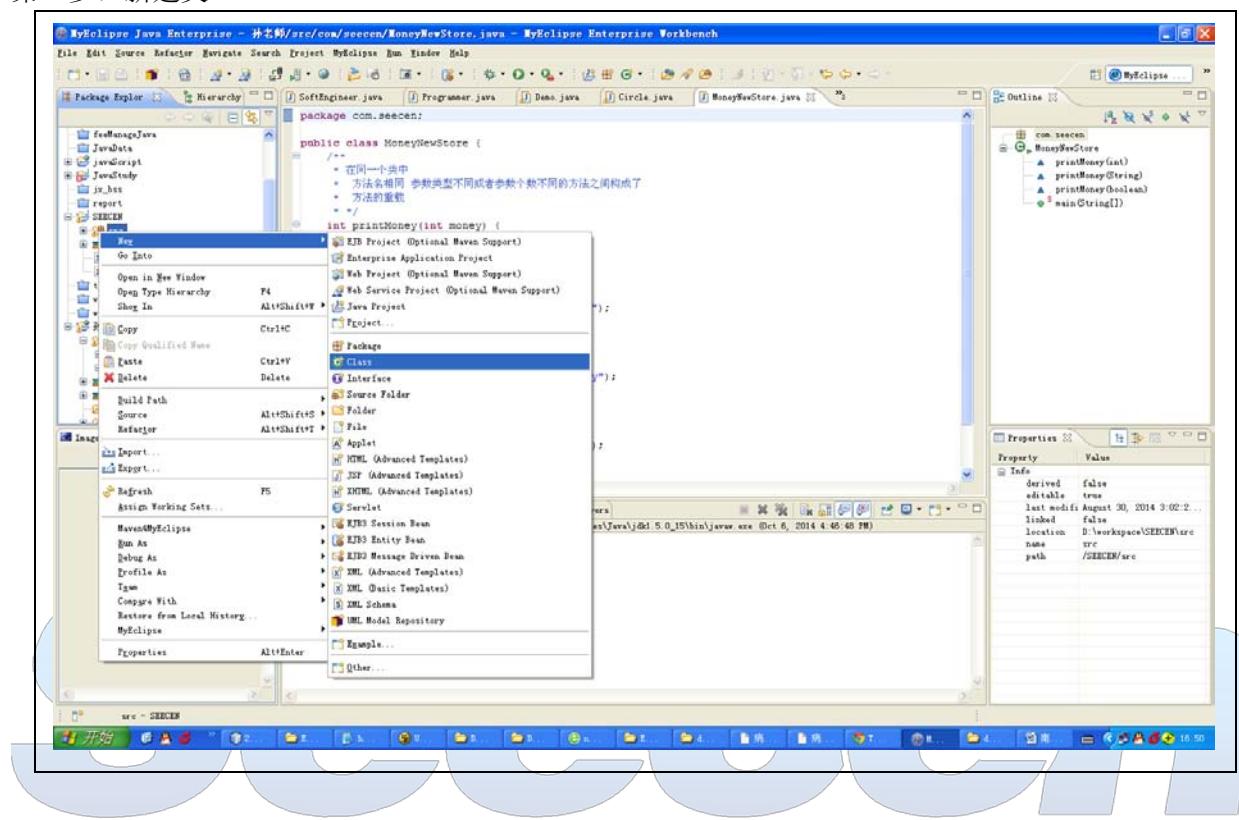
5、现在测试环境变量的配置成功与否。在【开始】---【运行】---输入 CMD 回车进入 DOS 命令行窗口输入“JAVAC”，输出帮助信息即为配置正确。

8.2.3 MyEclipse 简介

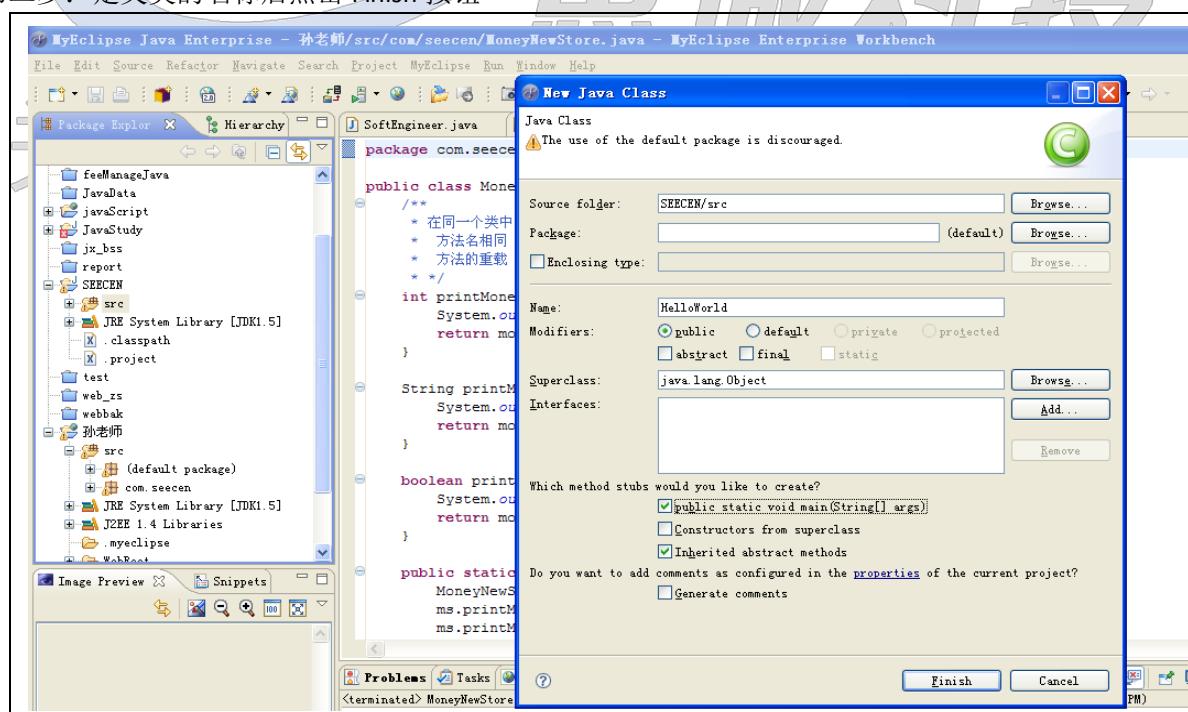
MyEclipse 企业级工作平台 (MyEclipse Enterprise Workbench, 简称 MyEclipse) 是对 Eclipse IDE (即 Integrated Development Environment, 是“集成开发环境”的英文缩写, 指可以辅助开发程序的应用软件。) 的扩展, 利用它我们可以在数据库和 J2EE 的开发、发布, 以及应用程序服务器的整合方面极大的提高工作效率。它是功能丰富的 J2EE 集成开发环境, 包括了完备的编码、调试、测试和发布功能, 完整支持 HTML, Struts, JSF, CSS, Javascript, SQL, Hibernate。

8.2.4 用 Myeclipse 创建一个类的方法

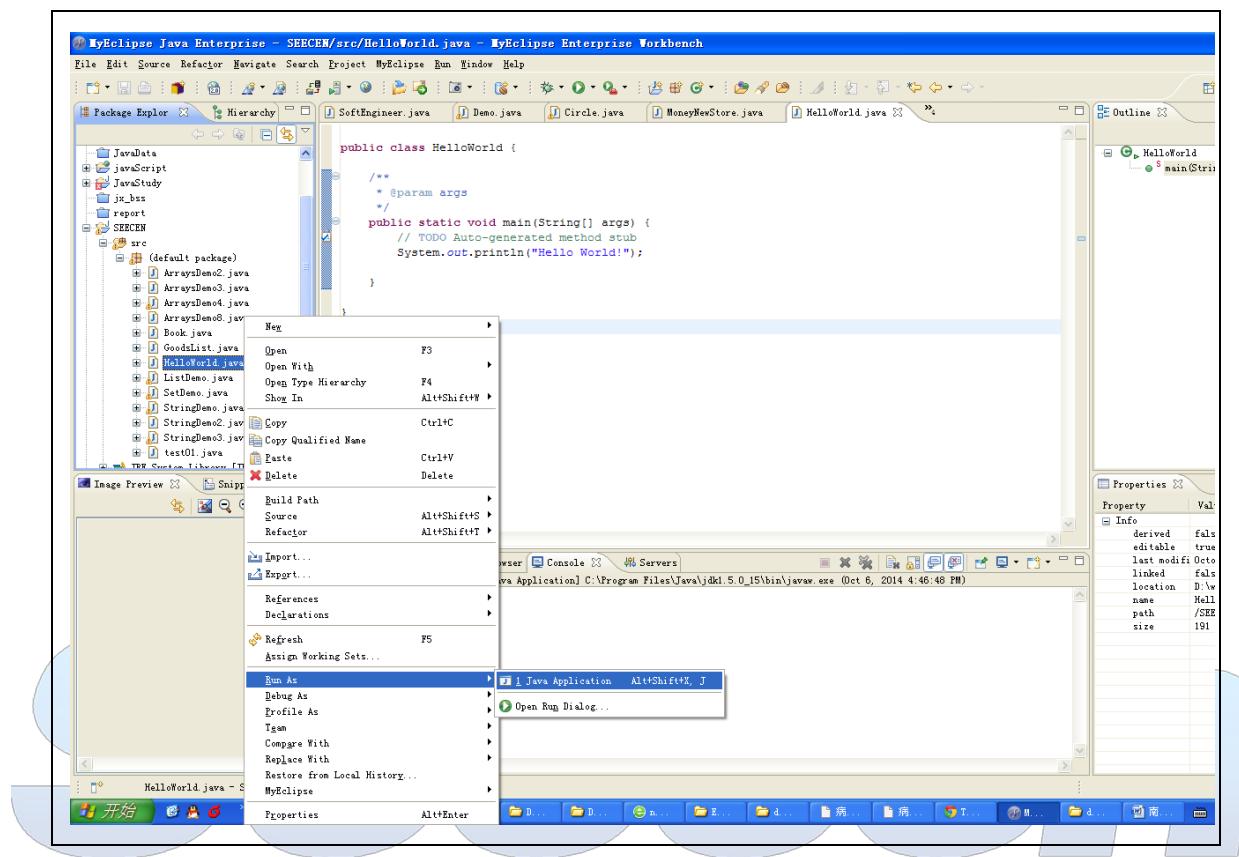
第一步：新建类



第二步：定义类的名称后点击 Finish 按钮



8.2.5 用 Myeclipse 运行一个类的方法



8.2.6 运行第一个 java 程序

The diagram illustrates the structure of a Java program, specifically the `HelloWorld.java` file:

```
public class HelloWorld{
    public static void main(String[ ] args){
        System.out.println("Hello World!!!!");
    }
}
```

Annotations explain the following concepts:

- 关键字**: Points to the `public`, `class`, `HelloWorld`, `main`, `String`, and `System.out.println` keywords.
- 类名与文件名完全一样**: Points to the class name `HelloWorld`.
- main方法四要素必不可少**: Points to the `public static void main(String[] args){}` declaration.
- main方法是Java程序执行的入口点**: Points to the `System.out.println` statement.
- 从控制台输出信息**: Points to the output in the console: "Hello World!!!!".
- {和}一一对应，缺一不可**: Points to the opening brace `{` and closing brace `}`.

代码解析:

- 在 java 中声明一个类的方式有两种，即 `public class` 类名称和 `class` 类名称。
- 使用 “`public class` 类名称” 声明一个类时，要求存储该文件的名字和类名称必须一致，否则程序将无法编译。
- 使用 “`class` 类名称” 声明一个类时，类名称可以与文件名称不一致。
- 在一个 Java 文件中可以有多个 `class` 类的定义，但是只能有一个 `public class` 定义。
- 在定义类名称时，开头的首字母为大写，实际上这属于 Java 的命名规范，只要是类的定义，则类名称中每个单词的首字母必须大写。
- `public static void main(String[] args)` 是程序的主方法，即所有的程序都会以此方法作为起点并运行下来。

`print` 的两种效果：

○ 如何使 `System.out.println("")`; 和 `System.out.print("\n")`; 达到同样的效果?

○ 使用转义符

转义符	说 明
<code>\n</code>	将光标移动到下一行的第一格
<code>\t</code>	将光标移到下一个水平制表位置

```
public class HelloWorld{
    public static void main(String[] args){
        System.out.print("Hello World!!!\n");
    }
}
```

打印输出信息后
将会自动换行

案例：编写一个小程序从控制台打印出您的姓名和年龄

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("我的姓名是: 张三");
        System.out.println("我的年龄是: 23");
    }
}
```

Java 控制台输入 Scanner , 终端输入语句

```
import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个数字");
        int num = input.nextInt();
        System.out.println(num);
    }
}
```

8.2.7 JAVA 注释

在 Java 的编写过程中我们需要对一些程序进行注释，除了自己方便阅读，更为别人更好理解自己的程序，所以我们需要进行一些注释，可以是编程思路或者是程序的作用，总而言之就是方便自己他人更好的阅读。

注：注释内容即不会被编译的内容(对代码本身没有任何影响)，只是解释说明

单行注释：

```
// 控制台输出一行 hello word!
System.out.println("hello world!");
```

多行注释

```
/*
 * main 方法
 * 输出一行 hello word
 */
public static void main(String[] args) {
    System.out.println("hello world!");
}
```

文档注释

```
/**
 * main 方法
 * 输出一行 hello word
 * @author seecen
 * @verstion 1.0
 */
```

```
public static void main(String[] args) {  
    System.out.println("hello world!");  
}
```

注：文档注释允许你在程序中嵌入关于程序的信息。你可以使用 javadoc 工具软件来生成信息，并输出到 HTML 文件中。

8.2.8 JAVA 变量数据类型

什么是变量：

- Java 变量是程序中最基本的存储单元，其要素包括变量名，变量类型和作用域。
- 从本质上讲，变量其实是内存中的一小块区域，使用变量名来访问这块区域，所以，每一个变量使用前必须要先声明（申请），然后进行赋值（填充内容），才能使用。

变量的划分：

- 局部变量：方法（函数）、控制块、构造块体内部声明的变量（包括形参），成为局部变量。
- 成员变量：类内方法外声明的变量成为成员变量。

区别：

- 作用域范围不同
- 实例变量有默认值，局部变量必须显式赋值后才能使用。

变量的声明：

使用变量的步骤：

- 第一步：声明变量，即“根据数据类型在内存申请空间”

数据类型 变量名；

int money;

- 第二步：赋值，即“将数据存储至对应的内存空间”

变量名 = 数值；

money = 1000 ;

第一步和第二步可以合并

数据类型 变量名=数值；

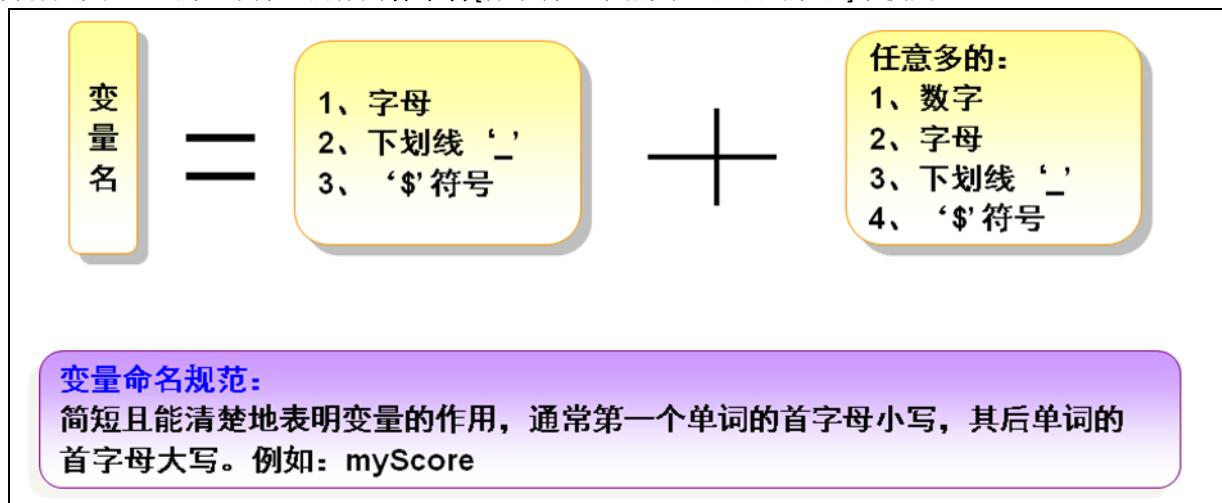
int money = 1000;

- 第三步：使用变量，即“取出数据使用”

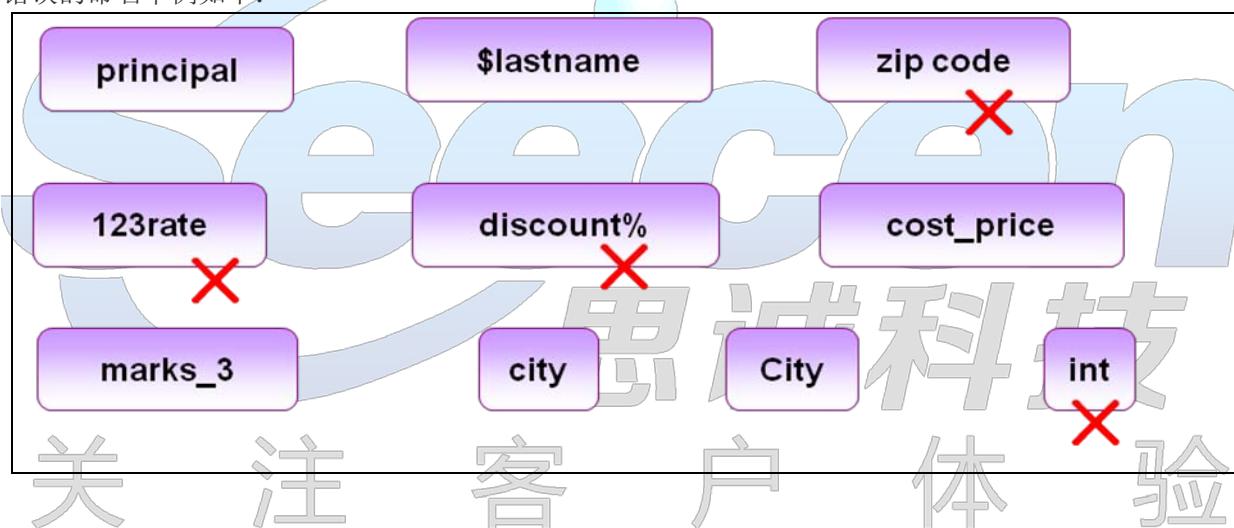
变量命名规则：

Java 中的包、类、方法、参数和变量的名字可由任何顺序的大小写字母、数字、下划线和美元符号组成，

但标示符不能以数字开头，也不能是 Java 中的保留关键字（关键字也成为保留字，是 Java 语言本身使用的，被赋予特定含义的一类标示符。用户只能按照系统的规定来使用它们，不允许对它们进行修改或自行定义，也不允许将它们作为标示符[标示符通常指变量名或函数名]来使用）。



错误的命名举例如下：



8.2.8.1 数据类型：

Java 数据类型包含基本数据类型和引用（复合）数据类型。基本数据类型分为数值型、字符型、和布尔型；引用类型又分为数组、类和接口类型。

1) 整数类型：

Java 把整数类型细分为字节型(byte)、短整型(short)、整型(int)、长整型(long)。

名称	类型标示符	默认值	取值范围	长度
字节型	byte	0	-128~127	1 字节
短整型	short	0	-32768~32767	2 字节
整形	int	0	-2147483648~2147483647	4 字节
长整型	long	0	-9223372036854774808~9223372036854774807	8 字节

2) 浮点型

名称	类型标示符	默认值	取值范围	长度
单精度	float	0.0f	2 的-149 次方~2 的 128 次方-1	4 字节
双精度	double	0.0	2 的-1074 次方~2 的 1024 次方-1	8 字节

3) 字符型

字符类型中每个字符占用两个字节，它使用的是 Unicode 字符集。字符类型可以与 int 类型转换。

4) 布尔型

布尔类型有两种取值，true 和 false，在内存中占 1 字节。

名称	类型标示符	默认值	取值范围	长度
字符型	char	0 或 '\u000000'	0 ~ 65535 \u0000' ~ '\uffff'	2 字节
布尔型	boolean	false	true, false	1 字节

案例 1：main 函数是入口函数 60 是不会打印的

```
public class Demo {  
    public static void main(String[] args) {  
        byte b;  
        b = 12;  
        System.out.println(b);  
        b = 3;  
        System.out.println(b);  
    }  
    {  
        byte b = 60;  
        System.out.println(b);  
    }  
}
```

案例 2：char 型

```
public class Demo {  
    public static void main(String[] arg) {  
        int a = 10;  
        int b = a + 30;  
        char c = 'a';  
        b = a + c; // 109  
        // System.out.println(b);  
        long n = 0x7FFFFFFFFFFFFFl;  
        int num = (int) n;  
        // System.out.println((int)'\t');  
        // System.out.println((int)'\b');  
        // System.out.println((int)'\n');
```

```
//           System.out.println((int)'\r');
//9  8  10 13
System.out.println(" " + '\t' + '\b' + '\n' + '\r' + num);
}
}
```

案例 3: long 型

```
public class Demo {
    public static void main(String[] args) {
        long b = 122;
        long a = 2200000000L;
        long mills = System.currentTimeMillis();
        // long year=mills/1000/60/60/24/365+1970;
        long max = 0xFFFFFFFFFFFFFL;
        long year = max / 1000 / 60 / 60 / 24 / 365 + 1970;
        System.out.println(year);
    }
}
```

案例 4: int 型

```
public class Demo {
    public static void main(String[] args) {
        int a = 10; // 声明与赋值一起使用
        // int a; a=10;
        // int b=2200000000;
    }
}
```

案例 5: float 型

```
public class Demo {
    public static void main(String[] args) {
        float f = 3.141592652578921212f;//f F
        double d = 3.141592652578921212D; //D d带小数位的默认double
        System.out.println(f);
        System.out.println(d);
    }
}
```

案例 6: char 型

```
public class Demo {  
    public static void main(String[] args) {  
        char a = 'a';  
        int n = a;  
        System.out.println(a);  
        System.out.println(n);  
        //随机产生a-z范围内的字母  
        long r = Math.round(Math.random() * 25);  
        char b = (char) (a + r);  
        // System.out.println(b);  
        char s = '1';  
        System.out.println((int) s);  
    }  
}
```

案例 7: boolean 型

```
public class Demo {  
    public static void main(String[] args) {  
        // boolean f=0; error  
        boolean f = false;  
        boolean isMan = true;  
        System.out.println(f);  
    }  
}
```

8.2.8.2 数据类型转换:**1) 自动转换**

转换规则:

- 执行算术运算时, 低类型(短字节)可以转换为高类型(长字节); 例如 int 型转换成 double 型, char 型转换成 int 型等等;
- 转换按数据长度增加的方向进行, 以保证精度不降低。如 int 型和 long 型运算时, 先把 int 转成 long 型后再进行运算。
- 若两种类型的字节数不同, 转换成字节数高的类型。

```
public class Demo {  
    public static void main(String[] args) {  
        double firstAvg = 81.29; //第一次平均分  
        double secondAvg; //第二次平均分  
        int rise = 2;  
        secondAvg = firstAvg + rise;  
        System.out.println("第二次平均分是: " + secondAvg);  
    }  
}
```

2) 强制类型

强制类型转换是通过类型转换运算来实现的。其一般形式为：(类型说明符)(表达式)。其功能是把表达式的运算结果强制转换成类型说明符所表示的类型。

```
public class Demo {  
    public static void main(String[] args) {  
        double firstAvg = 81.29; //第一次平均分  
        int rise = 2;  
        int secondAvg = (int) (firstAvg + rise);  
        System.out.println("第二次平均分是：" + secondAvg);  
    }  
}
```

8.2.9 JAVA 编码规范

类名命名规则：

类的名称应该是一个名词，采用大小写混合的方式。其中每个单词的首字母应大写。如：

```
/**  
 * 用户类  
 * @author seecen  
 * ...  
 */  
public class User{}  
/**  
 * 思诚用户类  
 * @author seecen  
 */  
public class SeecenUser{}
```

每个类定义前必须添加类的注释

方法命名规则：

方法名称应是一个动词或动名结构，采用大小写混合的方式，其中第一个单词的首字母用小写，其后单词的首字母大写。如：

```
public static void sayHelloWorld(){ }
```

变量命名规则：

变量命名一般采用大小写混合的方式，第一个单词的首字母小写，其后单词的首字母大写，变量名一般不要用下划线或美元符开头。变量名应简短且有意义，即能够指出其用途。如：

```
int age; // 年龄  
String userName; // 用户名
```

8.2.10 JAVA 运算符

Java 提供了丰富的运算符环境。Java 有 4 大类运算符：算术运算、位运算、关系运算和逻辑运算。Java 还定义了一些附加的运算符用于处理特殊情况。

8.2.10.1 算术运算符：

序号	运算符	含义
1	+	加法
2	-	减法
3	*	乘法
4	/	除法
5	%	模运算 (取余运算)
6	++	递增运算
7	--	递减运算
8	+=	加法赋值
9	-=	减法赋值

- 1、加法运算符，相当于 $1 + 1 = 2$
- 2、减法运算符，相当于 $2 - 1 = 1$
- 3、乘法运算符，相当于 $2 * 2 = 4$
- 4、除法运算符，相当于 $4 / 2 = 2$

```
public static void main(String[] args) {  
    1.     System.out.println(5 % 2); //整数运算取余  
    2.     System.out.println(2.0 % 1.0); //浮点数运算取余  
}
```

5、取模运算符，其运算结果是整数除法的余数，它还能被用于浮点类型数的取余运算。

运算结果第一行为 1，第二行为 0.0

6、递增运算符，

1)、 $a = x ++$, $x ++$ 不增值。可以看做 $a = x$ $x = x + 1$ 2)、 $a = ++x$, $++x$ 增值。可以看做 $a = x + 1$; $x = x + 1$

7、递减运算符

1)、 $a = x --$, $x --$ 不减值。可以看做 $a = x$ $x = x - 1$
2)、 $a = --x$, $--x$ 减值。可以看做 $a = x - 1$ $x = x - 1$

8、加法赋值符，例如 $x += 1$ ，相当于 $x = x + 1$ 9、减法赋值符，例如 $x -= 1$ ，相当于 $x = x - 1$

注：JAVA 运算符的优先级运算符的优先级决定了多个运算符在一个表达式中运算的顺序，其中最简单的是乘除的优先级大于加减。而一旦表达式比较复杂时，程序员经常会忘记其他优先级规则，所以应该用括号明确规定计算顺序。例：`int a = 100 - 12.5 * 3 + 5 / 2 + 2` 这个表达式的顺序如果不加打括号，任何人都会先计算乘除，然后才计算加减。而只要加上括号后，这个表达式就有了个不同的含义。比如：
`int a = (100 - 12.5) * 3 + 5 / (2 + 2)`

案例 1: `++ --` 运算符

```
public class Demo {
    public static void main(String[] args) {
        //      int a=10;
        //      int b= a++;
        //      int c= ++a;
        //      int d=c++;
        //      int e =++a+d++;
        int a = 10;
        int b = a--;
        int c = --a;
        int d = c--;
        int e = --a + d--;
        System.out.println(a);
        System.out.println(b);
        System.out.println(c);
        System.out.println(d);
        System.out.println(e);
    }
}
```

案例 2: `%`

```
public static void main(String[] args) {
    int a=10;
    int b=10%3; //10/3 后的余数
    //System.out.println(b);
    int s=a>b?a:b;
    System.out.println(s);
}
```

8.2.10.2 位运算符:

前面我们提过，所有的整数类型，除了 `char` 外，都是有符号的。JAVA 使用补码表示二进制数，在补码表示中，最高位为符号位，正数的符号位为 0，负数的符号位为 1

序号	运算符	含义
1	<code><<</code>	左移(移位运算符)
2	<code>>></code>	右移(移位运算符)
3	<code>>>></code>	无符号右移(移位运算符)
4	<code>~</code>	非(取反)(位逻辑运算符)
5	<code>^</code>	异或(位逻辑运算符)
6	<code>&</code>	与(位逻辑运算符)
7	<code> </code>	或(位逻辑运算符)

注：位运算符只存在于整数之间，前面说过 `byte`、`short`、`char` 在运算时都会转为 `int` 类型，所以这 3 种类型也可以参与位运算。浮点数不能进行位运算

上表中 7 种位运算符中，前 3 种是移位运算符，后 4 种是位逻辑运算符

移位运算符有 3 个：`<<`、`>>`、`>>>`

1、<<左移运算符

例：-2 << 2

解：-2 以二进制数表示为：11111111 11111111 11111111 11111110 整形为 32 位。左移运算符表示，将这个二进制数往左边移 2 位，即在末尾补 2 个 0。结果得到二进制数：11111111 11111111 11111111 11111000 去除符号位转成 10 进制数，得 8，其符号位为 1，即为负数，结果为 -8

2、>>右移运算符

例：5 >> 2

解：5 以二进制表示为：00000000 00000000 00000000 00000101 右移运算符表示，讲这个二进制数往右移动 2 位，即在前面补 2 个 0，后面多余的部分移除。结果得到二进制数：00000000 00000000 00000000 00000001 去除符号位转成 10 进制数，得 1，其符号位为 0，即为正数，结果为 1

3、>>>无符号右移

无符号右移动，即忽略符号位，将二进制数转成十进制数时也将符号位的 1 或 0 计算在内

例 1：-2 >>> 5

解：-2 以二进制数表示为：11111111 11111111 11111111 11111110 将这个二进制数往右移 3 位，即得：00000111 11111111 11111111 11111111 忽略符号位，再转为十进制数得到结果：134217727。

例 2：5 >>> 2

解：5 以二进制数表示为：00000000 00000000 00000000 00000101 将这个二进制数往右移 3 位，即得：00000000 00000000 00000000 00000001 忽略符号位，再转为十进制得到结果：1

4 个位逻辑运算符号：^、~、&、|

1、^异或运算符

异或运算先查看两个数的二进制表示值，并执行按位异或。按位异或的计算方法为，当且仅当只有一个数的二进制数的某位为 1 时，结果的该位才为 1，否则结果的该位为 0。

例 1：5 ^ 3

解：5 的二进制表示：00000000 00000000 00000000 00000101 (前面为符号位)

3 的二进制表示：00000000 00000000 00000000 00000000 00000011

结果为：00000000 00000000 00000000 00000 110

结果转成 10 进制数为：6

例 2：-5 ^ 2

解：-5 的二进制表示：11111111 11111111 11111111 11111 101

2 的二进制表示：00000000 00000000 00000000 00000 010

2、~取反运算符

取反运算符，顾名思义，取反运算符就是把某个二进制表示数中的 1 变成 0，0 变成 1。

例：~4

解：4 的二进制数表示为：00000000 00000000 00000000 00000 100 (前面为符号位)

取反为：11111111 11111111 11111111 11111 011 转成十进制数为：-5

3、&逻辑与运算符

逻辑与运算最直观的运算规律为：遇 0 得 0。

即两个二进制数进行逻辑与运算时，在相同位数上只要有 0，则结果的相同位上得 0。

例 1: 3 & 5

解: 3 的二进制表示为: 00000000 00000000 00000000 00000011

5 的二进制表示为: 00000000 00000000 00000000 00000101

逻辑与运算结果为: 00000000 00000000 00000000 00000001

转成十进制数结果为: 1

4、 | 逻辑或运算符

逻辑或运算最直观的运算规律为: 遇 1 得 1

即两个二进制数进行逻辑或运算时, 在相同位数上只要有 1, 则结果的相同位上得 1 例 1: 4 | 3

解: 4 的二进制表示为: 00000000 00000000 00000000 00000100

3 的二进制表示为: 00000000 00000000 00000000 00000011

逻辑或运算结果为: 00000000 00000000 00000000 00000111

转成十进制数结果为: 7

8.2.10.3 关系(逻辑)运算符:

关系(逻辑)运算符有: &&、 || 、 ! 、 == 、 != 、 > 、 < 、 >= 、 <= 包含逻辑运算符的表达式的返回值只可能是 true 或 false

运算符	说明
&&	逻辑与
	逻辑或
!	逻辑非
==	等于
!=	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于

1、 &&逻辑与运算

逻辑与运算符一般用来判断是否同时满足多个条件。先举一个最简单的例子: true && false

解: 上面的例子说明即要满足 true , 又要满足 false 。试想一下, 有没有任何一件事情既能满足真, 又能满足假的? 比如我们去购物, 有没有商品既是真的又是假的呢? 肯定不存在的, 所以上面那个表达式会返回 false 。

再来个例子: (2 > 1) && (2 < 4)

解: 在这个表达式中, 先判断 $2 > 1$, 这个表达式返回的是 true, 因为 2 是大于 1 的。再判断 $2 < 4$, 返回的也是 true。所以这个表达式可以看成 true && true , 所以返回 true。

由上面的例子看来, 可以看出&&运算符的运算规律: 遇假得假。

2、 || 逻辑或运算

逻辑或运算符用来判断多个条件中是否满足其中一个。

也先一个最简单的例子: `true || false`

解: 上面的例子说明可以满足 `true` 也可以满足 `false`。还是按上面的比方来说明, 购物的时候, 商家出售的商品有真有假, 有些商品我们购买的时候可以不在乎真假, 只要能用就可以了。那么就是说, 只要在 `true` 或 `false` 中只要有一个是真就可以了。所以上面的表达式会返回 `true`。

3、! 逻辑非运算

逻辑非运算符用来对已判断出的结果取反, 所以也叫取反运算符。最简单的例子: `!true`

解答: `true` 相当于已判断的结果, 然后再加上个取反运算符, 其结果为 `false`

再来个例子: `!(2 > 4)`

解答: `2 > 4` 是不成立的, 所以这个表达式的返回值为 `false`, 然后再取反, 所以整个表达式的运算结果为: `true`

由上面的例子可以得出: `!` 运算符的运算规律: 遇真得假, 遇假得真

4、== 等于运算

等于运算, 顾名思义, 是判断两个值是否相等的符号。

例: `1 == 2`

解: 相信大家都能看的出来, `1` 肯定是不等于 `2` 的, 所以这个表达式会返回 `false`。

`==` 运算符对于基本数据类型, 是比较值的。而对于引用数据类型, 比较内存地址是否相等的

5、!= 不等于运算

不等于运算, 顾名思义, 是判断两个值是否不等的符号例: `1 != 2`

解: `1` 肯定是不等于 `2` 的, 所以这个表达式会返回 `true`。

`!=` 运算符对于基本数据类型, 也是比较值的。而对于引用数据类型, 比的依然是内存地址

6、其它逻辑运算符都十分的好理解, 这里就不一一讲述了

运算符	说明
<code>></code>	大于
<code><</code>	小于
<code>>=</code>	大于等于
<code><=</code>	小于等于

8.2.11 JAVA 逻辑控制语句

程序三种结构 1、顺序结构 2、选择结构 3、循环结构
选择结构有 `If/else` `switch` 循环结构 `while` `do while` `for` 程序流程跳转语句 `break`、`continue`

在程序设计时, 经常需要使用选择结构在程序中完成逻辑判断和选择功能, 这就需要使用到选择语句。
Java 中的选择语句包括 `if` 语句 `if-else` 语句 `switch` 语句
选择语句用来控制选择结构, 对选择条件进行判断, 并根据判断结果选择要执行的程序语句, 改变程序执行流程。

8.2.11.1 基本 if 选择结构

1、if 选择语句语法格式：

语法格式：

 if(条件表达式)

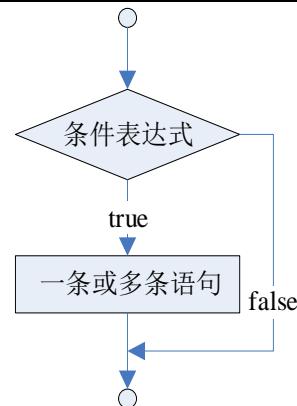
 语句

 或者

 if(条件表达式){

 一条或多条语句

 }



```
1 public class IfTest{  
2     public static void main(String[] args) {  
3         int week=5;  
4         if(week>=5){  
5             System.out.println("终于休息了！");  
6         }  
7     }  
8 }  
9 }  
10 }
```



8.2.11.2 基本 if-else 选择结构

语法格式：

if(条件表达式)

语句 1

else

语句 2

或者

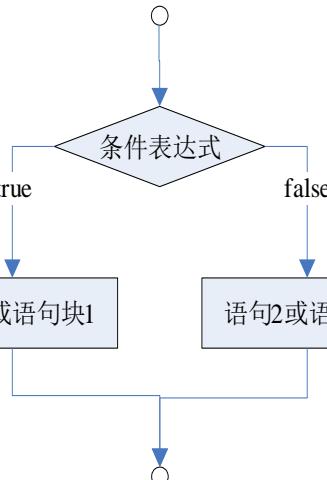
if(条件表达式){

语句块 1

}else{

语句块 2

}



```
1 /**
2  * 比较三个整数中的最大值
3 */
4 public class IfElseTest{
5     public static void main(String[] args){
6         int a=23,b=36,max; //声明三个整型变量
7         if(a>b){ //如果a大于b
8             max = a; //将较大值的a赋给变量max
9         }else{ //否则，如果a不大于b
10            max = b; //将b的值赋给变量max
11        }
12        System.out.println("最大值是：" + max); //输出变量max的值，即输出最大值
13    }
14 }
```

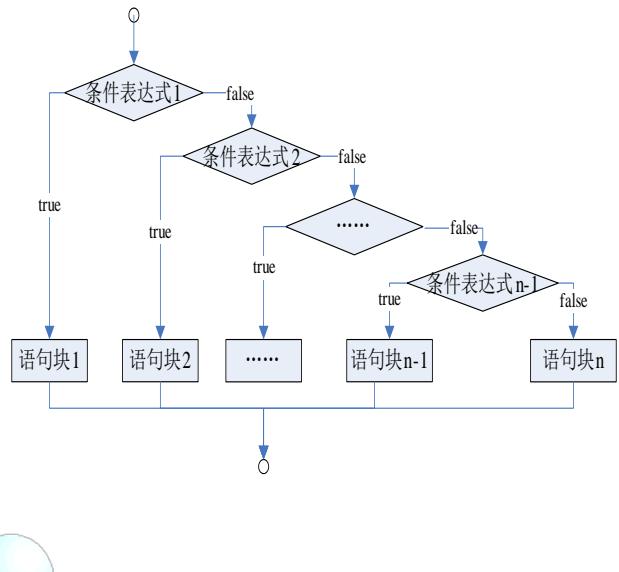
案例：

```
public class Demo {
    public static void main(String[] args) {
        boolean isMan = true;
        int age = 18;
        if (isMan && age > 20) {
            System.out.println("是个男人！");
        } else {
            System.out.println("不是个男人！");
        }
    }
}
```

8.2.11.3 多重 if 选择结构

语法格式：

```
if(条件表达式1){ //如果条件表达式1成立（结果为true）
    语句块1 //就执行语句块1中的代码
} else if(条件表达式2){ //否则，如果条件表达式2成立
    语句块2 //就执行语句块2中的代码
}
...
else if(条件表达式n-1){ //对其他条件进行判断
    语句块n-1 //如果条件表达式n-1成立
    语句块n-1 //就执行语句块n-1中的代码
} else{ //如果以上所有的条件都不成立
    语句块n //就执行语句块n
}
```



```
1 public class IfElseIfTest {
2     public static void main(String[] args) {
3         int score = 85; //定义代表学生成绩的变量score，并赋初值
4         if(score >= 90){ //如果成绩大于等于90分
5             System.out.println("您的成绩优秀！");
6             //执行这个语句块，并结束本if-else-if语句
7         } else if(score >= 80){ //满足这个条件表达式
8             System.out.println("您的成绩良好");
9             //执行这个语句块，并结束本if-else-if语句
10        } else if(score >= 70){ //如果成绩大于等于70分
11            System.out.println("您的成绩中等");
12            //执行这个语句块，并结束本if-else-if语句
13        } else if(score >= 60){ //如果成绩大于等于60分
14            System.out.println("您的成绩及格");
15            //执行这个语句块，并结束本if-else-if语句
16        } else{ //如果以上条件都不成立，即成绩小于60分
17            System.out.println("您的成绩不及格");
18        } //结束if语句是直接转到这里，执行if后面的语句
19    }
20 }
21 }
```

案例：

```
public class Demo {
    public static void main(String[] args) {
        int score = 75; //考试成绩
        if (score >= 90) {
            System.out.println("优秀");
        } else if (score >= 80) {
            System.out.println("良好");
        } else if (score >= 60) {
```

```
        System.out.println("中等");
    } else {
        System.out.println("差");
    }
}
```

案例：

输入一个存款年限与存款金额根据这个年限利率返回到期后能领取的总金额
1 利率 2.75% 2 利率 3.25%
3 利率 4.75% 5 利率 5.15%

```
import java.util.Scanner;
public class Demo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个存款年限");
        int n = input.nextInt();
        System.out.println("请输入一个存款金额");
        double mony = input.nextDouble();
        double rant = 0;
        if (n >= 1) {
            rant = 0.0275;
        }
        if (n >= 2) {
            rant = 0.0325;
        }
        if (n >= 3) {
            rant = 0.0475;
        }
        if (n >= 5) {
            rant = 0.0515;
        }
        double Sum = mony + (mony * rant) * n;
        System.out.println(Sum);
    }
}
```

关

注

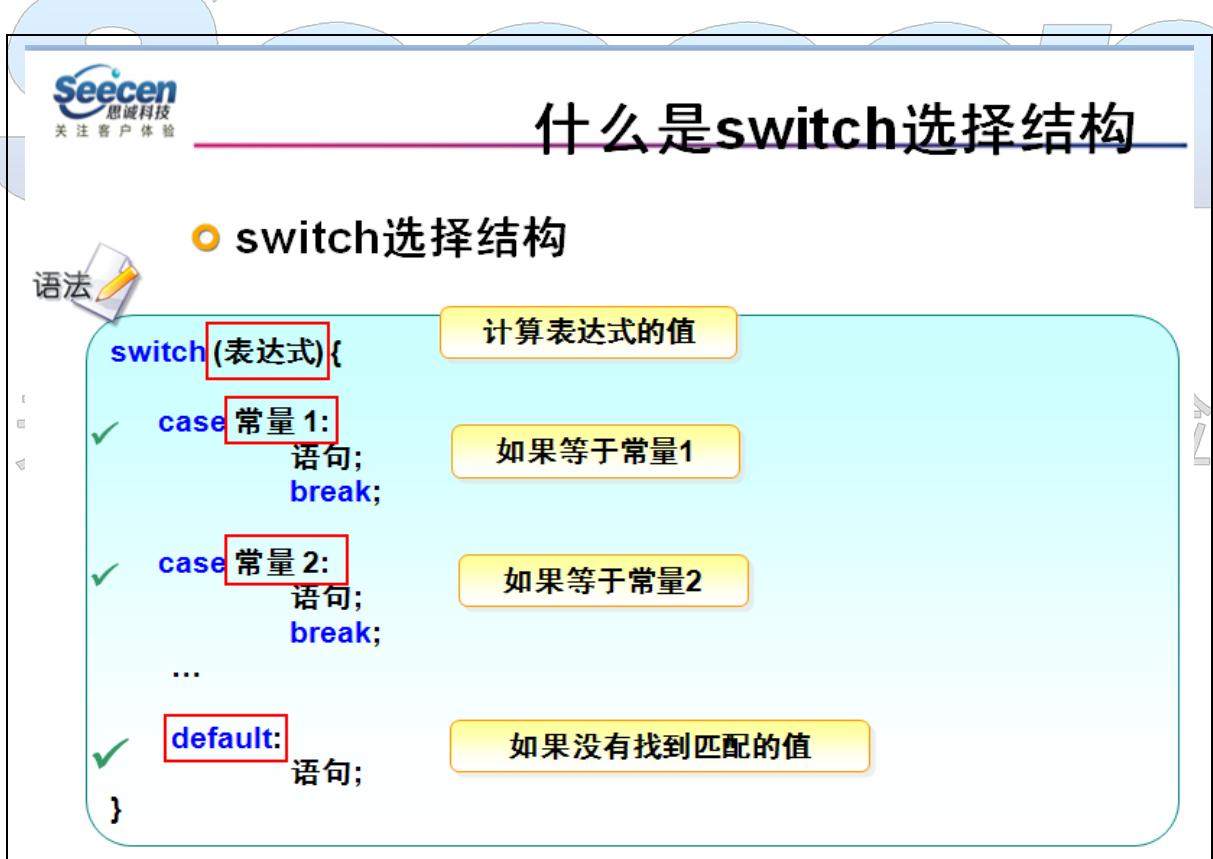
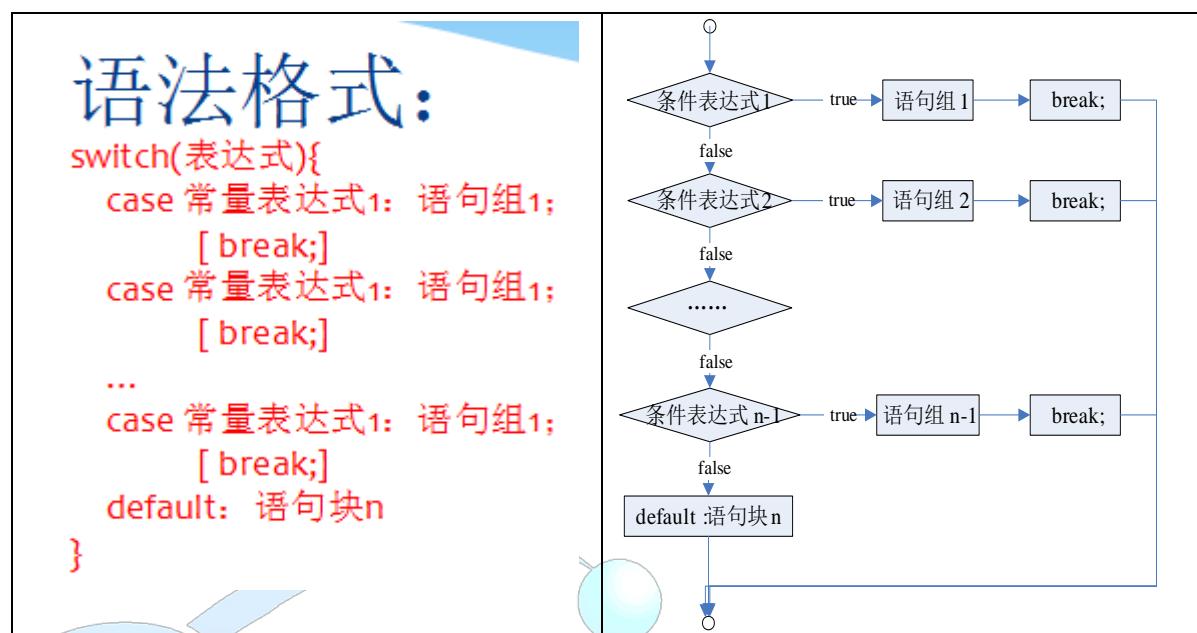
客

户

体

验

8.2.11.4 switch 选择结构



```
1  /**
2   * 根据一个学生的成绩等级判断学生的成绩范围,
3   * 假如学生的成绩等级是A、B、C、D、E
4  */
5  public class SwitchTest{
6
7      public static void main(String[] args){
8          char grade = 'A';
9          switch(grade){
10              case 'A':
11                  System.out.println("你的成绩范围是90-100分！");
12                  break;
13              case 'B':
14                  System.out.println("你的成绩范围是80-89分！");
15                  break;
16              case 'C':
17                  System.out.println("你的成绩范围是70-79分！");
18                  break;
19              case 'D':
20                  System.out.println("你的成绩范围是60-69分！");
21                  break;
22              case 'E':
23                  System.out.println("你的成绩范围是0-59分！");
24                  break;
25              default:
26                  System.out.println("你输入的成绩等级不正确, 请重新输入！");
27          }
28      }
29 }
```

常见错误：

常见错误4-1

代码改错

```
int mingCi = 1;
switch(mingCi){
    case 1:
        System.out.println("参加麻省理工大学组织的1个月夏令营");
    case 2:
        System.out.println("奖励惠普笔记本电脑一部");
    case 3:
        System.out.println("奖励移动硬盘一个");
    default:
        System.out.println("没有任何奖励");
}
```

输出结果是什么？

如果需要每个case执行完后跳出，
在每个case后不要忘记写break;

企业咨询 院校合作 大学生实训 www.seecen.com

常见错误4-2

代码
改错

```
int mingCi = 1;  
switch (mingCi){  
    case 1:  
        System.out.println("参加麻省理工大学组织的1个月夏令营");  
    case 2:  
        System.out.println("奖励惠普笔记本电脑一部");  
    case 2:  
        System.out.println("奖励移动硬盘一个");  
    default:  
        System.out.println("没有任何奖励");  
}
```

case后面的常量必须各不相同

代码错误

演示示例5: switch选择结构常见错误

代码
改错

```
String day = "星期一";  
switch(day){  
    case "星期一":  
        System.out.println("星期一: 青菜 ");  
        break;  
    case "星期二":  
        System.out.println("星期二: 鱼 ");  
        break;  
    ....  
    default:
```

switch后面小括号中表达式的值必须是整型或字符型

代码错误

案例：

```
import java.util.Scanner;
public class Demo {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("请对本次服务做出1-9数字的评价！");
        int num = input.nextInt();
        char c = 'a';
        switch (num) {
            case 0:
                System.out.println("非常满意");
                break;
            case 1:
                System.out.println("满意");
                break;
            case 2:
                System.out.println("一般");
                break;
            case 3:
                System.out.println("3分");
                break;
            case 4:
                System.out.println("4分");
                break;
            case 5:
                System.out.println("5分");
                break;
            case 6:
                System.out.println("6分");
                break;
            case 7:
                System.out.println("7分");
                break;
            case 8:
                System.out.println("8分");
                break;
            case 9:
                System.out.println("9分");
                break;
            default:
                System.out.println("有木有搞错！");
        }
    }
}
```



循环结构

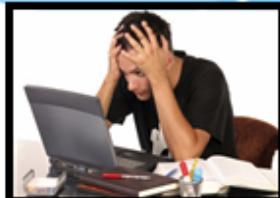
生活中的循环



打印50份试卷



10000米赛跑

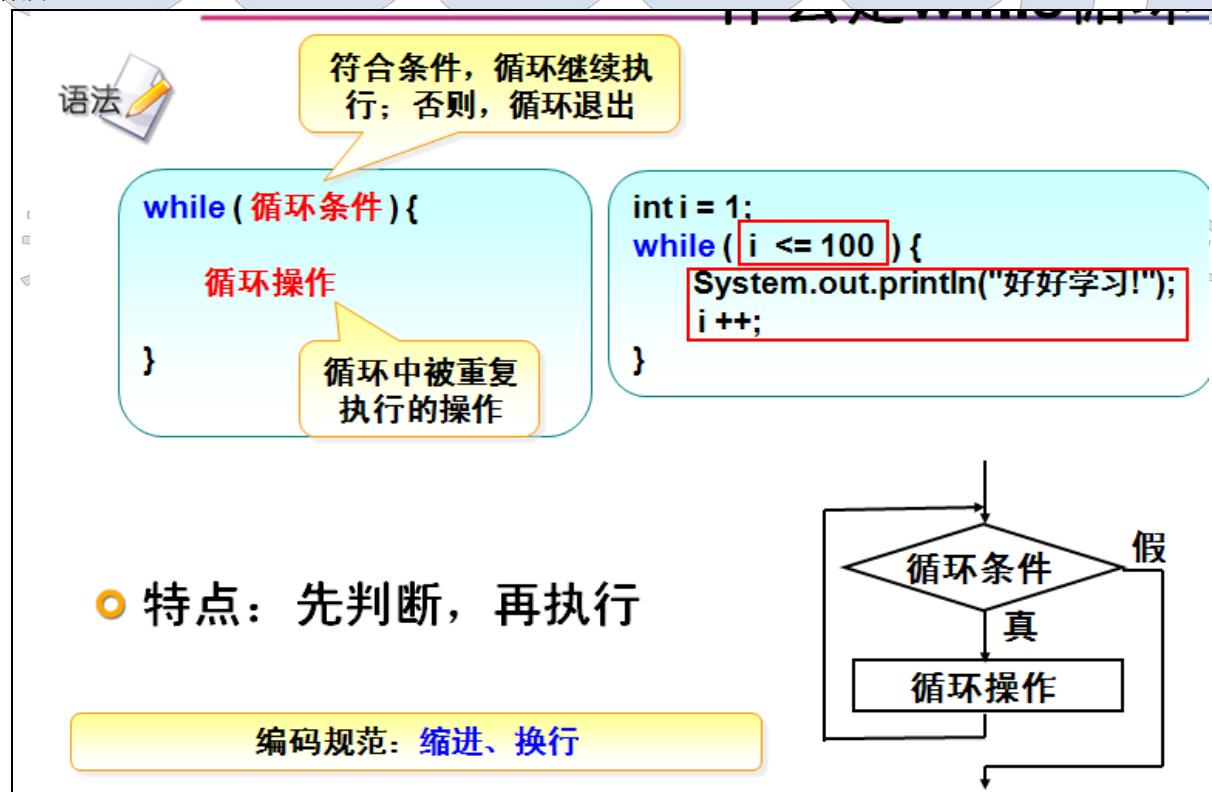


锲而不舍地学习



旋转的车轮

循环结构的特点





* 如何用程序描述下面这个故事呢?

循环操作

为了帮助张浩尽快提高成绩，老师给他安排了每天的学习任务。
其中上午阅读教材，学习理论部分，下午上机编程，掌握代码部分。
老师每天检查学习成果。如果不格，则继续进行



循环条件

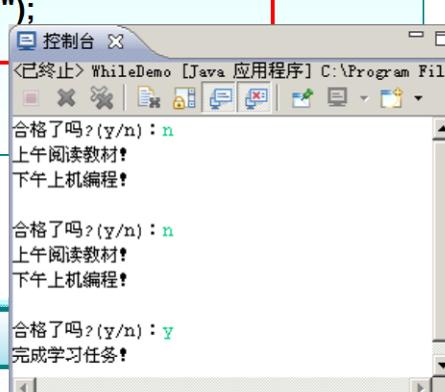
- * 使用while循环的步骤
 - * 1、分析循环条件和循环操作
 - * 2、套用while语法写出代码
 - * 3、检查循环是否能够退出

比较两个String类型的值是否相等

```
System.out.print("合格了吗?(y/n): ");
String answer = input.next();
while !"y".equals(answer)) {
    System.out.println("上午阅读教材！");
    System.out.println("下午上机编程！\n");
    System.out.print("合格了吗?(y/n): ");
    answer = input.next();
}
System.out.println("完成学习任务！");
```

循环条件 循环操作 避免死循环

演示示例：使用while循环结构



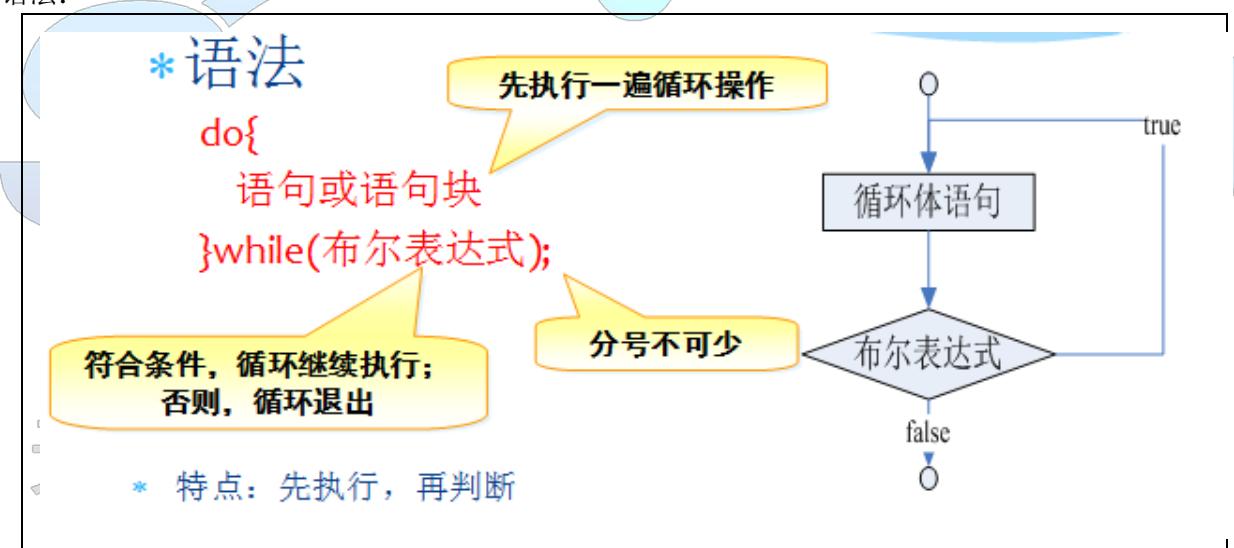
案例：

```
import java.util.Scanner;
public class Demo {
    public static void main(String[] args) {
        String answer; // 标识是否合格
```

```
Scanner input = new Scanner(System.in);
System.out.print("合格了吗?(y/n): ");
answer = input.next();
while (!"y".equals(answer)) {
    System.out.println("上午阅读教材！");
    System.out.println("下午上机编程!\n");
    System.out.print("合格了吗?(y/n): ");
    answer = input.next();
}
System.out.println("完成学习任务!");
```

8.2.11.6 do-while 循环

语法:



* 如何用程序讲述下面的故事?

经过几天的学习，老师给张浩一道测试题，
让他先上机编写程序完成，
然后老师检查是否合格。如果不合格，则继续编写。……

while (循环条件){
 循环操作
}

while循环先判断，再执行
不适合描述此故事

do {
 循环操作
}while (循环条件);



先执行一遍循环操作

```
do{  
    System.out.println("上机编写程序！");  
    System.out.print("合格了吗?(y/n)");  
    answer = input.next();  
    System.out.println("\n");  
  
}while(!"y".equals(answer));  
  
System.out.println("恭喜你通过了测试！");
```

循环条件



演示：使用do-while循环结构

```
-----1-----2-----3-----4-----  
1  /**  
2   *循环输出10句“Java语言”  
3   **/  
4  public class DoWhileTest{  
5      public static void main(String[] args){  
6          int i=0;  
7          do{  
8              System.out.println("Java语言");  
9              i++;  
10         }while(i<10);  
11     }  
12 }  
13
```

案例1：
从 1 加到 99 的和

```
public class Demo {  
    public static void main(String[] args) {  
        int sum = 0;  
        int i = 1;  
        do {  
            sum += i;  
            i++;  
        } while (i < 100);  
        System.out.println(sum);  
    }  
}
```

案例 2：用户修改密码

```
package com.seecen.classandobject;
import java.util.Scanner;
public class Register {
    public boolean verify(String name, String pwd1, String pwd2) {
        boolean flag=false;
        if(name.length()<3 || pwd1.length()<6){
            System.out.println("用户名长度不能小于3，密码长度不能小于6！");
        }else if(!pwd1.equals(pwd2)){
            System.out.println("两次输入的密码不相同！");
        }else{
            System.out.println("注册成功！请牢记用户名和密码。");
            flag=true;
        }
        return flag;
    }
    public static void main(String[] args) {
        Register r=new Register();
        Scanner input = new Scanner(System.in);
        String uname,p1,p2;
        boolean resp=false;
        System.out.println("****欢迎进入注册系统****\n");
        do{
            System.out.print("请输入用户名: ");
            uname=input.next();
            System.out.print("请输入密码: ");
            p1=input.next();
            System.out.print("请再次输入密码: ");
            p2=input.next();
            resp=r.verify(uname, p1, p2);
        }while(!resp);
    }
}
```

关注客户体验

案例 3：用户注册

```
package com.seecen.classandobject;
import java.util.Scanner;
public class Register2 {
    public String verify(String id, String cell, String phone) {
        String flag="注册成功！";
        String[] splitphone=new String[3];
        splitphone=phone.split("-",2);
        if(id.length()!=16 && id.length()!=18){
            flag="身份证号必须是16位或18位！";
        }
    }
}
```

```
        }else if(cell.length()!=11){
            flag="手机号码必须是11位! ";
        }else if(splitphone[0].length()!=4 && splitphone[0].length()!=7){
            flag="座机号码区号必须为4位，电话号码必须是7位! ";
        }
        return flag;
    }

    public static void main(String[] args) {
        Register2 r=new Register2();
        Scanner input = new Scanner(System.in);
        String ID,p1,p2;
        String resp;

        System.out.println("****欢迎进入注册系统*** \n");
        do{
            System.out.print("请输入身份证: ");
            ID=input.next();
            System.out.print("请输入手机号: ");
            p1=input.next();
            System.out.print("请输入座机号: ");
            p2=input.next();
            resp=r.verify(ID, p1, p2);
            System.out.println(resp);
        }while(!resp.equals("注册成功! "));
    }
}
```

do-while 和 while-do 的异同

● while循环和do-while循环的区别

- 语法不同

while (循环条件) {

循环操作

}

先判断，再执行

do {

先执行，再判断

循环操作

} while(循环条件);

- 执行次序不同

- 初始情况不满足循环条件时

■ while循环一次都不会执行

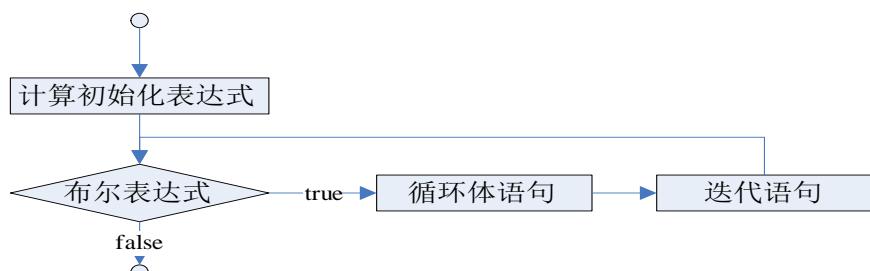
■ do-while循环不管任何情况都**至少执行一次**

8.2.11.7 for 循环

语法

for(初始化表达式; 条件表达式; 迭代语句)

{ 循环体语句 }



for循环的语法和执行顺序



语法
1 `for(参数初始化 ; 条件判断 ; 更新循环变量){`
2 `循环操作 ; 循环体被执行`
3
4

```
for( int i = 0; i < 100; i++ ){
    System.out.println("好好学习！");
}
```

关 注 客 户 体 验

* 回顾问题：输出100次“好好学习！”
使用while循环结构

```
int i=0;
while(i<100){
    System.out.println("好好学习！");
    i++;
}
```

特点：循环次数固定

使用for循环结构

```
for(int i=0;i<100;i++){
    System.out.println("好好学习！");
}
```

for比while更简洁

案例：

```
public class Demo {  
    public static void main(String[] args) {  
        int daikuan = 180000;  
        double rate = 0.054;  
        double mony = daikuan / 120.0;  
        for (int i = 1; i < 11; i++) {  
            System.out.println("第" + i + "年的还款明细");  
            for (int j = 1; j <= 12; j++) {  
                int n = (i - 1) * 12 + j - 1;  
                double returnMoney = mony + (daikuan - mony * n) * rate / 12;  
                System.out.print(returnMoney + "\t");  
            }  
            System.out.println("");  
        }  
    }  
}
```

常见的错误案例

for循环常见问题4-1

代码修改

编译错误：变量i没有初始化

```
int i=0;  
for(;i<10;i++){  
    System.out.println("这是 "+i);  
}
```

语法

表达式1省略，循环变量的初始值在for语句之前由赋值语句取得

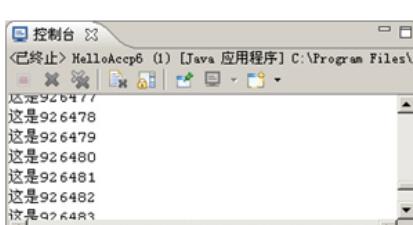
**for(<初始化循环变量>;<循环条件>;<修改循环变量的值>){
<循环体语句>;
}**

不能省略

for循环常见问题4-2

 编译正确，但是缺少
循环条件，造成死循环

```
for(int i=0;;i++){  
    System.out.println("这是 "+i);  
}
```

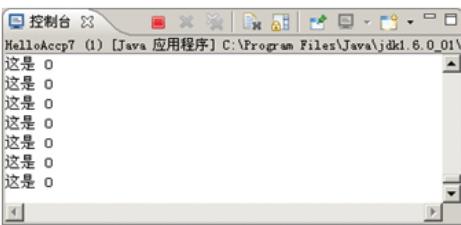


for循环常见问题4-3

 编译通过，但是循环变量的
值无变化，造成死循环

```
for(int i=0;i<10;){  
    System.out.println("这是 "+i);  
    i++;  
}
```

省略表达式3，在循环体内应设法改
变循环变量的值以结束循环



The diagram shows a Seecen logo at the top left. To its right, the text "for循环常见问题4-4" is displayed. Below this, a yellow icon labeled "代码改错" (Code Fix) is shown. A purple callout box contains the text: "表达式全省略, 无条件判断, 循环变量无改变, 应在循环体内设法结束循环; 否则会造成死循环" (The expression is omitted, there is no conditional judgment, the loop variable does not change, and it should be set to end the loop within the loop body; otherwise, it will cause a deadlock). Below the callout is a code snippet in a light blue box:

```
for(;;){  
    System.out.println("这是测试");  
}
```

A yellow speech bubble labeled "死循环" (Deadlock) points to a screenshot of a Java console window titled "控制台". The window shows the output of the above code: "这是测试" repeated multiple times. An orange arrow points from the "死循环" bubble to the scroll bar of the console window.

8.2.11.8 for each 循环

For-Each 循环的缺点：丢掉了索引信息。

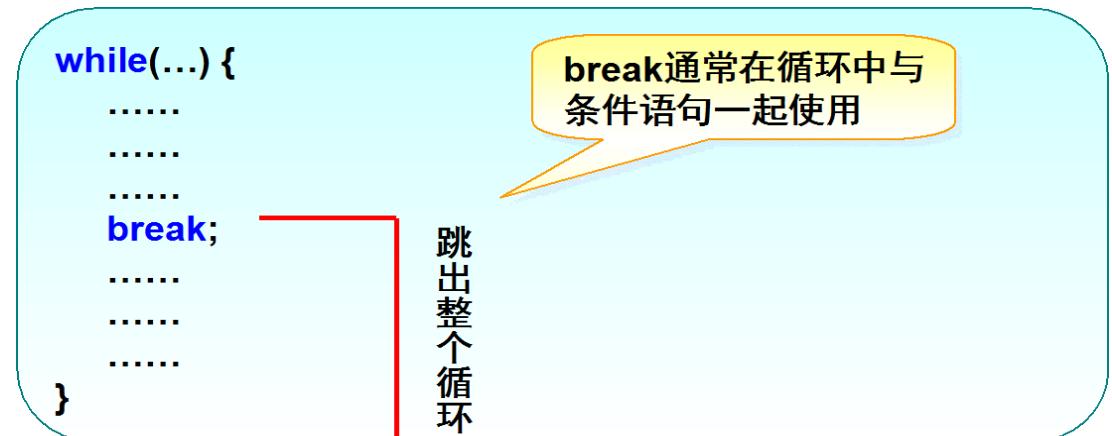
当遍历集合或数组时，如果需要访问集合或数组的下标，那么最好使用旧式的方式来实现循环或遍历，而不要使用增强的 for 循环，因为它丢失了下标信息。

案例：

```
package com.seecen;  
  
import java.util.ArrayList;  
import java.util.Iterator;  
import java.util.List;  
  
public class Demo {  
    public static void main(String[] args) {  
        int[] arr = { 1, 2, 3, 4, 5 };  
        // 新式写法, 增强的for循环  
        for (int element : arr) {  
            System.out.println(element);  
        }  
    }  
}
```

8.2.11.9 break 控制语句

break: 改变程序控制流用于 do-while、while、for 中时, 可跳出循环而执行循环后面的语句



```
1 public class BreakTest{  
2     public static void main(String[] args){  
3         for(int i=0;i<10;i++){  
4             if(i>5){  
5                 break;  
6             }  
7             System.out.println("Java语言");  
8         }  
9     }  
10 }  
11 }
```



案例:

```
import java.util.Scanner;  
  
public class Demo {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        float sum = 0;  
        for (int i = 0; i < 5; i++) { //循环5次录入5门课成绩  
            System.out.print("请输入第" + (i + 1) + "门课的成绩: ");  
            float score = input.nextInt();  
            if (score < 0) { //输入负数  
                break;  
            }  
            sum = sum + score; //累加求和  
        }  
    }  
}
```

8.2.11.10 continue 控制语句

continue: 改变程序控制流用于 do-while、while、for 中时, 可跳出本次循环, 执行下一次循环



```
for (int i = 0; i < total; i++) {  
    System.out.print("请输入第" + (i + 1) + "位学生的成绩: ");  
    score = input.nextInt();  
    if (score < 80) {  
        continue;  
    }  
    num++;  
}  
System.out.println("80分以上的学生成绩是:");  
double rate = (double) num / total * 100;  
System.out.println("80分以上的学生成绩所占的比例为: " + rate + "%");
```

对录入的分数进行判断,
如果小于80, 跳出本次循
环, 执行下一次循环

演示示例4：使用continue语句

案例：

```
import java.util.Scanner;  
public class Demo {  
    public static void main(String[] args) {  
        Scanner input = new Scanner(System.in);  
        int total = 3;  
        float score = 0;  
        int num = 0;  
        for (int i = 0; i < total; i++) {  
            System.out.print("请输入第" + (i + 1) + "位学生的成绩: ");  
            score = input.nextInt();  
            if (score < 80) {  
                continue;  
            }  
            num++;  
        }  
        System.out.println("80分以上的学生成绩是: " + num);  
        double rate = (double) num / total * 100;  
        System.out.println("80分以上的学生成绩所占的比例为: " + rate + "%");  
    }  
}
```

8.2.11.11 break 和 continue 的对比

Seecen
思诚科技
关注客户体验

对比break和continue

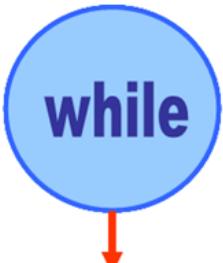
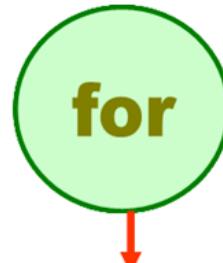
- 使用场合
 - break可用于switch结构和循环结构中
 - continue只能用于循环结构中
- 作用（循环结构中）
 - break语句终止某个循环，程序跳转到循环块外的下一条语句。
 - continue跳出本次循环，进入下一次循环

8.2.11.12 循环结构的总结

Seecen
思诚科技
关注客户体验

循环结构总结2-1

提问 到目前为止所学的循环结构有哪些？

-  while
-  do-while
-  for

需要多次重复执行一个或多个任务的问题考虑使用循环来解决

无论哪一种循环结构，都有4个必不可少的部分：初始部分、循环条件、循环体、迭代部分

**Seecen 思诚科技
关注客户体验**

循环结构总结2-2

- 区别1：语法

while 循环:

```
while(<条件>){  
    //循环体  
}
```

do-while 循环:

```
do{  
    //循环体  
} while(<条件>);
```

for 循环:

```
for(初始化;条件;迭代){  
    //循环体  
}
```

- 区别2：执行顺序
 - **while 循环：**先判断，再执行
 - **do-while 循环：**先执行，再判断
 - **for 循环：**先判断，再执行
- 区别3：适用情况
 - 循环次数确定的情况，通常选用**for 循环**
 - 循环次数不确定的情况，通常选用**while 和 do-while 循环**

8.2.11.13 本节作业

- 1、对录入的信息进行有效性验证
- 2、判断整数 n 是否为素数
- 3、三角形的判断
- 4、操作符运算（输入操作符和数字后得出结果）
- 5、月供计算
- 6、2006 年培养学员 8 万人，每年增长 25%，请问按此增长速度，到哪一年培训学员人数将达到 20 万人？

提示：

1、循环条件和循环操作分别是什么?
2、**int year = 2006;**
double students = 80000;
while ...
3、**2007年培训学员数量 = 80000 * (1 + 0.25)**

- 7、打印出 1000 以内的“水仙花数”。所谓“水仙花数”是指一个三位数，其各位数字立方和等于该数本身。例如：153 是一个“水仙花数”，因为 $1^3 + 5^3 + 3^3 = 153$ 。
- 8、实现九九乘法表的打印功能，打印结果如下

```
G:\demo>java jiujiutest
1*1=1
1*2=2    2*2=4
1*3=3    2*3=6    3*3=9
1*4=4    2*4=8    3*4=12   4*4=16
1*5=5    2*5=10   3*5=15   4*5=20   5*5=25
1*6=6    2*6=12   3*6=18   4*6=24   5*6=30   6*6=36
1*7=7    2*7=14   3*7=21   4*7=28   5*7=35   6*7=42   7*7=49
1*8=8    2*8=16   3*8=24   4*8=32   5*8=40   6*8=48   7*8=56   8*8=64
1*9=9    2*9=18   3*9=27   4*9=36   5*9=45   6*9=54   7*9=63   8*9=72   9*9=81

G:\demo>
```

9、编写程序求 $1!+2!+\dots+30!$ 的结果

8.2.12 类和对象

OOP 思想 (Object Oriented Programming 面向对象程序)

对象：万物皆对象；

类：是一组具有共同特性的所有对象成员的集合，是一个抽象描述。

类的定义可以看作是创建软件对象的模板。如同“月饼模子”所做的事情一样。

在内存中创建一个数据结构已容纳新创建的对象的属性。

将对象和一定的行为集建立关联。

例如：一个学生类可以用来描述学生登记系统中所有注册的学生对象

类的命名规范：

类的名字必须由大写字母开头而单词中的其他字母均为小写；如果类名称由多个单词组成，则每个单词的首字母均应为大写例如 TestPage；如果类名称中包含单词缩写，则这个所写词的每个字母均应大写，如：XMLExample,还有一点命名技巧就是由于类是设计用来代表对象的，所以在命名类时应尽量选择名词

类是客观存在的，抽象的，概念的东西。对象是具体的，实际的，代表一个事物。

例如：车是一个类，汽车，自行车就是他的对象。类是对象的模版，对象是类的一个个体。

8.2.12.1 本节目标

掌握类和对象的特征

理解封装

会创建和使用对象

8.2.12.2 类的定义

语法:

```
public class 类名 {  
    // 定义属性部分  
    属性1的类型属性1;  
    属性2的类型属性2;  
    ...  
    属性n的类型属性n;  
    // 定义方法部分  
    方法1;  
    方法2;  
    ...  
    方法m;  
}
```

案例：创建类

```
package com.seecen;  
  
public class Demo {  
    // 类的成员变量  
    int x, y, z;  
  
    // 类的成员方法  
    public class D {  
        double r;  
  
        double getircles() {  
            return 3.14 * r * r;  
        }  
    }  
}
```



8.2.12.3 对象的定义

- 1、对象的声明：类名对象名；
- 2、对象的创建对象名 = new 类名(); 或类名对象名 = new 类名(); new 作用：分配内存空间。

```
public class Test {  
    public static void main(String[] args) {  
        Test test = new Test();  
    }  
}
```

8.2.12.4 Static

静态变量:

有时候我们希望无论是否产生了对象或无论产生了多少对象的情况下，某些特定的数据在内存空间里只有一份，就在数据类型之前加上 static。

例如： static String China = “中国”；

所有中国人都有国家名称，每一个中国人都共享这个国家的名称，不必再每一个中国人的实例对象中都单独分配一个用于表示国家名称

静态方法:

我们有时候希望不必创建对象就可以调用某个方法，换句话说也就是使该方法不必和对象捆绑在一起，要实现这种效果，只需要在方法名前面加上 Static 关键字即可。

类的静态方法经常被称为“类方法”

注意：

1. 在静态方法里只能直接调用同类中其他的静态成员（包括变量和方法），而不能直接访问类中的非静态变量。使用时不需要创建对象的实例。
2. 静态方法不能以任何方式引用 this,super
3. Main 方法是静态的。

静态代码块:

一个类中可以使用不包含任何方法体的静态代码块，当类被载入时，静态代码块被执行，且只执行一次，静态代码块经常用于进行类属性的初始化。

例如：

```
class StaticCode {  
    static String country;  
    static {  
        country = "china";  
    }  
}
```

4. 用 static 修饰的类的属性称为类的静态属性
5. 用 static 修饰的类的方法称为类的静态方法
6. A: 非静态的成员方法可以访问 static 属性或许方法
7. B: 静态方法只能访问 static 属性或许方法
8. C: 静态方法不能访问非静态的成员属性或者成员方法

8.2.12.5 构造函数及 this 关键字

this 关键字:

代表使用该方法的当前对象， this 关键字的用法实例见下面的构造函数案例；

每创建一个类的实例都去初始化它的所有变量时乏味的，如果一个对象在被创建时就完成了所有的初始工作，将会很简洁，因此 Java 在类里提供了一个特殊的成员函数，叫构造函数（ConstructorMethod）

案例 1：求面积

```
package com.seecen;

import java.util.ArrayList;
import java.util.List;

public class Circle {
    // 类的成员变量
    double radio;
    // 构造函数
    public Circle(double radio) {
        this.radio = radio;
    }
    // 类的成员方法
    public double getArea() {
        return Math.PI * radio * radio;
    }
    public static void main(String[] args) {
        // 调用构造方法实例化
        Circle acircle = new Circle(2.4445);
        // 调用成员方法
        double area = acircle.getArea();
        System.out.println("圆的半径是: " + acircle.radio + "圆的面积是: " +
area);
    }
}
```

结果:

圆的半径是: 2.4445 圆的面积是:18.77283901433626

关注 客户体验

案例 2: 给 Dog 类增加 Dog(name)构造方法

8.2.12.6 final 关键字

该关键字可以应用于变量也可以应用于方法和类。（当一个类为 final 时意味着该类不能有子类）

1. 应用于变量是表示变量的值不会变化，且行为与常量相似。

2. 在方法中声明可以防止该方法被子类重写。

将类声明为 final 意味着该类不能被继承或有子类。通常是用在提高系统安全性。

8.2.12.7 对象的创建 NEW

语法:

类名对象名 = new 类名;

案例:

```
package com.seecen;
import java.util.ArrayList;
import java.util.List;

public class Demo1 {
    public static void main(String[] args) {
        // 对象的创建
        Circle c = new Circle(2.5);
        // 对象的使用
        double x = c.getArea();
    }
}
```

8.2.12.8 类中的方法调用

案例 1: 平均成绩

```
public class ScoreCalc {
    int java; // Java成绩
    int c; // C#成绩
    int db; // DB成绩
    public int calcTotalScore() {
        int total = java + c + db;
        return total;
    }
    public void showTotalScore() {
        System.out.println("总成绩是: " + calcTotalScore());
    }
    public int calcAvg() {
        int avg = (java + c + db) / 3;
        return avg;
    }
    public void showAvg() {
        System.out.println("平均成绩是: " + calcAvg());
    }
}
```

```
import java.util.*;
```

```
public class TestScoreCalc {  
    public static void main(String[] args) {  
        ScoreCalc sc = new ScoreCalc();  
        Scanner input = new Scanner(System.in);  
        System.out.print("请输入Java成绩: ");  
        sc.java = input.nextInt();  
        System.out.print("请输入C#成绩: ");  
        sc.c = input.nextInt();  
        System.out.print("请输入DB成绩: ");  
        sc.db = input.nextInt();  
        sc.showTotalScore();  
        sc.showAvg();  
    }  
}
```

案例 2：领养宠物并打印宠物信息

```
package com.seecen.epepet.obj;  
  
public class Penguin {  
    String name = "无名氏"; // 昵称  
    int health = 100; // 健康值  
    int love = 0; // 亲密度  
    String sex = "Q仔"; // 性别  
  
    public void print() {  
        System.out.println("宠物的自白: \n我的名字叫" + this.name + ",  
        健康值是" + this.health  
        + ", 和主人的亲密度是" + this.love + ", 性别是" + this.sex  
        + ".");  
    }  
}
```

```
package com.seecen.epepet.obj;  
  
public class Dog {  
    String name = "无名氏"; // 昵称, 默认值是"无名氏"  
    int health = 100; // 健康值, , 默认值是100  
    int love = 0; // 亲密度  
    String strain = "聪明的拉布拉多犬"; // 品种  
  
    public void print() {  
        System.out.println("宠物的自白: \n我的名字叫" + this.name + ",  
        健康值是" + this.health  
        + ", 和主人的亲密度是" + this.love + ", 性别是" + this.sex  
        + ".");  
    }  
}
```

```
    健康值是" + this.health
        + "，和主人的亲密度是" + this.love + "，我是一只" +
this.strain + "。");
}
}
```

```
package com.seecen.epepet.obj;

import java.util.Scanner;

public class Test {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.println("欢迎您来到宠物店！");
        // 1、输入宠物名称
        System.out.print("请输入要领养宠物的名字：");
        String name = input.next();
        // 2、选择宠物类型
        System.out.print("请选择要领养的宠物类型：(1、狗狗 2、企鹅)");
        switch (input.nextInt()) {
            case 1:
                // 2.1、如果是狗狗
                // 2.1.1、选择狗狗品种
                System.out.print("请选择狗狗的品种：(1、聪明的拉布拉多犬" +
"2、酷酷的雪娜瑞)");
                String strain = null;
                if (input.nextInt() == 1) {
                    strain = "聪明的拉布拉多犬";
                } else {
                    strain = "酷酷的雪娜瑞";
                }
                // 2.1.2、创建狗狗对象并赋值
                Dog dog = new Dog();
                dog.name = name;
                dog.strain = strain;
                // 2.1.3、输出狗狗信息
                dog.print();
                break;
            case 2:
                // 2.2、如果是企鹅
                // 2.2.1、选择企鹅性别
                System.out.print("请选择企鹅的性别：(1、Q仔 2、Q妹)");
                String sex = null;
                if (input.nextInt() == 1)
```

```
        sex = "Q仔";
    else
        sex = "Q妹";
    // 2.2.2、创建企鹅对象并赋值
    Penguin pgn = new Penguin();
    pgn.name = name;
    pgn.sex = sex;
    // 2.2.3、输出企鹅信息
    pgn.print();
}
}
}
```

8.2.12.9 封装

封装 (Encapsulation) 是面向对象方法的重要原则，就是把对象的属性和操作（或服务）结合为一个独立的整体，并尽可能隐藏对象的内部实现细节。

概念

封装是把过程和数据包围起来，对数据的访问只能通过已定义的接口。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。封装是一种信息隐藏技术，在 java 中通过关键字 `private` 实现封装。什么是封装？封装把对象的所有组成部分组合在一起，封装定义程序如何引用对象的数据，封装实际上使用方法将类的数据隐藏起来，控制用户对类的修改和访问数据的程度。

例子

```
public class Man
{
    //对属性的封装，一个人的姓名，年龄，妻子都是这个对象（人）的私有属性
    private String name;
    private int age;
    private Woman wife;
    //对该人对外界提供方法的封装，可以设定妻子，姓名，年龄也可以获得男人的姓名和年龄
    // 方法封装
    public void setWife(Woman wife) {
        this.wife = wife;
    }
    public String getName() {
        return name;
    }
}
```

```
public void setName(String name) {  
    this.name = name;  
}  
public int getAge() {  
    return age;  
}  
public void setAge(int age) {  
    this.age = age;  
}  
}  
  
public class Woman {  
    // 属性封装  
    private String name;  
    private int age;  
    private Man husband;  
    // 方法封装  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
    public int getAge() {  
        return age;  
    }  
    public void setAge(int age) {  
        this.age = age;  
    }  
    public Man getHusband() {  
        return husband;  
    }  
    public void setHusband(Man husband) {  
        this.husband = husband;  
    }  
}  
/**
```

- * 仔细看就会发现, Man 类没有提供 getWife 的方法, 这是因为男人不想让自己的妻子被外界访问,
- * 直接下来呢, 就是封装可以把一个对象的属性私有, 而提供一些可以被外界访问的属性的方法,
- * 比如说, name 属性, Man 和 Woman 类都有相应的 get 和 set 方法, 外界都可以通过这些方法访问和修改
- *

* 同时对一些对象不想让外界访问的属性，就不提供其方法，比如说 Man 的 wife 属性，就没有 get 方法

```
*  
* 外界是不能得到 Man 类的 wife 属性的  
*  
*/
```

上面那例子可能没有突出封装的好处，下面来个超简单的表达下：

```
public class Show  
{  
    public static void show(String str)  
    {  
        System.out.println(str);  
    }  
}
```

上面就是对 System.out.println(); 的封装。

调用的时候：

```
public class Use  
{  
    public static void main(String[] args)  
    {  
        Show.show("封装");  
    }  
}
```

这样用的时候就不用使：System.out.println("封装");

封装的作用

思诚科技

①对象的数据封装特性彻底消除了传统结构方法中数据与操作分离所带来的种种问题，提高了程序的可复用性和可维护性，降低了程序员保持数据与操作内容的负担。

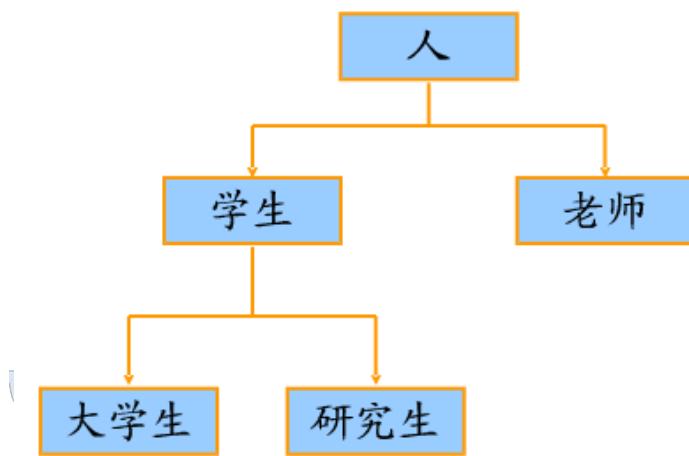
②对象的数据封装特性还可以把对象的私有数据和公共数据分离开，保护了私有数据，减少了可能的模块间干扰，达到降低程序复杂性、提高可控性的目的。^[1]

修饰符	同一个类中	同包下的不同类中	同包子类中	不同包子类中	不同包下的其他类中
Public	√	√	√	√	√
Protected	√	√	√	√	✗
Default	√	√	√	✗	✗
Private	√	✗	✗	✗	✗

8.2.12.10 继承

继承是使用已存在的类的定义作为基础建立新类的技术，新类的定义可以增加新的数据或新的功能，也可以用父类的功能，但不能选择性地继承父类。这种技术使得复用以前的代码非常容易，能够大大缩短开发周期，降低开发费用。比如可以先定义一个类叫车，车有以下属性：车体大小，颜色，方向盘，论坛，而又由车这个类派生出教程和卡车两个类，为轿车添加一个小后备箱，而为卡车添加一个大货箱。

大学系统人员分类树



继承的概念（续）

- ❖ 最高层是最普遍的、最一般的情况，往下每一层都比上一层更具体，并包含有高层的特征，通过这样的层次结构使下层的类能自动享用上层类的特点和性质；
- ❖ 继承其实就是自动地共享基类中成员属性和成员方法的机制。

在Java中实现继承

❖ 在Java中实现继承需要使用到**extends**关键字；

❖ 实现继承的一般语法是：

[访问修饰符] **class** 派生类名 **extends** 基类名 {
 成员列表
}

❖ 如：

```
class Student extends Person  
{  
    ....  
}
```

```
class Person { //定义人类  
    public String mName; //姓名  
    public int mAge; //年龄  
    public void dining() { System.out.println("吃饱了..."); } //吃饭的方法  
}  
  
class Student extends Person { //学生类继承于人类  
    public float mGrade; //成绩  
    public void examination() { System.out.println("考试及格了..."); } //考试的方法  
}  
  
class Teacher extends Person { //教师类继承于人类  
    public float mSalary; //薪水  
    public void prelection() { System.out.println("上课很累..."); } //上课的方法  
}  
  
public class InheritanceDemo { //该类用于容纳main方法  
    public static void main(String[] args) {  
        Student std = new Student(); //实例化学生对象  
        std.mName = "张三"; std.mAge = 18; //为姓名和年龄赋值，访问的是父类中的成员  
        std.dining(); //调用吃饭的方法，访问的是父类中的成员  
        std.examination(); //调用考试方法，访问的是子类中的成员  
  
        Teacher tea = new Teacher(); //实例化教师对象  
        tea.mName = "谭浩强"; tea.mAge = 65;  
        tea.dining();  
        tea.prelection();  
    }  
}
```

继承是面向对象编程中的重要内容，利用继承的方式，在父对象已经定义了的公共的状态和行为的基础上，子对象以父对象为样本，继承父对象的公共方法和特征，并且可以在子对象中增加新的属性和方法。

JAVA 中一个类只能有一个直接父类

平常我们没有定义 **extends** 的类默认是 **Object** 类的子类

如何使用继承

使用继承

编写父类

编写子类，继承父类

```
class Pet {  
    //公共的属性和方法  
}
```

```
class Dog extends Pet {  
    //子类特有的属性和方法  
}
```

```
class Penguin extends Pet {  
}
```

只能继承一个父类

继承关键字

对比

C#用“:”

案例 1：继承创建宠物对象并输出信息

```
package com.seecen.extend;  
public class Pet {  
    private String name = "无名氏"; // 昵称  
    private int health = 100; // 健康值  
    private int love = 0; // 亲密度  
    public Pet() {  
        this.health = 95;  
        System.out.println("执行宠物的无参构造方法。");  
    }  
    public Pet(String name) {  
        this.name = name;  
    }  
    public String getName() {  
        return name;  
    }  
    public int getHealth() {  
        return health;  
    }  
    public int getLove() {  
    }
```

```
        return love;
    }
    public void print() {
        System.out.println("宠物的自白: \n我的名字叫" +
                           this.name + ", 我的健康值是" + this.health
                           + ", 我和主人的亲密程度是" + this.love + "。");
    }
}
```

```
package com.seecen.extend;
public class Dog extends Pet {
    private String strain;// 品种
    public Dog(String name, String strain) {
        super(name); //此处不能使用this.name=name;
        this.strain = strain;
    }
    public String getStrain() {
        return strain;
    }
}
```

```
package com.seecen.extend;
public class Penguin extends Pet {
    private String sex;// 性别
    public Penguin(String name, String sex) {
        super(name);
        this.sex = sex;
    }
    public String getSex() {
        return sex;
    }
    public void setSex(String sex) {
        this.sex = sex;
    }
}
```

```
package com.seecen.extend;
public class Test {
    public static void main(String[] args) {
```

```
// 1、创建宠物对象pet并输出信息
Pet pet = new Pet("贝贝");
pet.print();
// 2、创建狗狗对象dog并输出信息
Dog dog = new Dog("欧欧", "雪娜瑞");
dog.print();
// 3、创建企鹅对象pgn并输出信息
Penguin pgn = new Penguin("楠楠", "Q妹");
pgn.print();
}
```

8.2.12.11 重载与继承中的重写实现 Java 的多态性

继承关系中方法的重写：

子类与父类之间存在方法名相同数列表也相同这个方法叫做方法的重写

如果子类重写了父类的方法那么子类对象中调用的方法就会调用子类中重写的方法

如果子类中想调用父类的方法 super.方法名()

多态的意思就是使用相同的名字来定义不同的方法，在 Java 中普通类型的多态成为重载，这就以为这可以使用相同名字的方法，而方法是不同数量、不同类型的参数来区分

方法重写

方法重载

VS

	位置	方法名	参数表	返回值	访问修饰符
方法重写	子类	相同	相同	相同	不能比父类更严格
方法重载	同类	相同	相同	无关	无关

方法重载



一个类是否可以有多个构造方法?

方法重载，指同一个类中多个方法:

方法名相同

参数列表不同

与返回值、访问修饰符无关



System.out.println(45);
System.out.println(true);
System.out.println("狗在玩耍！");

```
public Penguin() {  
    //代码  
}  
public Penguin(String name, int health, int love, String sex) {  
    //代码  
}
```

示例 1

示例 2

首先我们来讲解：**重载 (Overloading)**

(1) 方法重载是让类以统一的方式处理不同类型数据的一种手段。多个同名函数同时存在，具有不同的参数个数/类型。

重载 Overloading 是一个类中多态性的一种表现。

(2) Java的方法重载，就是在类中可以创建多个方法，它们具有相同的名字，但具有不同的参数和不同的定义。

调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法，这就是多态性。

(3) 重载的时候，方法名要一样，但是参数类型和个数不一样，返回值类型可以相同也可以不相同。无法以返回型别作为重载函数的区分标准。

下面是重载的例子：

```
package com.seecen; //这是包名  
//这是这个程序的第一种编程方法，在 main 方法中先创建一个 Dog 类实例，然后在 Dog 类的构造方法中利用 this 关键字调用不同的 bark 方法。
```

不同的重载方法 bark 是根据其参数类型的不同而区分的。

```
//注意：除构造器以外，编译器禁止在其他任何地方中调用构造器。
package com.seecen;

public class Dog {
    Dog()
    {
        this.bark();
    }
    void bark()//bark()方法是重载方法
    {
        System.out.println("no barking!");
        this.bark("female", 3.4);
    }
    void bark(String m, double l)//注意：重载的方法的返回值都是一样的，
    {
        System.out.println("a barking dog!");
        this.bark(5, "China");
    }
    void bark(int a, String n)//不能以返回值区分重载方法，而只能以“参数类型”和“类名”来
区分
    {
        System.out.println("a howling dog");
    }
}

public static void main(String[] args)
{
    Dog dog = new Dog();
    //dog.bark(); [Page]
    //dog.bark("male", "yellow");
    //dog.bark(5, "China");
```

然后我们再来谈谈**重写 (Overriding)**

(1) 父类与子类之间的多态性，对父类的函数进行重新定义。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在[Java](#)中，子类可继承父类中的方法，而不需要重新编写相同的方法。

但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。

方法重写又称方法覆盖。

(2) 若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。

如需父类中原有的方法，可使用 super 关键字，该关键字引用了当前类的父类。

(3) 子类函数的访问修饰权限不能少于父类的;

下面是重写例子:

概念: 即调用对象方法的机制。

动态绑定的内幕:

1、编译器检查对象声明的类型和方法名，从而获取所有候选方法。试着把上例 Base 类的 test 注释掉，这时再编译就无法通过。

2、重载决策：编译器检查方法调用的参数类型，从上述候选方法选出唯一的那个（其间会有隐含类型转化）。

如果编译器找到多于一个或者没找到，此时编译器就会报错。试着把上例 Base 类的 test (byte b) 注释掉，这时运行结果是 1 1。

3、若方法类型为 private static final，java 采用静态编译，编译器会准确知道该调用哪个方法。

4、当程序运行并且使用动态绑定来调用一个方法时，那么虚拟机必须调用对象的实际类型相匹配的方法版本。

在例子中，b 所指向的实际类型是 TestOverriding，所以 b. test (0) 调用子类的 test。

但是，子类并没有重写 test (byte b)，所以 b. test ((byte)0) 调用的是父类的 test (byte b)。

如果把父类的 (byte b) 注释掉，则通过第二步隐含类型转化为 int，最终调用的是子类的 test (int i)。

学习总结:

多态性是面向对象编程的一种特性，和方法无关，

简单说，就是同样的一个方法能够根据输入数据的不同，做出不同的处理，即方法的重载——有不同的参数列表（静态多态性）

而当子类继承自父类的相同方法，输入数据一样，但要做出有别于父类的响应时，你就要覆盖父类方法，

即在子类中重写该方法——相同参数，不同实现（动态多态性）

OOP 三大特性：继承，多态，封装。

```
public class Base
{
    void test(int i)
    {
        System.out.print(i);
    }
    void test(byte b)
    {
        System.out.print(b);
    }
}
```

```
        }
    }
public class TestOverriding extends Base
{
    void test(int i)
    {
        i++;
        System.out.println(i);
    }
    public static void main(String[] args)
    {
        Base b=new TestOverriding();
        b.test(0)
        b.test((byte)0)
    }
}
```

这时的输出结果是 1 0，这是运行时动态绑定的结果。

重写的主要优点是能够定义某个子类特有的特征：

```
public class Father{
    public void speak() {
        System.out.println(Father);
    }
}
public class Son extends Father{
    public void speak() {
        System.out.println("son");
    }
}
```

这也叫做多态性，重写方法只能存在于具有继承关系中，重写方法只能重写父类非私有的方法。

当上例中 Father 类 speak() 方法被 private 时，Son 类不能重写出 Father 类 speak() 方法，此时 Son 类 speak() 方法相当与在 Son 类中定义的一个 speak() 方法。

Father 类 speak() 方法一旦被 final 时，无论该方法被 public, protected 及默认所修饰时，Son 类根本不能重写 Father 类 speak() 方法，

试图编译代码时，编译器会报错。例：

```
public class Father{
```

```
final public void speak() {  
    System.out.println("Father");  
}  
  
}  
  
public class Son extends Father{  
    public void speak() {  
        System.out.println("son");  
    }  
}  
} //编译器会报错;
```

Father 类 speak() 方法被默认修饰时, 只能在同一包中, 被其子类被重写, 如果不在同一包则不能重写。

Father 类 speak() 方法被 protoeted 时, 不仅在同一包中, 被其子类被重写, 还可以不同包的子类重写。

重写方法的规则:

- 1、参数列表必须完全与被重写的方法相同, 否则不能称其为重写而是重载。
- 2、返回的类型必须一直与被重写的方法的返回类型相同, 否则不能称其为重写而是重载。
- 3、访问修饰符的限制一定要大于被重写方法的访问修饰符 (public>protected>default>private)
- 4、重写方法一定不能抛出新的检查异常或者比被重写方法申明更加宽泛的检查型异常。例如:

父类的一个方法申明了一个检查异常 IOException, 在重写这个方法是就不能抛出 Exception, 只能抛出 IOException 的子类异常, 可以抛出非检查异常。

而重载的规则:

- 1、必须具有不同的参数列表;
- 2、可以有不责骂的返回类型, 只要参数列表不同就可以了;
- 3、可以有不同的访问修饰符;
- 4、可以抛出不同的异常;

重写与重载的区别在于:

重写多态性起作用, 对调用被重载过的方法可以大大减少代码的输入量, 同一个方法名只要往里面传递不同的参数就可以拥有不同的功能或返回值。

用好重写和重载可以设计一个结构清晰而简洁的类, 可以说重写和重载在编写代码过程中的作用非同一般。

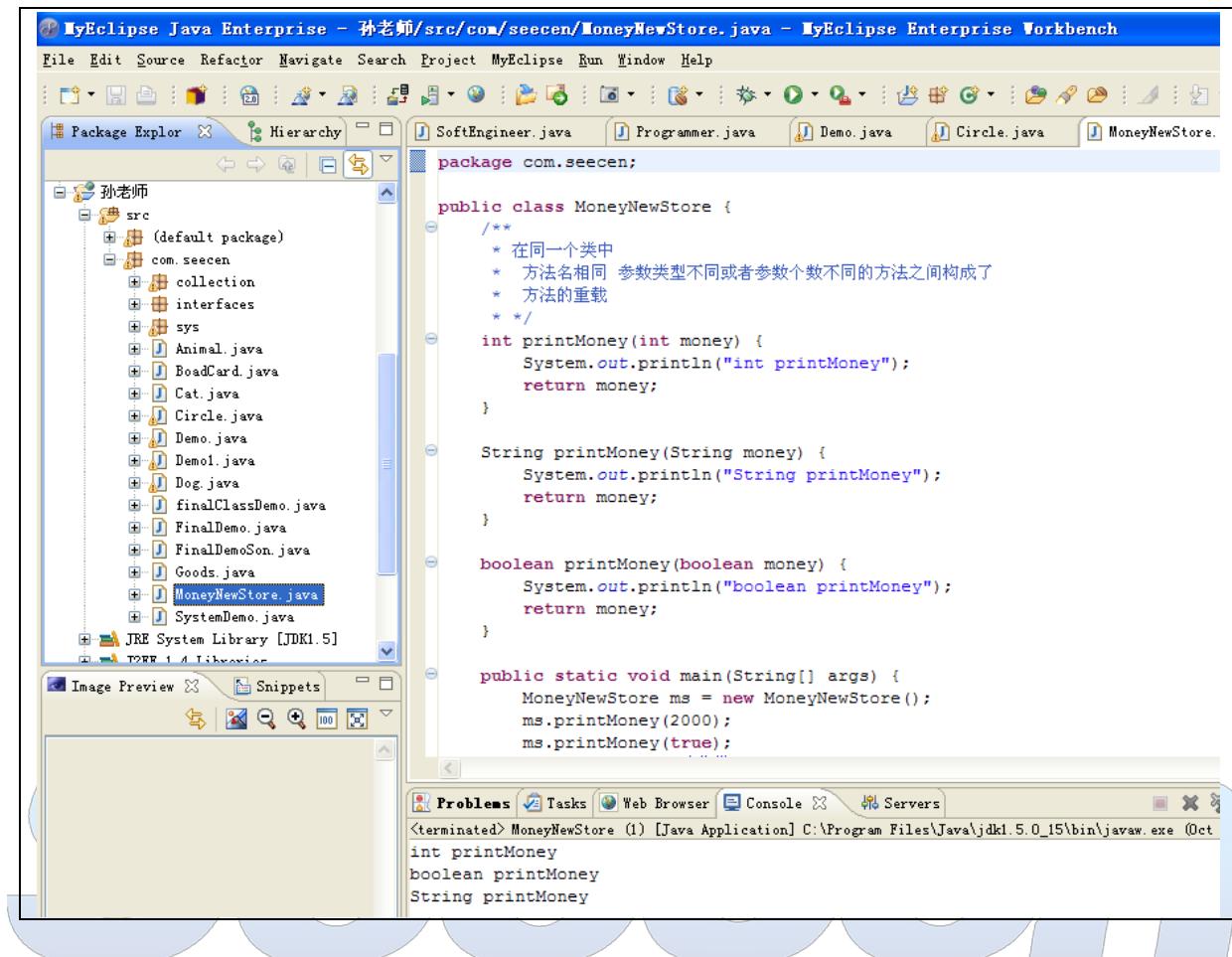
案例：方法重载

```
package com.seecen.reload;
public class MoneyNewStore {
    int printMoney(int money) {
        System.out.println("int printMoney");
        return money;
    }
    String printMoney(String money) {
        System.out.println("String printMoney");
        return money;
    }
    boolean printMoney(boolean money) {
        System.out.println("boolean printMoney");
        return money;
    }
    public static void main(String[] args) {
        MoneyNewStore ms = new MoneyNewStore();
        ms.printMoney(2000);
        ms.printMoney(true);
        ms.printMoney("购物券");
    }
}
```

效果



思诚科技
关注客户体验



案例：重写
父类

思诚科技

关注 客户体验

```
package com.seecen.rewrite;
public abstract class Pet {
    private String name = "无名氏"; // 昵称
    private int health = 100; // 健康值
    private int love = 0; // 亲密度
    public Pet() {
        this.health = 95;
        System.out.println("执行宠物的无参构造方法。");
    }
    public Pet(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public int getHealth() {
        return health;
    }
}
```

```
public int getLove() {
    return love;
}
public void print() {
    System.out.println("宠物的自白: \n我的名字叫" + this.name + "， 健康值是"
+ this.health + "， 和主人的亲密度是" + this.love + "。");
}
```

子类继承父类

```
package com.seecen.rewrite;
public class Dog extends Pet {
    private String strain;// 品种
    private String name;
    public Dog(String name, String strain) {
        this.name=name;// 此处不能使用
        this.strain = strain;
    }
    public String getStrain() {
        return strain;
    }
    public String getName(){
        return name;
    }
    public void print(){
        System.out.println("这是子类的print方法, 不是父类的方法!");
        System.out.println("名字叫: " + getName() +"品种是: "+getStrain());
    }
}
```

子类重写 print 方法实现

```
package com.seecen.rewrite;
public class Test {
    public static void main(String[] args) {
        Dog dog = new Dog("欧欧", "雪娜瑞");
        dog.print();
    }
}
```

8.2.12.12 抽象类、抽象方法

在 Java 语言规范中，程序设计人员可以构造抽象类和抽象方法，抽象定义是指由关键字 `abstract` 定义，在实现内容上没有完全定义的方法，抽象类本身不具备实际的功能，只能用于派生子类，而定义的抽象方法必须在子类派生时重载。

抽象类

- (1) 在 Java 中当一个类被 `abstract` 关键字修饰时，该类就叫抽象类。
- (2) 抽象类是从多个具体类中抽象出来的父类，属于高层次的抽象。

抽象类遵循的原则

- (1) 抽象类必须使用 `abstract` 关键字进行修饰。
- (2) 抽象类不能被实例化（无法使用 `new` 关键字创建对象实例）。
- (3) 抽象类可以包含属性，方法，构造方法，初始化块，内部类，枚举类。
- (4) 含有抽象方法的类必须定义成抽象类。

抽象类的语法

```
abstract class <类名>{  
    [属性声明]  
    [方法声明]  
}
```

抽象方法

- (1) 在 Java 中当一个类的方法被 `abstract` 关键字修饰后，该方法就叫抽象方法。
- (2) 抽象方法所在的类必须定义为抽象类。

抽象方法的语法

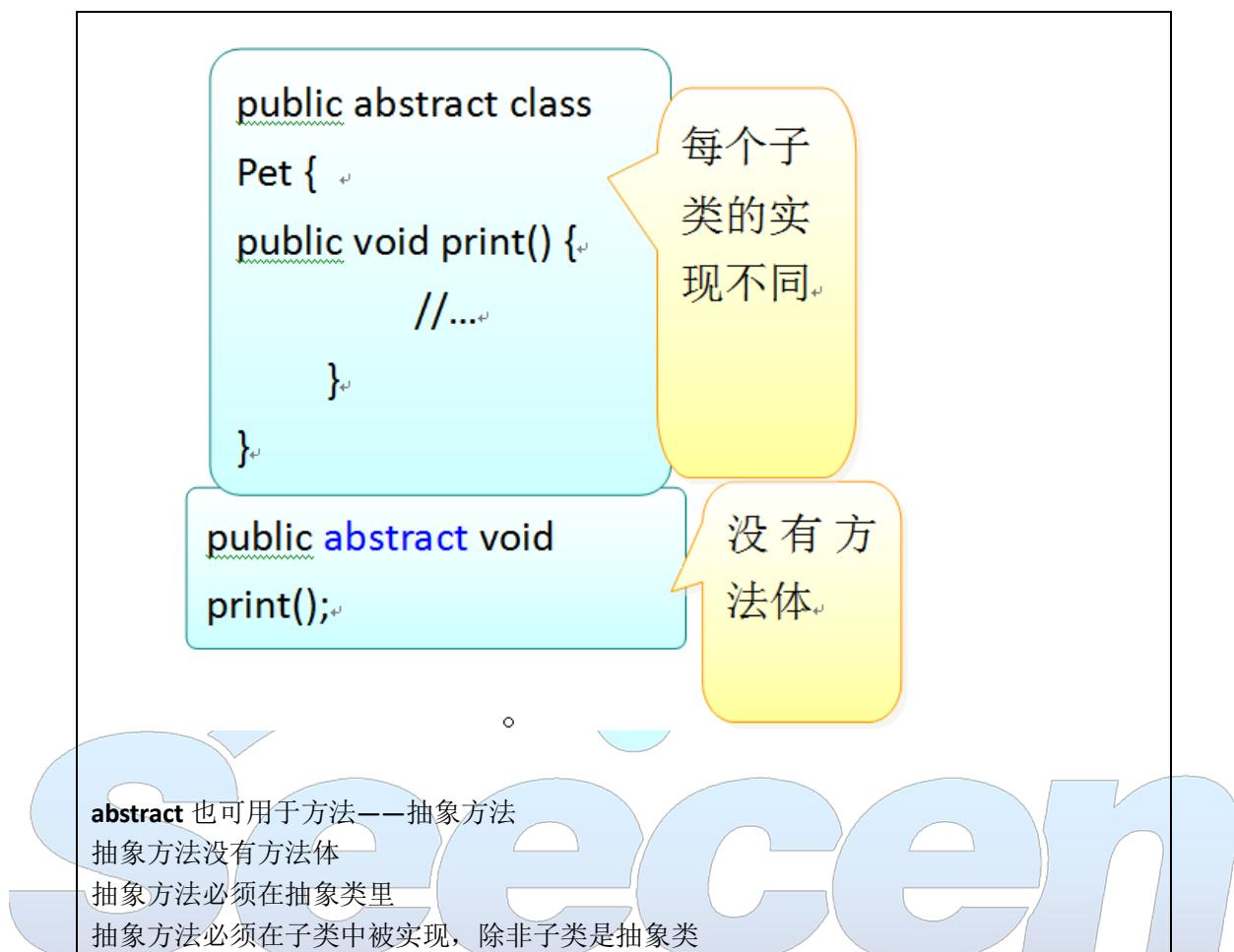
```
[访问修饰符] abstract <返回值类型> <方法名> ([参数列表]) ;
```

注意事项

- (1) 抽象类与抽象方法都必须使用 `abstract` 关键字进行修饰，但不能使用 `abstract` 关键字修饰属性或局部变量。



以下代码有什么问题？



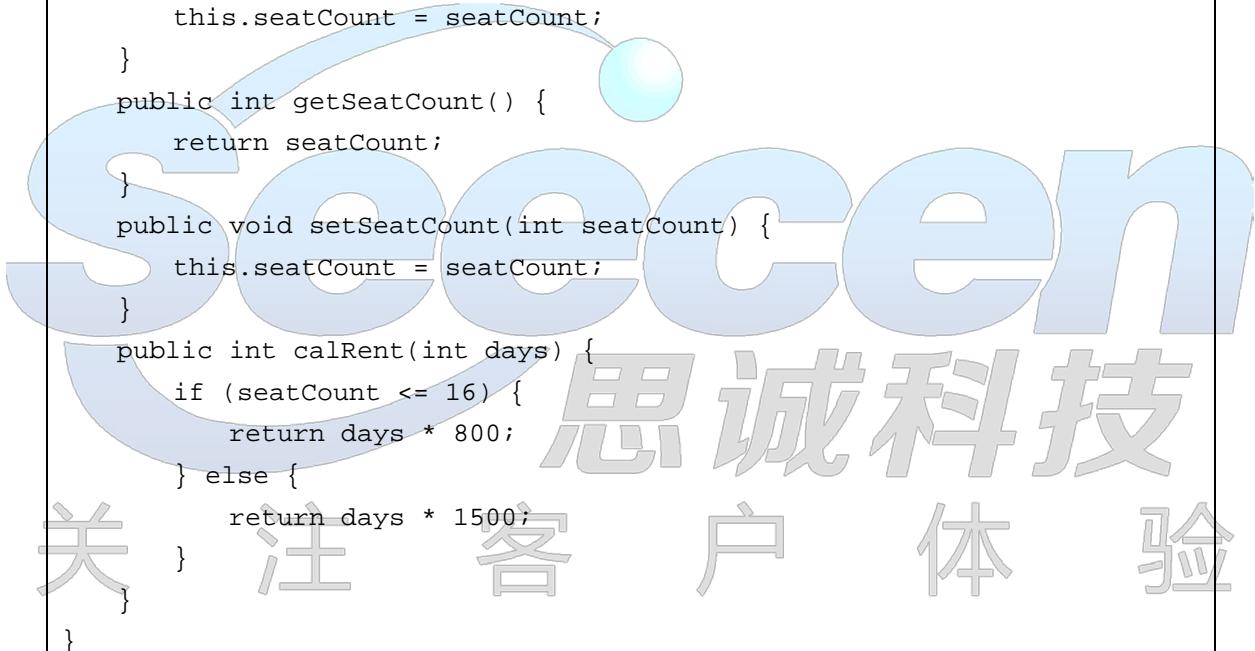
在面向对象领域，抽象类主要用来进行类型隐藏。我们可以构造出一个固定的一组行为的抽象描述，但是这组行为却能够有任意个可能的具体实现方式。这个抽象描述就是抽象类，而这一组任意个可能的具体实现则表现为所有可能的派生类。模块可以操作一个抽象体。由于模块依赖于一个固定的抽象体，因此它可以是不允许修改的；同时，通过从这个抽象体派生，也可扩展此模块的行为功能。熟悉 OCP 的读者一定知道，为了能够实现面向对象设计的一个最核心的原则 OCP(Open-Closed Principle)，抽象类是其中的关键所在。

案例：增加租赁卡车业务，计算汽车租赁的总租金

```
package com.seecen.abstract1;  
public abstract class MotoVehicle {  
    private String no;// 汽车牌号  
    private String brand;// 汽车品牌  
    public MotoVehicle() {  
    }  
    public MotoVehicle(String no, String brand) {  
        this.no = no;  
        this.brand = brand;  
    }  
    public String getNo() {  
        return no;  
    }
```

```
public String getBrand() {
    return brand;
}
public abstract int calRent(int days);
}
```

```
package com.seecen.abstract1;
public final class Bus extends MotoVehicle {
    private int seatCount;// 座位数
    public Bus() {
    }
    public Bus(String no, String brand, int seatCount) {
        super(no, brand);
        this.seatCount = seatCount;
    }
    public int getSeatCount() {
        return seatCount;
    }
    public void setSeatCount(int seatCount) {
        this.seatCount = seatCount;
    }
    public int calRent(int days) {
        if (seatCount <= 16) {
            return days * 800;
        } else {
            return days * 1500;
        }
    }
}
```



```
package com.seecen.abstract1;
public final class Car extends MotoVehicle {
    private String type;// 汽车型号
    public Car() {
    }
    public Car(String no, String brand, String type) {
        super(no, brand);
        this.type = type;
    }
    public String getType() {
        return type;
    }
}
```

```
    }
    public void setType(String type) {
        this.type = type;
    }
    public int calRent(int days) {
        if ("1".equals(type)) {// 代表550i
            return days * 500;
        } else if ("2".equals(type)) {// 2代表商务舱GL8
            return 600 * days;
        } else {
            return 300 * days;
        }
    }
}
```



```
package com.seecen.abstract1;

import java.util.Scanner;
public class TestRent {
    public static void main(String[] args) {
        String no,brand,mtype,type;
        int seatCount,days,rent;
        Car car;
        Bus bus;
        Scanner input = new Scanner(System.in);
        System.out.println("欢迎您来到汽车租赁公司！");
        System.out.print("请输入要租赁的天数：");
        days=input.nextInt();
        System.out.print("请输入要租赁的汽车类型（1: 轿车      2、客车）：");
        mtype = input.next();
        if("1".equals(mtype)){
            System.out.print("请输入要租赁的汽车品牌（1、宝马      2、别克）：");
            brand=input.next();
            System.out.print("请输入轿车的型号");
            if("1".equals(brand))
                System.out.print("1、550i: ");
            else
                System.out.print("2、商务舱GL8  3、林荫大道");
            type=input.next();
            no="京BK5543";//简单起见，直接指定车牌号
            System.out.println("分配给您的车牌号是："+no);
            car =new Car(no,brand,type);
            rent=car.calRent(days);
        }
```

```
    }
    else{
        System.out.print("请输入要租赁的客车品牌 (1、金杯 2、金龙) :");
        brand=input.next();
        System.out.print("请输入客车的座位数:");
        seatCount=input.nextInt();
        no="京AU8769";//简单起见，直接指定汽车牌号
        System.out.println("分配给您的汽车牌号是:"+no);
        bus=new Bus(no,brand,seatCount);
        rent=bus.calRent(days);
    }
    System.out.println("\n顾客您好！您需要支付的租赁费用是"+rent+"。");
}
}
```

8.2.12.13 接口

Java 语言中规范中定义了一种被称为“接口”（interface）的抽象类，通过利用关键字 `interface` 来指明一个类必须做什么，但不需要明确改怎么做。

- ◆ 接口的实现类一般需要实现接口中的所有方法，除非实现类定义为抽象(`abstract`)类型
- ◆ 一个接口可以继承另一个接口，在这种情况下，如果一个类实现的是一个继承其他接口的接口，那么该实现类必须实现这个接口及其父接口的所有方法

1、定义接口

使用 `interface` 来定义一个接口。接口定义同类的定义类似，也是分为接口的声明和接口体，其中接口体由常量定义和方法定义两部分组成。定义接口的基本格式如下：

```
[修饰符] interface 接口名 [extends 父接口名列表]{  
[public][static][final]常量;  
[public][abstract] 方法;  
}
```

修饰符：可选，用于指定接口的访问权限，可选值为 `public`。如果省略则使用默认的访问权限。

接口名：必选参数，用于指定接口的名称，接口名必须是合法的 Java 标识符。一般情况下，要求首字母大写。

extends 父接口名列表：可选参数，用于指定要定义的接口继承于哪个父接口。当使用 `extends` 关键字时，父接口名为必选参数。

方法：接口中的方法只有定义而没有被实现。

例如，定义一个用于计算的接口，在该接口中定义了一个常量 PI 和两个方法，具体代码如下：

```
public interface CallInterface{  
    final float PI=3.14159f;//定义用于表示圆周率的常量 PI
```

```
float getArea(float r); // 定义一个用于计算面积的方法 getArea()
float getCircumference(float r); // 定义一个用于计算周长的方法 getCircumference()
}
```

注意：

与 Java 的类文件一样，接口文件的文件名必须与接口名相同。

实现接口

接口在定义后，就可以在类中实现该接口。在类中实现接口可以使用关键字 `implements`，其基本格式如下：

```
[修饰符] class <类名> [extends 父类名] [implements 接口列表]{  
}
```

修饰符：可选参数，用于指定类的访问权限，可选值为 `public`、`abstract` 和 `final`。

类名：必选参数，用于指定类的名称，类名必须是合法的 Java 标识符。一般情况下，要求首字母大写。
extends 父类名：可选参数，用于指定要定义的类继承于哪个父类。当使用 `extends` 关键字时，父类名为必选参数。

implements 接口列表：可选参数，用于指定该类实现的是哪些接口。当使用 `implements` 关键字时，接口列表为必选参数。当接口列表中存在多个接口名时，各个接口名之间使用逗号分隔。
在类中实现接口时，方法的名字、返回值类型、参数的个数及类型必须与接口中的完全一致，并且必须实现接口中的所有方法。例如，编写一个名称为 `Cire` 的类，该类实现上面定义的接口 `Calculate`，具体代码如下：

```
public class Cire implements CallInterface  
{  
    public float getArea(float r)  
    {  
        float area=PI*r*r; // 计算圆面积并赋值给变量 area  
        return area; // 返回计算后的圆面积  
    }  
    public float getCircumference(float r)  
    {  
        float circumference=2*PI*r; // 计算圆周长并赋值给变量 circumference  
        return circumference; // 返回计算后的圆周长  
    }  
    public static void main(String[] args)  
    {  
        Cire c = new Cire();  
        float f = c.getArea(2.0f);  
        System.out.println(Float.toString(f));  
    }  
}
```

}

在类的继承中，只能做单重继承，而实现接口时，一次则可以实现多个接口，每个接口间使用逗号“,”分隔。这时就可能出现常量或方法名冲突的情况，解决该问题时，如果常量冲突，则需要明确指定常量的接口，这可以通过“接口名.常量”实现。如果出现方法冲突时，则只要实现一个方法就可以了。下面通过一个具体的实例详细介绍以上问题的解决方法。

打印机实现原理：

案例 1：软件工程师编写代码、讲解业务

父类

```
package com.seecen;
public interface Person {
    public String getName();
}
```

子类 1

```
package com.seecen;
public interface BizAgent extends Person {
    public void giveBizSpeech();
}
```

子类 2

```
package com.seecen;
public interface Programmer extends Person {
    public void writeProgram();
}
```

实现类实现接口及父类的所有方法

```
package com.seecen.impl;

import com.seecen.BizAgent;
import com.seecen.Programmer;
public class SoftEngineer implements Programmer, BizAgent {
    private String name;// 软件工程师姓名
    public SoftEngineer(String name) {
        this.name = name;
    }
    public String getName() {
        return name;
    }
    public void giveBizSpeech() {
```

```
        System.out.println("我会讲业务。");
    }
    public void writeProgram() {
        System.out.println("我会写代码。");
    }
}
```

代码运行

```
package com.seecen;

import com.seecen.impl.SoftEngineer;
public class Test {
    public static void main(String[] args) {
        Programmer programmer = new SoftEngineer("小明");
        System.out.println("我是一名软件工程师，我的名字叫"
                +programmer.getName()+"。");
        programmer.writeProgram();
        BizAgent bizAgent=(BizAgent)programmer;
        bizAgent.giveBizSpeech();
    }
}
```

案例 2：一封家书
接口类

```
package com.seecen.letter;
public interface HomeLetter {

    /**
     * 书写称谓。
     */
    public void writeTitle();
    /**
     * 书写问候。
     */
    public void writeHello();
    /**
     * 书写内容。
     */
    public void writeBody();
    /**
     * 书写祝福。
     */
}
```

```
public void writeGreeting();
/**
 * 书写落款。
 */
public void writeSelf();
}
```

实现类

```
package com.seecen.letter.impl;

import com.seecen.letter.HomeLetter;
public class HomeLetterImpl implements HomeLetter {
    public void writeBody() {
        System.out.println("我在这里挺好的。\\n" +
                           "我会努力学习的，已经学到 Java OOP 啦！\\n" +
                           "您二老保重身体啊！\\n");
    }
    public void writeGreeting() {
        System.out.println("此致\\n 敬礼");
    }
    public void writeHello() {
        System.out.println("你们好吗？\\n");
    }
    public void writeSelf() {
        System.out.println("\\t\\t 周杰 \\n\\t2010 年 6 月 1 日");
    }
    public void writeTitle() {
        System.out.println("亲爱的爸爸、妈妈：");
    }
}
```

```
package com.seecen.letter;
public class HomeLetterWriter {
    public static void write(HomeLetter letter){
        letter.writeTitle();
        letter.writeHello();
        letter.writeBody();
        letter.writeGreeting();
        letter.writeSelf();
    }
}
```

展现类

```
package com.seecen.letter;
import com.seecen.letter.impl.HomeLetterImpl;
public class Test {
    public static void main(String[] args) {
        //1、创建家书对象
        HomeLetter letter = new HomeLetterImpl();
        //2、书写家书
        HomeLetterWriter.write(letter);
    }
}
```

8.2.12.14 深入理解 Java 的接口和抽象类

一. 抽象类

在了解抽象类之前，先来了解一下抽象方法。抽象方法是一种特殊的方法：它只有声明，而没有具体的实现。抽象方法的声明格式为：

```
1 abstract void fun();
```

抽象方法必须用 `abstract` 关键字进行修饰。如果一个类含有抽象方法，则称这个类为抽象类，抽象类必须在类前用 `abstract` 关键字修饰。因为抽象类中含有无具体实现的方法，所以不能用抽象类创建对象。

下面要注意一个问题：在《JAVA 编程思想》一书中，将抽象类定义为“包含抽象方法的类”，但是后面发现如果一个类不包含抽象方法，只是用 `abstract` 修饰的话也是抽象类。也就是说抽象类不一定必须含有抽象方法。个人觉得这个属于钻牛角尖的问题吧，因为如果一个抽象类不包含任何抽象方法，为何还要设计为抽象类？所以暂且记住这个概念吧，不必去深究为什么。

```
1 [public] abstract class className {
2     abstract void fun();
3 }
```

从这里可以看出，抽象类就是为了继承而存在的，如果你定义了一个抽象类，却不去继承它，那么等于白白创建了这个抽象类，因为你不能用它来做任何事情。对于一个父类，如果它的某个方法在父类中实现出来没有任何意义，必须根据子类的实际需求来进行不同的实现，那么就可以将这个方法声明为 `abstract` 方法，此时这个类也就成为 `abstract` 类了。

包含抽象方法的类称为抽象类，但并不意味着抽象类中只能有抽象方法，它和普通类一样，同样可以拥有成员变量和普通的成员方法。注意，抽象类和普通类的主要有三点区别：

1) 抽象方法必须为 `public` 或者 `protected`（因为如果为 `private`，则不能被子类继承，子类便无法实现该方法），缺省情况下默认为 `public`。

2) 抽象类不能用来创建对象；

3) 如果一个类继承于一个抽象类，则子类必须实现父类的抽象方法。如果子类没有实现父类的抽象方法，则必须将子类也定义为 **abstract** 类。

在其他方面，抽象类和普通的类并没有区别。

二. 接口

接口，英文称作 **interface**，在软件工程中，接口泛指供别人调用的方法或者函数。从这里，我们可以体会到 Java 语言设计者的初衷，它是对行为的抽象。在 Java 中，定一个接口的形式如下：

```
1 [public] interface InterfaceName {  
2  
3 }
```

接口中可以含有变量和方法。但是要注意，接口中的变量会被隐式地指定为 **public static final** 变量（并且只能是 **public static final** 变量，用 **private** 修饰会报编译错误），而方法会被隐式地指定为 **public abstract** 方法且只能是 **public abstract** 方法（用其他关键字，比如 **private**、**protected**、**static**、**final** 等修饰会报编译错误），并且接口中所有的方法不能有具体的实现，也就是说，接口中的方法必须都是抽象方法。从这里可以隐约看出接口和抽象类的区别，接口是一种极度抽象的类型，它比抽象类更加“抽象”，并且一般情况下不在接口中定义变量。

要让一个类遵循某组特定的接口需要使用 **implements** 关键字，具体格式如下：

```
1 classClassName implements Interface1, Interface2, [...] {  
2 }
```

可以看出，允许一个类遵循多个特定的接口。如果一个非抽象类遵循了某个接口，就必须实现该接口中的所有方法。对于遵循某个接口的抽象类，可以不实现该接口中的抽象方法。

三. 抽象类和接口的区别

1. 语法层面上的区别

- 1) 抽象类可以提供成员方法的实现细节，而接口中只能存在 **public abstract** 方法；
- 2) 抽象类中的成员变量可以是各种类型的，而接口中的成员变量只能是 **public static final** 类型的；
- 3) 接口中不能含有静态代码块以及静态方法，而抽象类可以有静态代码块和静态方法；
- 4) 一个类只能继承一个抽象类，而一个类却可以实现多个接口。

2. 设计层面上的区别

- 1) 抽象类是对一种事物的抽象，即对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。举个简单的例子，飞机和鸟是不同类的事物，但是它们都有一个共性，就是都会飞。那么在设计的时候，可以将飞机设计为一个类 **Airplane**，将鸟设计为一个类 **Bird**，但是不能将飞行这个特性也设计为类，因此它只是一个行为特性，并不是对一类事物的抽象描述。此时可以将飞行设计为一个接口 **Fly**，包含方法 **fly()**，然后 **Airplane** 和 **Bird** 分别根据自己的需要实现 **Fly** 这个接口。然后至于有不同种类的飞机，比如战斗机、民用飞机

等直接继承 Airplane 即可，对于鸟也是类似的，不同种类的鸟直接继承 Bird 类即可。从这里可以看出，继承是一个“是不是”的关系，而接口实现则是“有没有”的关系。如果一个类继承了某个抽象类，则子类必定是抽象类的种类，而接口实现则是有没有、具备不具备的关系，比如鸟是否能飞（或者是否具备飞行这个特点），能飞行则可以实现这个接口，不能飞行就不实现这个接口。

2) 设计层面不同，抽象类作为很多子类的父类，它是一种模板式设计。而接口是一种行为规范，它是一种辐射式设计。什么是模板式设计？最简单例子，大家都用过 ppt 里面的模板，如果用模板 A 设计了 ppt B 和 ppt C，ppt B 和 ppt C 公共的部分就是模板 A 了，如果它们的公共部分需要改动，则只需要改动模板 A 就可以了，不需要重新对 ppt B 和 ppt C 进行改动。而辐射式设计，比如某个电梯都装了某种报警器，一旦要更新报警器，就必须全部更新。也就是说对于抽象类，如果需要添加新的方法，可以直接在抽象类中添加具体的实现，子类可以不进行变更；而对于接口则不行，如果接口进行了变更，则所有实现这个接口的类都必须进行相应的改动。

下面看一个网上流传最广泛的例子：门和警报的例子：门都有 open() 和 close() 两个动作，此时我们可以定义通过抽象类和接口来定义这个抽象概念：

```
1 abstractclassDoor {  
2     publicabstractvoidopen();  
3     publicabstractvoidclose();  
4 }
```

或者：

```
1 interfaceDoor {  
2     publicabstractvoidopen();  
3     publicabstractvoidclose();  
4 }
```

但是现在如果我们需要门具有报警 alarm() 的功能，那么该如何实现？下面提供两种思路：

1) 将这三个功能都放在抽象类里面，但是这样一来所有继承于这个抽象类的子类都具备了报警功能，但是有的门并不一定具备报警功能；

2) 将这三个功能都放在接口里面，需要用到报警功能的类就需要实现这个接口中的 open() 和 close()，也许这个类根本就不具备 open() 和 close() 这两个功能，比如火灾报警器。

从这里可以看出，Door 的 open()、close() 和 alarm() 根本就属于两个不同范畴内的行为，open() 和 close() 属于门本身固有的行为特性，而 alarm() 属于延伸的附加行为。因此最好的解决办法是单独将报警设计为一个接口，包含 alarm() 行为，Door 设计为单独的一个抽象类，包含 open 和 close 两种行为。再设计一个报警门继承 Door 类和实现 Alarm 接口。

```
1 interfaceAlram {  
2     voidalarm();  
3 }  
4 abstractclassDoor {  
5     voidopen();  
6     voidclose();  
7 }  
8 classAlarmDoor extendsDoor implementsAlarm {  
9     voidopen() {  
10         //....  
11     }  
12     voidclose() {  
13         //....  
14     }  
15     voidalarm() {  
16         //....  
17     }  
18 }  
19 }
```

8.2.12.15 本节作业

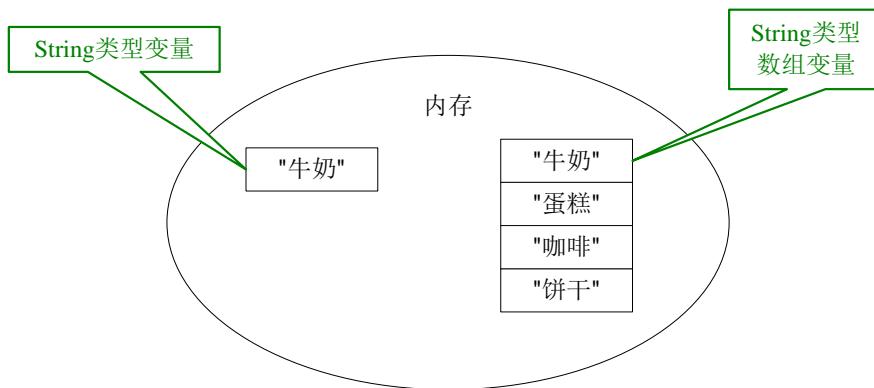
- 1、学生类的对象实例化使用
- 2、学生类的方法重载练习
- 3、鸟、鱼的抽象类继承练习
- 4、打印机的继承练习
- 5 接口练习宠物商店
- 6、接口和继承的练习：
狗吃饭、狗玩接飞盘游戏、企鹅游泳、输出企鹅信息

8.2.13 数组

数组是一个变量，存储相同数据类型的一组数据

声明一个**变量**就是在内存空间划出一块合适的空间

声明一个**数组**就是在内存空间划出一串连续的空间



8.2.13.1 结构和基本要素

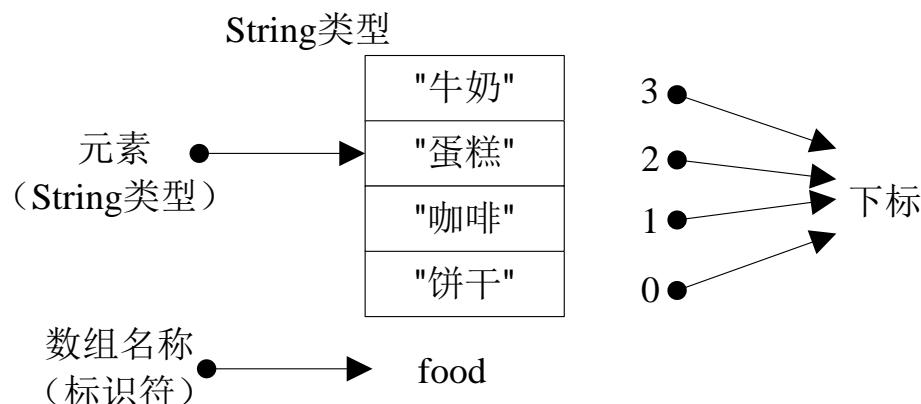


数组只有一个名称，即标识符

元素下标标明了元素在数组中的位置，从 0 开始

数组中的每个元素都可以通过下标来访问

数组长度固定不变，避免数组越界



○ 使用数组四步走:

1、声明数组

```
int[] a;
```

2、分配空间

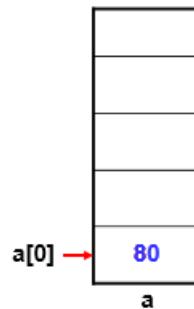
```
a = new int[5];
```

3、赋值

```
a[0] = 8;
```

4、处理数据

```
a[0] = a[0] * 10;
```



语法:

```
int[ ] score1;           //Java 成绩  
int score2[ ];          //C#成绩  
String[ ] name;         //学生姓名制定统一的 web 标准
```



声明数组时不规定
数组长度表头<th>

公告栏

数据类型 数组名[] ;控件中输入的

数据类型[] 数组名 ;数据主体



8.2.13.2 案例：统计字符出现次数

```
package com.seecen.classandobject;
import java.util.Scanner;
public class Counter {
    public int counter(String inputs, String word){
        int counter=0; //计数器，初始化0
        String[] temps = new String[inputs.length()];
        //字符串转换成数组
        for(int i=0;i<temps.length;i++){
            temps[i]=inputs.substring(i,i+1);
        }
        //比较字母，计数
        for(int j=0;j<temps.length;j++){
            if(temps[j].equals(word)){
                counter++;
            }
        }
        return counter;
    }
    public static void main(String[] args) {
        Scanner myinput = new Scanner(System.in);
        Counter cnt = new Counter();
        //定义2个变量，接收用户的输入
        String myString;
        String myWord;
        System.out.print("请输入一个字符串: ");
        myString=myinput.next();
        System.out.print("请输入要查找的字符: ");
        myWord=myinput.next();
        //调用方法，输出结果
        int geshu=cnt.counter(myString,myWord);
        System.out.print("\\" +myString+"\\" 中包含 "+geshu+" 个
\\ "+myWord+"\\" 。 ");
    }
}
```

关注思诚科技 关注思诚科技

{}

8.2.13.3 案例：登入系统并购买商品

```
package com.seecen.classandobject;

import java.util.Scanner;
public class Goods {

    String[] goods = new String[]{"电风扇", "洗衣机", "电视机", "冰箱", "空调机"};
    double[] price= new
    double[]{124.23,4500,8800.90,5000.88,4456,12000.46};
    public boolean login(){
        boolean flag = false;
        Scanner input = new Scanner(System.in);
        System.out.print("请输入用户名: ");
        String name=input.next();
        System.out.print("请输入密码: ");
        String pwd=input.next();
        if(name.equals("TOM")&&pwd.equals("123")){
            System.out.println("登录成功!");
            flag=true;
        }else{
            System.out.println("用户名或密码不匹配，登录失败!");
        }
        return flag;
    }
    public StringBuffer change(double d){
        StringBuffer str=new StringBuffer(String.valueOf(d));
        for(int i=str.indexOf(".")-3;i>0;i=i-3){
            str.insert(i,',');
        }
        return str;
    }
    public void showGoods(){
        System.out.print("*****欢迎进入商品批发城*****");
        System.out.print("\n\t编号\t商品\t价格\n");
        for(int i=0;i<goods.length;i++){
            System.out.print("\t"+(i+1));
            System.out.print("\t"+goods[i]);
        }
    }
}
```

```
        System.out.print("\t"+change(price[i])+"\n");
    }
    System.out.println("*****");
}
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    Goods g = new Goods();
    int serial,num;
    double totle = 0;
    if(g.login()){
        g.showGoods();
        System.out.print("请输入您批发的商品编号: ");
        serial=input.nextInt();
        System.out.print("请输入批发数量: ");
        num=input.nextInt();
        totle=g.price[serial-1]*num;//计算总金额
        System.out.print("您需要付款: "+g.change(totle));
    }
}
```

8.2.13.4 案例：数组声明

```
Public class Demo {
    public static void main(String[] args) {
        int[] score = { 60, 80, 90, 70, 85 };
        int sum = 0;
        double avg;
        for (int i = 0; i < score.length; i++) {
            sum = sum + score[i];
        }
        avg = sum / score.length;
        System.out.println("5位同学的平均成绩是" + avg);
    }
}
```

8.2.13.5 案例：冒泡和选择排序算法

```
package com.seecen;
```

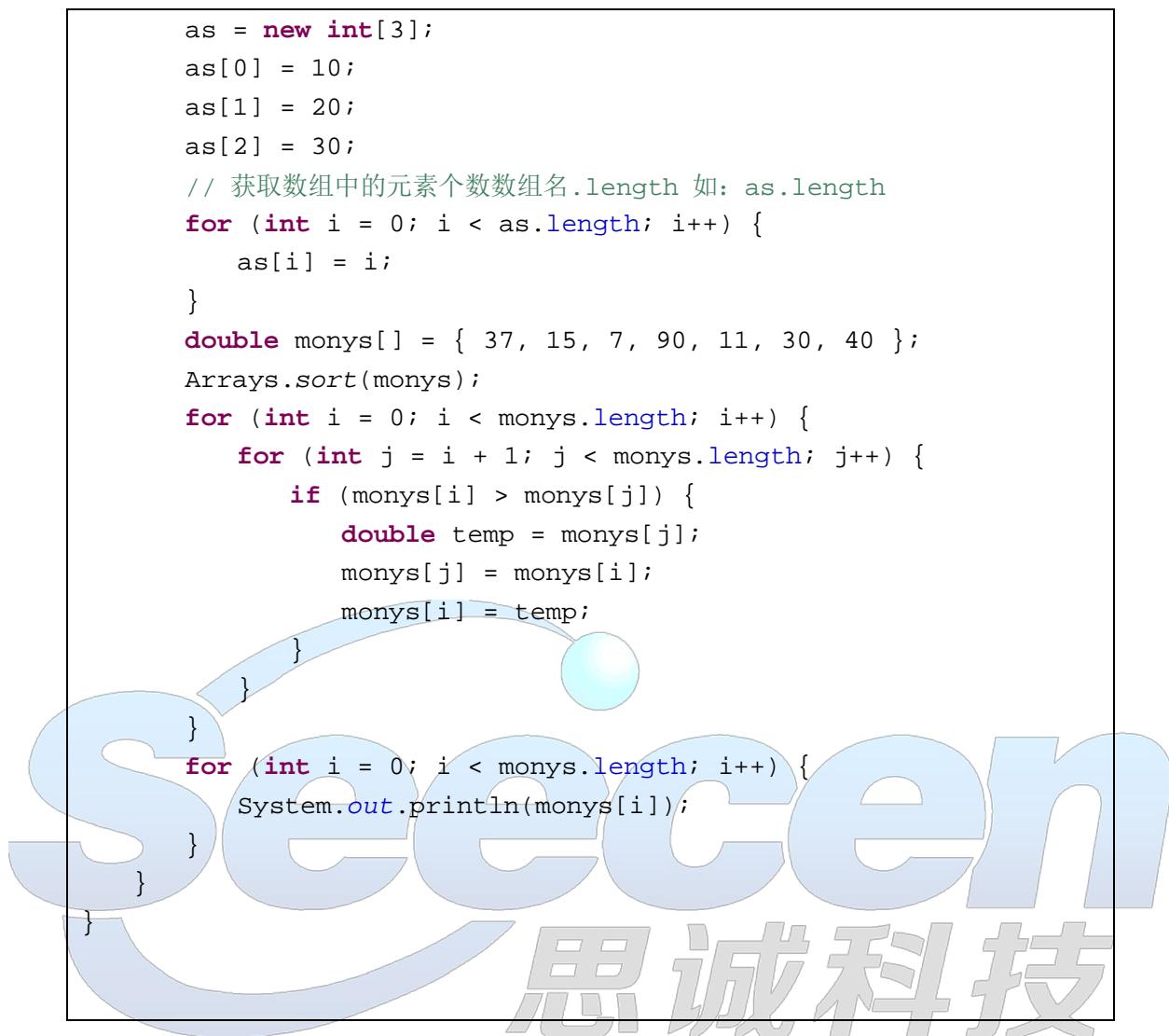
```
import java.util.Arrays;
public class Demo {
    public static void main(String[] args) {
        int[] ary = { 8, 2, 3, 7, 1 };
        long l1 = System.currentTimeMillis();//
        selectSort(ary);
        // bubbleSort(ary);
        long l2 = System.currentTimeMillis();//
        // System.out.println(l2-l1); //Arrays.sort(ary);
        System.out.println(Arrays.toString(ary));
    }
    public static void selectSort(int[] ary) {
        for (int i = 0; i < ary.length - 1; i++) {
            for (int j = i + 1; j < ary.length; j++) {
                if (ary[i] > ary[j]) {
                    int t = ary[i];
                    ary[i] = ary[j];
                    ary[j] = t;
                }
            }
        }
    }
    public static void bubbleSort(int[] ary) {
        for (int i = 0; i < ary.length - 1; i++) {
            for (int j = 0; j < ary.length - 1; j++) {
                if (ary[j] > ary[j + 1]) {
                    int t = ary[j];
                    ary[j] = ary[j + 1];
                    ary[j + 1] = t;
                }
            }
        }
    }
}
```

关注客户体验

8.2.13.6 案例：数组排序

```
import java.util.Arrays;

public class Demo {
    public static void main(String[] args) {
        int as[];
```



8.2.13.7 案例：输入水果后重新排序

```
Package com.seecen.classandobject;

import java.util.*;

public class SortFruit {
    public static void main(String[] args) {
        String fruit[] = new String[5];
        Scanner input = new Scanner(System.in);
        for (int i = 0; i < fruit.length; i++) {
            System.out.print("请输入第" + (i + 1) + "种水果: ");
            fruit[i] = input.next();
        }
        Arrays.sort(fruit);
        System.out.println("\n这些水果在字典中出现的顺序是: ");
    }
}
```

```
    for (int i = 0; i < fruit.length; i++) {
        System.out.println(fruit[i]);
    }
}
```

8.2.13.8 案例：逆序

```
import java.util.Arrays;
public class Demo {
    public static void main(String[] args) {
        char[] chars = new char[] { 'a', 'c', 'u', 'b', 'e', 'p', 'f',
        'z' };
        System.out.print("原字符序列: ");
        for (int i = 0; i < chars.length; i++) {
            System.out.print(chars[i] + " ");
        }
        Arrays.sort(chars); // 对数组进行升序排序
        System.out.print("\n升序排序后: ");
        for (int i = 0; i < chars.length; i++) {
            System.out.print(chars[i] + " ");
        }
        System.out.print("\n逆序输出为: ");
        for (int i = chars.length - 1; i >= 0; i--) {
            System.out.print(chars[i] + " ");
        }
    }
}
```

8.2.13.9 案例：数组插入算法

```
package com.seecen;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

public class Demo {
```

```
public static void main(String[] args) {
    // 在排序好的数组插入一个数字，数字要插入到合适的位置上
    int intArr[] = { 11, 22, 33, 44, 55, 66, 77 };
    int insertNum = 34;
    // 找到要插入的位置
    int insertIndex = 0;
    for (int i = 0; i < intArr.length; i++) {
        if (insertNum < intArr[i]) {
            insertIndex = i;
            break;
        }
    }
    // 将insertNum放入要插入的位置，然后后面每个都向后移动一位角标
    int intArr1[] = new int[intArr.length + 1];
    for (int i = 0; i < intArr1.length; i++) {
        if (i >= insertIndex) {
            if (i == insertIndex) // 这个只做一次
                intArr1[i] = insertNum;
            if (i + 1 < intArr1.length) // i+1会越界，加判断
                intArr1[i + 1] = intArr[i];
            } else {
                // 插入之前执行，开始执行插入以后就不执行
                intArr1[i] = intArr[i];
            }
        }
    }
    // 循环输出
    for (int i : intArr1) {
        System.out.print(i + " ");
    }
}
```

关注客户体验

8.2.13.10 案例：数组截取拷贝

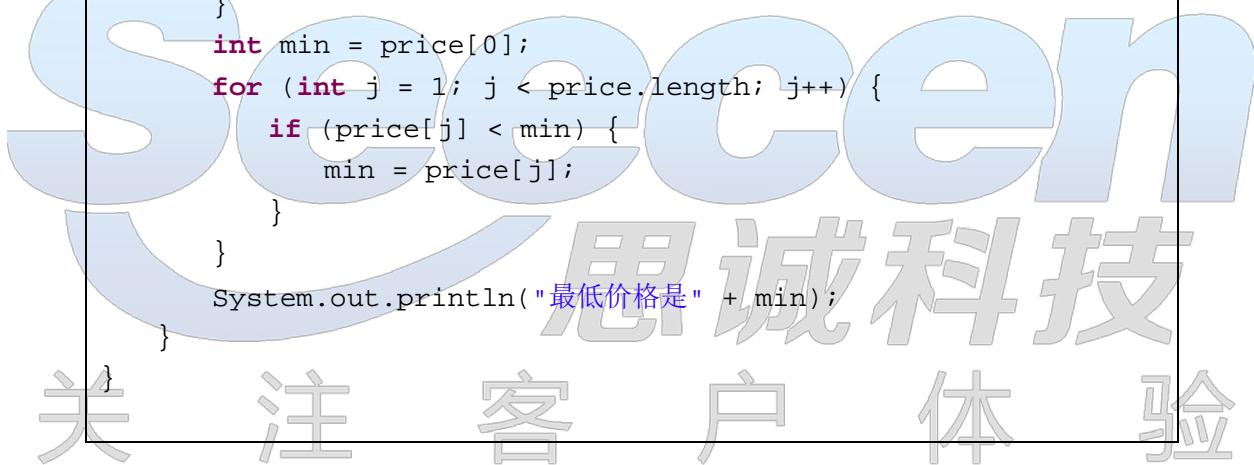
```
Public class ArraysDemo2 {
    public static void main(String[] args) {
        int a[] = { 1, 3, 5, 7, 9, 11, 13 };
        int b[] = { 2, 4, 6, 8, 10 };
        System.arraycopy(a, 1, b, 2, 3);
        for (int j = 0; j < b.length; j++) {
            System.out.println(b[j]);
        }
    }
}
```

```
// 运行结果 2 4 3 5 7
}
```

8.2.13.11 案例：求最低价格：

```
import java.util.Scanner;

public class Demo {
    public static void main(String[] args) {
        int[] price = new int[3];
        System.out.println("请输入4家店的价格");
        Scanner input = new Scanner(System.in);
        for (int i = 0; i < price.length; i++) {
            System.out.println("第" + (i + 1) + "个价格是");
            price[i] = input.nextInt();
        }
        int min = price[0];
        for (int j = 1; j < price.length; j++) {
            if (price[j] < min) {
                min = price[j];
            }
        }
        System.out.println("最低价格是" + min);
    }
}
```



8.2.13.12 二维数组

二维数组是数组的数组。

基本的定义方式有两种形式，如：

```
type[][] i = new type[2][3]; (推荐)
type i[][] = new type[2][3];
```

二维数组的每个元素都是一个一维数组，这些数组不一定都是等长的。

声明二维数组的时候可以只指定第一维大小，空缺出第二维大小，之后再指定不同长度的数组。但是注意，**第一维大小不能空缺（不能只指定列数不指定行数）**。

如：

```
//二维变长数组
int[][] a = new int[3][];
a[0] = new int[2];
a[1] = new int[3];
a[2] = new int[1];
//Error: 不能空缺第一维大小
//int[][] b = new int[][3];
```

二维数组也可以在定义的时候初始化，使用花括号的嵌套完成，这时候不指定两个维数的大小，并且根据初始化值的个数不同，可以生成不同长度的数组元素。

案例：静态初始化

```
package com.seecen;

public class Demo {
    public static void main(String[] args) {
        int a[][] = { { 1, 2, 3, 4, 5 }, { 6, 7, 8 } };
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[i].length; j++) {
                System.out.println("a[" + i + "]" + "[" + j + "]=" +
a[i][j]);
            }
        }
    }
}
//运行结果:
a[0][0]=1
a[0][1]=2
a[0][2]=3
a[0][3]=4
a[0][4]=5
a[1][0]=6
a[1][1]=7
a[1][2]=8
```

8.2.13.13 案例：二维数组动态初始化

```
package com.seecen;
//package com.seecen;
Public class Demo {
    public static void main(String[] args) {
        int b[][] = new int[3][4];
        for (int i = 0; i < 3; i++) {
```

```
        for (int j = 1; j < 4; j++) {
            b[i][j] = i * j;
            System.out.println("a[" + i + "][" + j + "]=" + i + "*" +
+ j
                + "=" + b[i][j]);
        }
    }
}

// 运行结果:
a[0][1]=0*1=0
a[0][2]=0*2=0
a[0][3]=0*3=0
a[1][1]=1*1=1
a[1][2]=1*2=2
a[1][3]=1*3=3
a[2][1]=2*1=2
a[2][2]=2*2=4
a[2][3]=2*3=6
```

8.2.13.14 常见的错误

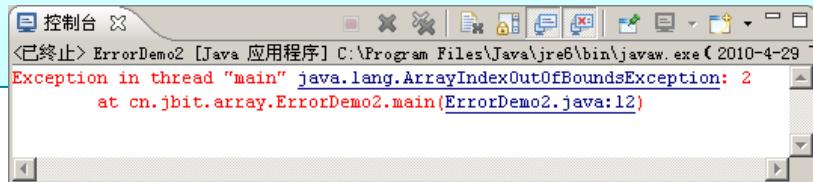
```
public class ErrorDemo2 {

    public static void main(String[] args) {

        int[] scores = new int[2];
        scores[0] = 90;
        scores[1] = 85;
        scores[2] = 65;

        System.out.println(scores[2]);
    }
}
```

编译出错，数组越界



8.2.13.15 本节作业

- 1、终端输入不同的水果后，水果重新排序
- 2、终端输入姓名后取出姓
- 3、终端输入字符串以及需要找的字符后，得出需要找的那个字符在字符串中出现的
- 4、取出学生身高最高的那个学生

8.2.14 常用集合接口

8.2.14.1 Set 接口：

Set 接口继承于 Collection 接口，而且它不允许集合中存在重复项，每个具体的 Set 实现类依赖添加的对象的 equals()方法来检查唯一性。并且最多包含一个 null 元素。Set 接口没有引入新方法，所以 Set 就是一个 Collection。Set 集合中的元素是没有顺序的，所以无法通过下标进行取值，须通过迭代的方式进行获取。

特征：无序且不可重复。

HashSet: 为快速查找设计的 Set。存入 HashSet 的对象必须定义 hashCode()。
TreeSet: 保存次序的 Set，底层为树结构。使用它可以从 Set 中提取有序的序列。
LinkedHashSet: 具有 HashSet 的查询速度，且内部使用链表维护元素的顺序(插入的次序)。于是在使用迭代器遍历 Set 时，结果会按元素插入的次序显示。

案例：

```
package com.seecen;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class Demo1 {
    public static void main(String[] args) {
        Set set = new HashSet();
        set.add(1);
        set.add(4);
        set.add(5);
        set.add(2);
        set.add(3);
        set.add(1);
        set.add(2);
        System.out.println("集合的大小是: "+set.size());
        Iterator its = set.iterator();
        while(its.hasNext()){
            System.out.print(its.next()+"\t");
        }
    }
}
```

```
}
```

```
}
```

```
}
```

结果：

集合的大小是：5

1 2 3 4 5

8.2.14.2 List 接口：

List 接口同样也继承于 Collection 接口，但是与 Set 接口恰恰相反，List 接口的集合类中的元素是对象有序且可重复。List 是基本的位置性集合，将元素加入 List 时，可以加入 List 的特定位置或者加到末尾。

特征：有序且可重复。

两个重要的实现类：ArrayList 和 LinkedList

1.ArrayList 特点是有序可重复的，由数组实现的 List。允许对元素进行快速随机访问，但是向 List 中间插入与移除元素的速度很慢。ListIterator 只应该用来由后向前遍历 ArrayList，而不是用来插入和移除元素。因为那比 LinkedList 开销要大很多。

2.LinkedList 是一个双向链表结构的。对顺序访问进行了优化，向 List 中间插入与删除的开销并不大。随机访问则相对较慢。(使用 ArrayList 代替。)还具有下列方法：addFirst(), addLast(), getFirst(), getLast(), removeFirst() 和 removeLast()，这些方法(没有在任何接口或基类中定义过)使得 LinkedList 可以当作堆栈、队列和双向队列使用。

案例：

```
package com.seecen;
import java.util.ArrayList;
import java.util.List;

public class Demo1 {
    public static void main(String[] args) {
        List list = newArrayList();
        list.add(12);
        list.add(12);
        list.add(13);
        list.add(15);
        list.add(16);
        list.add(12);
        list.add(13);
        for(int i=0;i<list.size();i++){
            System.out.println(list.get(i));
        }
    }
}
```

结果：

12
12
13
15
16
12
13

8.2.14.3 Map 接口：

Map 是一种把键对象和值对象映射的集合，它的每一个元素都包含一对键对象和值对象。Map 没有继承于 Collection 接口，Map 提供 key 到 value 的映射。一个 Map 中不能包含相同的 key，每个 key 只能映射一个 value。Map 接口提供 3 种集合的视图，Map 的内容可以被当作一组 key 集合，一组 value 集合，或者一组 key-value 映射。

Map 接口常用实现类：

HashMap: Map 基于散列表的实现。插入和查询“键值对”的开销是固定的。可以通过构造器设置容量 capacity 和负载因子 load factor，以调整容器的性能。

HashMap 和 Hashtable 类似，不同之处在于 HashMap 是非同步的，并且允许 null，即 null value 和 null key。但是将 HashMap 视为 Collection 时 (values()方法可返回 Collection)，其迭代子操作时间开销和 HashMap 的容量成比例。JDK1.2 之后加入的，性能高于 Hashtable。

Hashtable: Hashtable 继承 Map 接口，实现一个 key-value 映射的哈希表。任何非空 (non-null) 的对象都可作为 key 或者 value。添加数据使用 put(key, value)，取出数据使用 get(key)，这两个基本操作的时间开销为常数。Hashtable JDK1.0 就存在的，是线程安全、同步的，性能较低。

LinkedHashMap: 类似于 HashMap，但是迭代遍历它时，取得“键值对”的顺序是其插入次序，或者是最近最少使用(LRU)的次序。只比 HashMap 慢一点。而在迭代访问时反而更快，因为它使用链表维护内部次序。

TreeMap : 基于红黑树数据结构的实现。查看“键”或“键值对”时，它们会被排序(次序由 Comparable 或 Comparator 决定)。TreeMap 的特点在于，你得到的结果是经过排序的。TreeMap 是唯一的带有 subMap() 方法的 Map，它可以返回一个子树。

WeakHashMap : 弱键(weak key)Map，Map 中使用的对象也被允许释放：这是为解决特殊问题设计的。如果没有 map 之外的引用指向某个“键”，则此“键”可以被垃圾收集器回收。

案例：

```
package com.seecen;

import java.util.*;

public class Demo {
    public static void main(String[] args) {
        HashMap hm = new HashMap();
        hm.put("John Doe", new Double(3434.34));
        hm.put("TomSmith", new Double(123.22));
        hm.put("JaneBaker", new Double(1378.00));
        hm.put("ToddHall", new Double(99.22));
        hm.put("RalphSmith", new Double(-19.08));
        Set set = hm.entrySet();
        Iterator i = set.iterator();
        while (i.hasNext()) {
            Map.Entry me = (Map.Entry) i.next();
            System.out.print(me.getKey() + ":");
            System.out.println(me.getValue());
        }
    }
}
```

效果：

```
John Doe:3434.34
TomSmith:123.22
JaneBaker:1378.0
RalphSmith:-19.08
ToddHall:99.22
```

