第 3 章

JavaScript 客户端编程

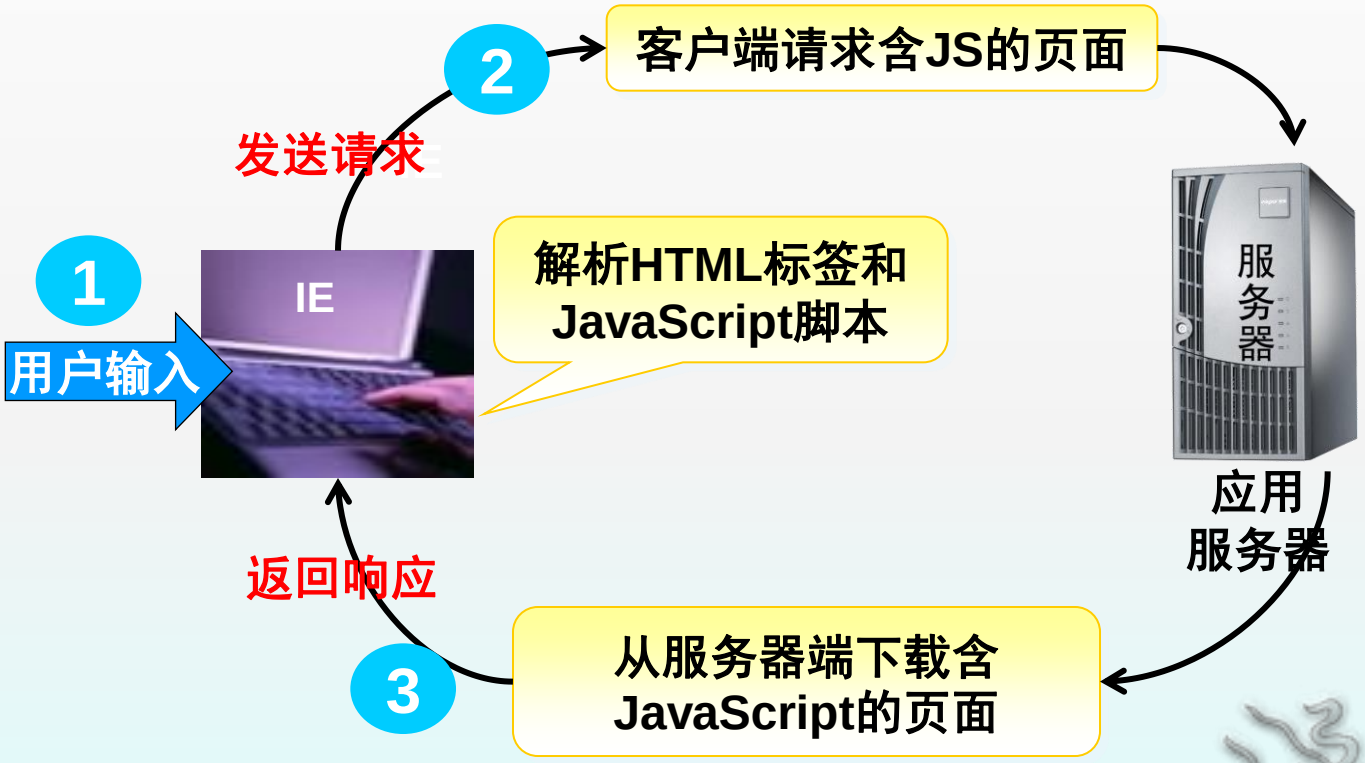
目 录

3.1 JavaScript概 述

3.1.1 什么是JavaScript

- JavaScript是一种基于对象的脚本语言，主要用于开发Web应用程序的客户端部分。
- JavaScript代码由浏览器解释执行，实现一些数据验证、页面动态显示等效果。
- JavaScript由Netscape公司开发，在1995年配合其浏览器推出，与Java并无本质联系，但其语法与Java非常相近。

脚本的执行原理


2
客户端请求含JS的页面
发送请求
1
用户输入
IE
解析HTML标签和
JavaScript脚本
服务器
应用
服务器
返回响应
3
从服务器端下载含
JavaScript的页面

JavaScript可以做什么

- 控制文档的外观和内容：通过Document对象的write方法可以在浏览器解析文档时将HTML写入文档中。
- 验证表单输入内容：在客户端验证表单中的输入，避免向服务器提交非法数据，节约服务器资源。
- 客户端计算和处理：从表单中读取输入数据并进行计算。
- 设置和检索cookie：将用户名、账号等信息持久地保存于Cookie中，在用户下一次访问网站时自动地读取这些信息。
- 捕捉用户事件并相应地调整页面：根据键盘或鼠标的动作，使页面的某一部分变得可编辑。
- 与服务器端应用程序进行交互：这是AJAX技术的基础，可在后台获取服务器端数据，并更新局部页面。

3.1.2 在网页中嵌入JavaScript脚本

示例：

```
<html>
<head>
<title>脚本的基本结构</title>
<script type="text/javascript">
    var count=0;
    document.write(“欢迎进入javascript!");
    for ( i=0; i<5; i++ )
        document.write("<h2>欢迎进入javascript!
</h2>");
</script>
</head>
<body>
<h1> BODY部分的内容</h1>
</body>
</html>
```

如何使用JavaScript实现此部分内容？

引入JavaScript脚本的方式有3种：

- 使用<script>标记在HTML文档中直接嵌入脚本
- 在HTML文档中链接JavaScript源文件
- 在HTML标记内添加JavaScript代码

方式1：在HTML文档中直接嵌入脚本

```
<html>
<head>
    <script type="text/javascript" >
        ..............
    </script>
</head>
</html>
```

方式2：在HTML文档中链接JavaScript源文件

```
<script type="text/javascript" src="文件名.js"> </script>
```

示例：03-02.html

```
<html>
    <head>
    <script src="test01.js" type="text/javascript"></script>
    </head>
    <body>
    </body>
</html>
```

示例：test01.js

```
document.write ("Hello, world!") ;
alert ("Hello, world!");
```

方式3：在HTML标记中嵌入JavaScript脚本

示例：03-02.html

```
<html>
<head> <title>在标记内添加脚本测试</title> </head>
<body>
    <button onClick="alert('这是标记内的脚本！ ')">
   在标记内添加脚本测试</button>
</body>
</html>
```

3.1.3 使用JavaScript输入与输出信息

在JavaScript中，常用的字符串输入输出方法有document对象的write()方法、window对象的alert()方法、文本框及消息输入框等。消息框包括确认框、提示框等。

```
document. write(“待输出的字符串”);  //向页面输出文本
```

```
window.alert (“待输出的字符串”);        //弹出带“确定”按钮的对话框
confirm (“待输出的字符串”);              //确认框
```

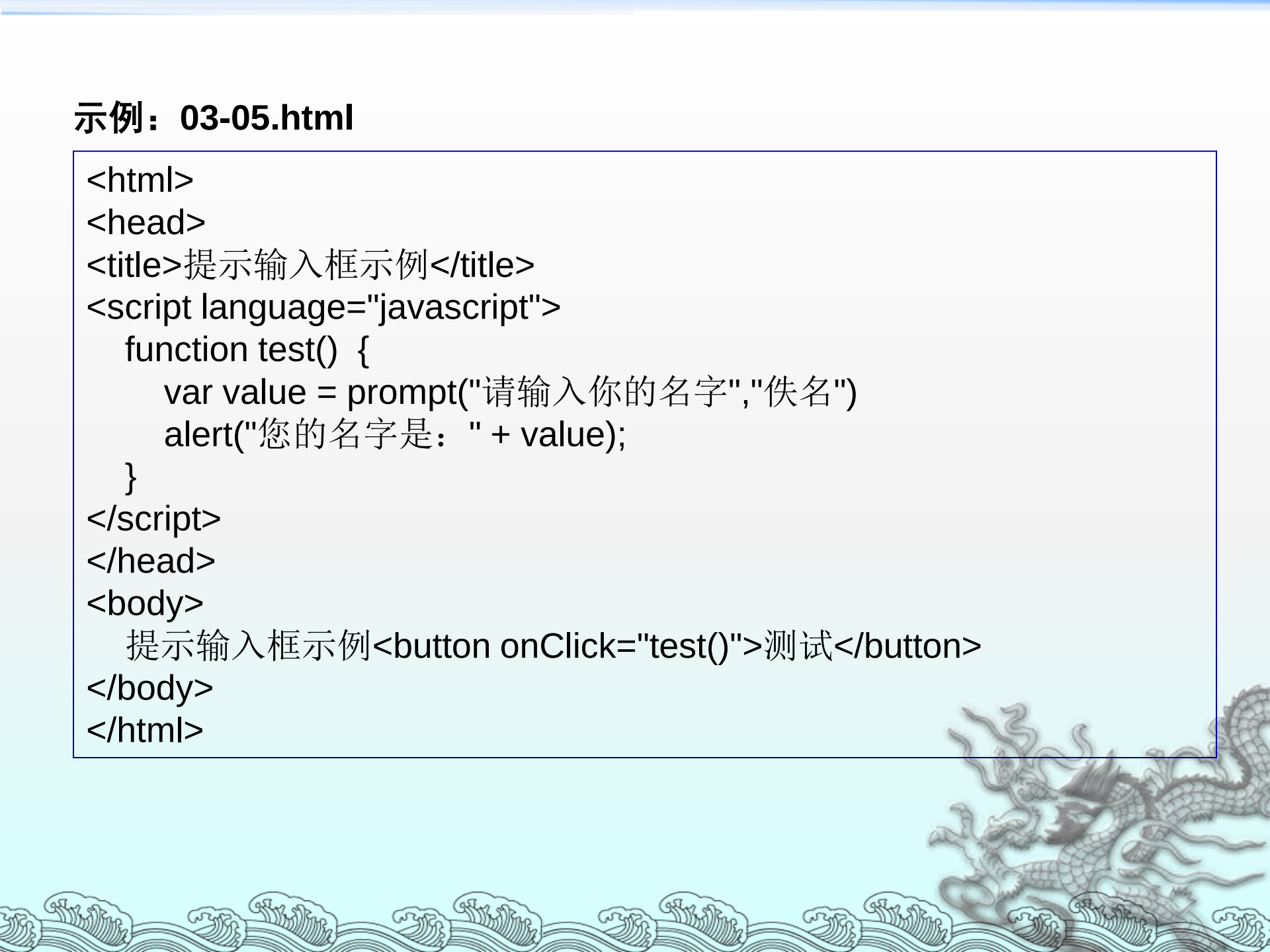
```
prompt("提示文本","默认值")          //提示输入框
```

示例：03-04.html

```
<html>
<head> <title>确认框示例</title>
 <script language="javascript">
      function test()
      {
           var value = confirm("确定要执行该操作吗？");
           alert("你的选择是："+value);
      }
 </script>
</head>
<body>
    确认框示例<button onClick="test()">测试</button>
</body>
</html>
```

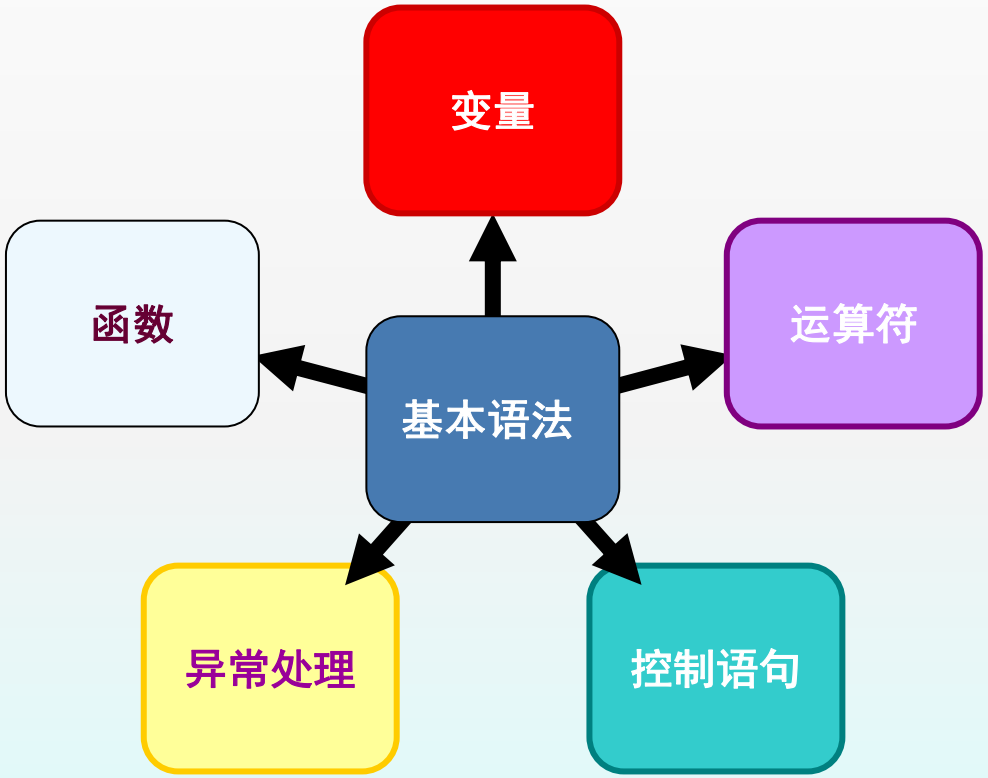
示例：03-05.html

```
<html>
<head>
<title>提示输入框示例</title>
<script language="javascript">
    function test()  {
        var value = prompt("请输入你的名字","佚名")
        alert("您的名字是：" + value);
    }
</script>
</head>
<body>
    提示输入框示例<button onClick="test()">测试</button>
</body>
</html>
```



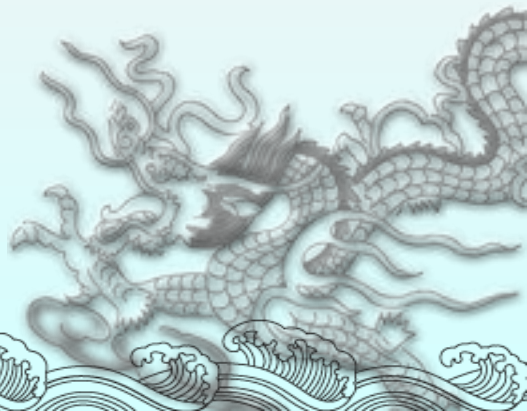
3.2 JavaScript基本语法

基本语法示意



3.2.1 数据类型

- string（字符串）类型：是用单引号或双引号括起来的一个或几个字符。
- number（数值）类型：可以是整数和浮点数。
- boolean（布尔）类型：值为true或者false。
- object（对象）类型：用于定义对象。



3.2.2 变量

定义变量

```
var   count;
```

赋值

```
count = 5;
```

同时声明和赋值变量

```
var count = 10;
```

声明多个变量

```
var x, y, z = 10;
```

- 变量名通常以字母，也可以以下划线或者$符号开头
- 变量名大小写敏感，即"x"和"X"为不同的变量

3.2.3 运算符和表达式

- 运算符对一个或多个变量或值（操作数）进行运算，并返回一个新值
- 根据所执行的运算，运算符可分为以下类别
  - **字符串运算符：** +
  - **算术运算符：** +、-、 * 、 / 、%、++、--、-(求反)
  - **比较运算符：** ==、!=、>、>=、<、<=
  - **逻辑运算符：** &&、||、！
  - **位运算符：** &、|、~、^、<<、>>、>>>
  - **赋值运算符：** =、+=、-=、*=、/=、%= 等

3.2.4 流程控制

- if条件语句
- switch多分支

```
if(条件)
{
  //JavaScript代码;
}
else
{
 //JavaScr
}
```

```
switch (表达式)
{
        case 常量1 :
          JavaScript语句1;
          break;
        case 常量2 :
          JavaScript语句2;
          break;
        ...
          default :
          JavaScript语句3;
}
```

示例：03-06.html

```
<html>
<head><title> switch语句示例 </title>
 < script type="text/javascript">
    var now  = new Date();    var date =  now.getDay();
    switch (date)
    {
        case 1:   alert("今天是星期一");    break;
        case 2:   alert("今天是星期二");    break;
        case 3:   alert("今天是星期三");    break;
        case 4:   alert("今天是星期四");    break;
        case 5:   alert("今天是星期五");    break;
        case 6:   alert("今天是星期六");    break;
        default:   alert("今天是星期日");
    }
 </script>
</head>
<body></body>
</html>
```

for、while循环语句

```
for(初始化；条件；增量)
{
    语句集;
 }

while(条件)
{
    语句集;
}
```

示例：03-07.html

```
<html>
<head>
<title>for循环语句打印乘法口诀</title>
<script type="text/javascript" >
    for (var i = 1 ; i <= 9 ; i++) {
        for (var j = 1 ; j <= 9 ; j++) {
            if (j<=i) {  //只打印下三角
             document.write ("\t" + j + "*" + I + "=" + (i*j) );
            }
        }
        document.write ("<br/>");   //换行
    }
</script>
</head>
<body></body>
</html>
```

◈ Break 语句

- 出现在循环语句或switch语句内，用于强行跳出循环或switch语句。
- 在嵌套循环中，break语句只跳出当前循环体，并不跳出整个嵌套循环。

◈ Continue语句

- 用在循环结构中，作用是跳过循环体内剩余的语句而提前进入下一次循环。

示例：

```
sum =0；
i=0；
while（true） {
    i++；
    if（i>100）break；
    if（i%2 = =0）continue；
    sum = sum +i；
}
```

5. 异常处理语句

- 抛出异常：throw
- 捕获异常：try
- 处理异常：catch
- 清理现场：finally

```
<script  type = “text / javascript” >
<!--
   var n1 = 7 ;
   var n2 = 0;
   try{
      if ( n2==0 )    throw new Error(“除0”);
      document. write( n1/n2 );
   } catch ( err ) {
      document. write( “出错” + err. message);
   } finally {
      delete n1;
      delete n2;
      document. write (“<br />操作完成” );
   }
-- >
</script>
```

3.2.5 函数

- **函数**是已命名的代码块，其中的语句作为一个整体被调用执行，且只有在调用时才会执行。
- 函数定义通常放在HTML文档的<head>块中，也可以放在其他位置，但要确保先定义后使用。
- 函数可以由事件触发调用，也可以由脚本中的代码调用

示例：按用户输入的次数显示问候语

◈ 创建函数

```
function showHello( )
{
   var count=document.myForm.txtCount.value ;
    for(i=0; i<count; i++)
       document.write("<H2>HelloWorld</H2>");
}
```

◈ 调用函数

函数调用一般[illegible]使用，格式为：

事件名＝“函数名[illegible]”；

表示单击此按钮时，调用函数showHello( )执行

```
<INPUT type="submit" name="Submit" value="显示HelloWorld"
onClick="showHello( )">
```

说明：

- 函数名是调用函数时所引用的名称，在同一个脚本文件里函数名必须唯一。
- 形式参数表用以接收传入数据，在调用函数时，其实参的个数和类型必须与形参相一致。
- 大括号中是函数的执行语句，如果要返回一个值，则应该在最后一行使用return语句。

函数调用方式

- 语句调用

```
<script type="text/javascript">
    function add(a,b){
        return (a+b);
    }
    var result = add(2,3);
    alert( "a与b两数之和为 "+ result);
</script>
```

- 事件调用

```
<html>
<head><title> javascript函数的事件调用</title>
<script type="text/javascript">
      function showmessage( ) {
            alert("这是JavaScript事件调用函数");
      }
</script>
</head>
<body>
      <input type="button" value="鼠标单击事件调用函数"
      onClick="showmessage ()" />
</body>
</html>
```

变量的作用域

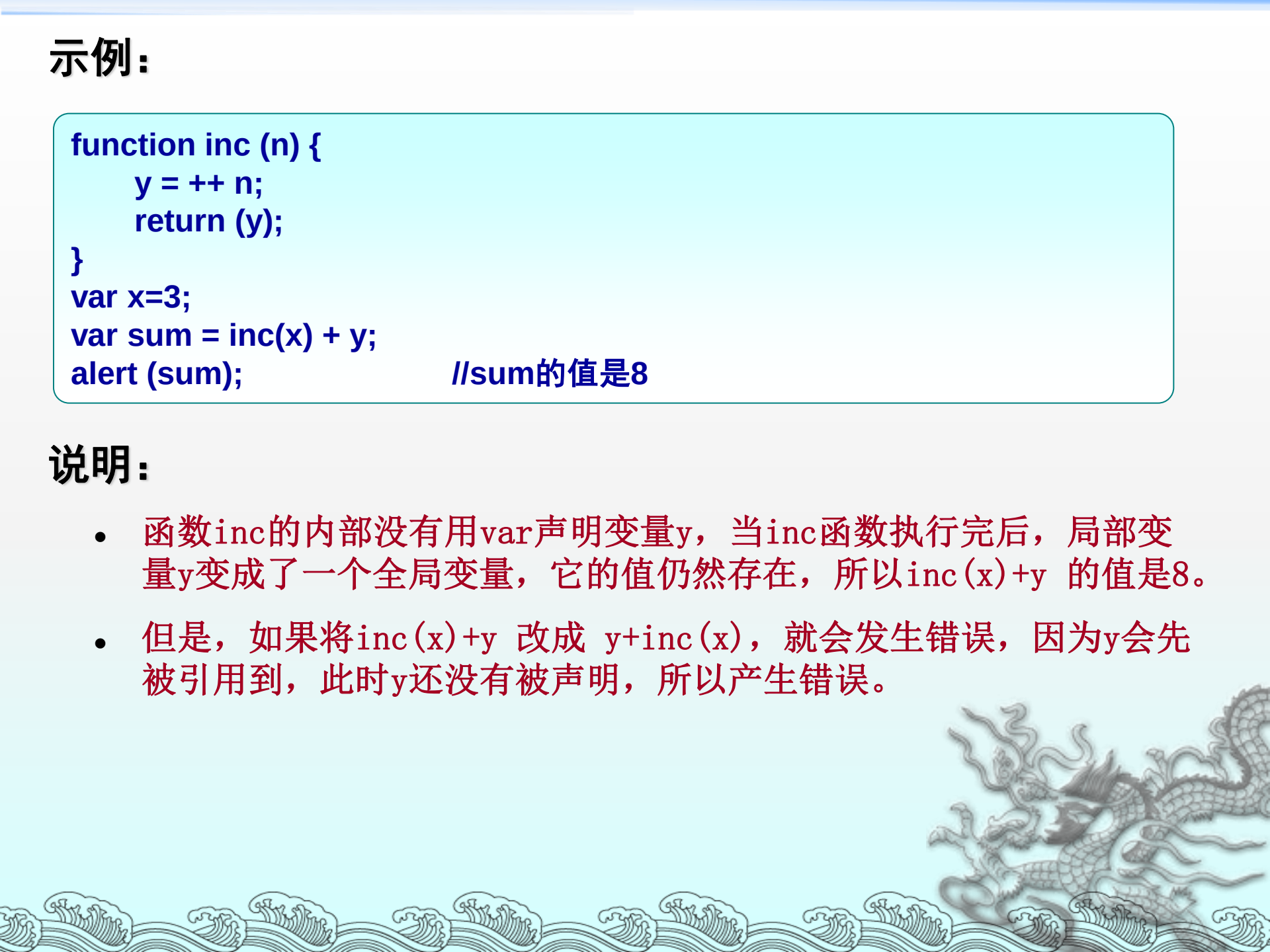
- 在函数之外定义的变量为全局变量，可在各个函数之间共享。
- 在函数内部使用var声明的变量为局部变量，只在当前函数内部有效
- 但那些在函数内部没有用var声明的变量，在赋值后也会被当做全局变量使用。

示例：

```
function inc (n) {
      y = ++ n;
      return (y);
}
var x=3;
var sum = inc(x) + y;
alert (sum);                    //sum的值是8
```

说明：

- 函数inc的内部没有用var声明变量y，当inc函数执行完后，局部变量y变成了一个全局变量，它的值仍然存在，所以inc(x)+y 的值是8。
- 但是，如果将inc(x)+y 改成 y+inc(x)，就会发生错误，因为y会先被引用到，此时y还没有被声明，所以产生错误。



3.2.6 事件处理（Event Driven）

- 在客户端脚本中，**JavaScript通过事件响应来获得与用户的交互。**

- **Javascript标准事件：**
  - （1）onload 和 onUnload事件
  - （2）onClick事件
  - （3）onFocus、onBlur和onChange事件
  - （4）onMouseOver 和 onMouseOut事件
  - （5）onSubmit事件

示例：

```
<body onLoad = "alert ( ‘Welcome to JavaScript world!’ ); " >
      // 页面代码
</body>
```

```
<button name="button1" onClick = "btn1Click();"> click me</button>
```

```
<input type="text" size="30" id="email" onchange="checkEmail()" />
```

```
<a href="#" onmouseover="alert ('onMouseOver event'); return false">
Click Me</a>
```

```
<form method="post" action="xx.aspx" onsubmit="return checkForm()" >
    // 表单内容
</form>
```

3.3 JavaScript对象编程

1、什么是对象

- JavaScript是一种基于对象的语言（Object-Based），而不是面向对象的语言（object-oriented），因为它没有提供面向对象语言的核心特性：抽象、继承、重载。
- JavaScript使用的对象可以分为三类：
  - ✓ 内置对象：由JavaScript语言本身提供
  - ✓ 宿主对象：由JavaScript的运行环境（浏览器）提供
  - ✓ 用户定义对象：用户可以创建自己的对象，或创建自己的类并实例化对象。

2、有关对象操作的语句

- New关键词：创建新的对象

  格式：var obj = new ObjectType(Parameters)

  例如：var d = new Date(“2018-2-1”);

  var today = new Date();

- With语句：设置代码运行的对象

  格式：

  ```
  with object {
      ..........
  }
  ```

  说明：任何对变量的引用被认为是这个对象的属性。

续：有关对象操作的语句

- for … in 语句：

  格式：For（属性名 in 对象名） {

  ……

  }

  含义：用于遍历数组或者对象的属性（对数组或者对象的属性进行循环操作）。该语句的优点是无需知道数组的元素个数或对象的属性个数，就可以安全的遍历。

  例如：
  ```
  for(var prop in object)
      document.write(object[prop]);
  ```

3.3.1 常用JavaScript对象

1、String对象：用于操纵和处理文本字符串

- 声明一个String对象： var str = “Hello”;

  或：var str = new String(“Hello”) ;

- 只有一个属性：
  - ✓ Length：返回字符串的长度
- 方法：
  - ✓ toLowerCase() ：转换成小写
  - ✓ toUpperCase() ：转换成大写
  - ✓ indexOf(charactor, fromIndex)：字符串查找
  - ✓ lastIndexOf(charactor)：查找最后一次出现的子串
  - ✓ substring(start, end)：字符串截取
  - ✓ charAt(pos)：返回串中指定位置的一个字符
  - ✓ oncat(str1, str2)：字符串拼接
  - ✓ replace()、split（）等

示例：

```
var msg = "Hello" + "World" ;      // "+"用于字符串，可以实现字符串拼接
var msg = concat (“Hello”, “World”);    // 和上句等效
document.writeln ( msg.length );          // 输出字符串的长度10
var idx = msg.indexOf ("World");         // 检索子串出现的位置，这里为5
// 截取从第5个字符往后到第10个字符之间的所有字符（“World”）
document.writeln( msg.substring(idx,10) );
document.writeln (msg.toUpperCase());   // 输出为"HELLOWORLD"
```

2、Array对象

- 可以在单个变量中存储多个值

  格式：var myArr = new Array();

  var myArr = new Array(3);

  var myArr = new Array(“January”, “February”);

  var myArr = [“January”, “February”] ;

- 访问数组元素

  元素赋值：myArr[0] = “January”;

  引用元素：document.write(myArr) ;

  document.write(myArr[0], myArr[1]) ;

- length属性：用于设置或返回数组中的元素个数
- 方法：concat() 、sort()等

示例：

```
var mycars = new Array();      // 创建数组对象，可以不指定元素个数
mycars[0] = "BMW"; mycars[1] = "AUDI";          // 为数组元素赋值
var yourcars = new Array(3);          // 创建数组对象并指定元素个数
var hiscars = new Array ("Buick", "Benz");  // 创建数组对象并同时赋值
for (x in mycars) {                    // 遍历数组元素，x能得到元素下标
    document.write(mycars[x] + "<br />")     // 输出数组元素
}
```

```
var arr1 = new Array ("Tom", "Jerry")
var arr2 = new Array ("Bingo")
var arr = arr1.concat(arr2);          // 连接两个数组，生成新的数组
document.write (arr.join(" | "));    // 将arr中的元素拼接成字符串，使用“|”分隔
arr = new Array(7, 5, 3, 8, 6);
document.write (arr.sort());          // 对数组元素进行排序
```

3、Date对象：提供有关日期、时间的操作

- 创建：

```
Var MyDate = New Date();
Var MyDate = New Date(2017, 9, 1);
```

- 方法：
  - ✓ 获取日期和时间：

    getYear( )、getMonth( )、getDate( )、getDay( )、getHours( )……
  - ✓ 设置日期和时间：

    setYear( )、setMonth( )、setDate( )、setDay( )、setHours( )……

注意:

Date对象的月份可接收的数值为0-11，代表1-12月。

示例：页面时钟

```
<html>
<head>
<script type="text/javascript">
    function startTime() {
        var d = new Date();          var h = d.getHours();
        var m = d.getMinutes();      var s = d.getSeconds();
        m = checkTime(m);            s = checkTime(s);
        document.getElementById ('clock'). innerHTML
            = "本地时间： " + h + ":" + m + ":" + s;
        setTimeout( 'startTime()' , 500);
    }
    function checkTime(i) {    if (i<10)    i = "0" + i;    return i    }
</script>
</head>
<body onload="startTime()">  <div id="clock" />  </body>
</html>
```

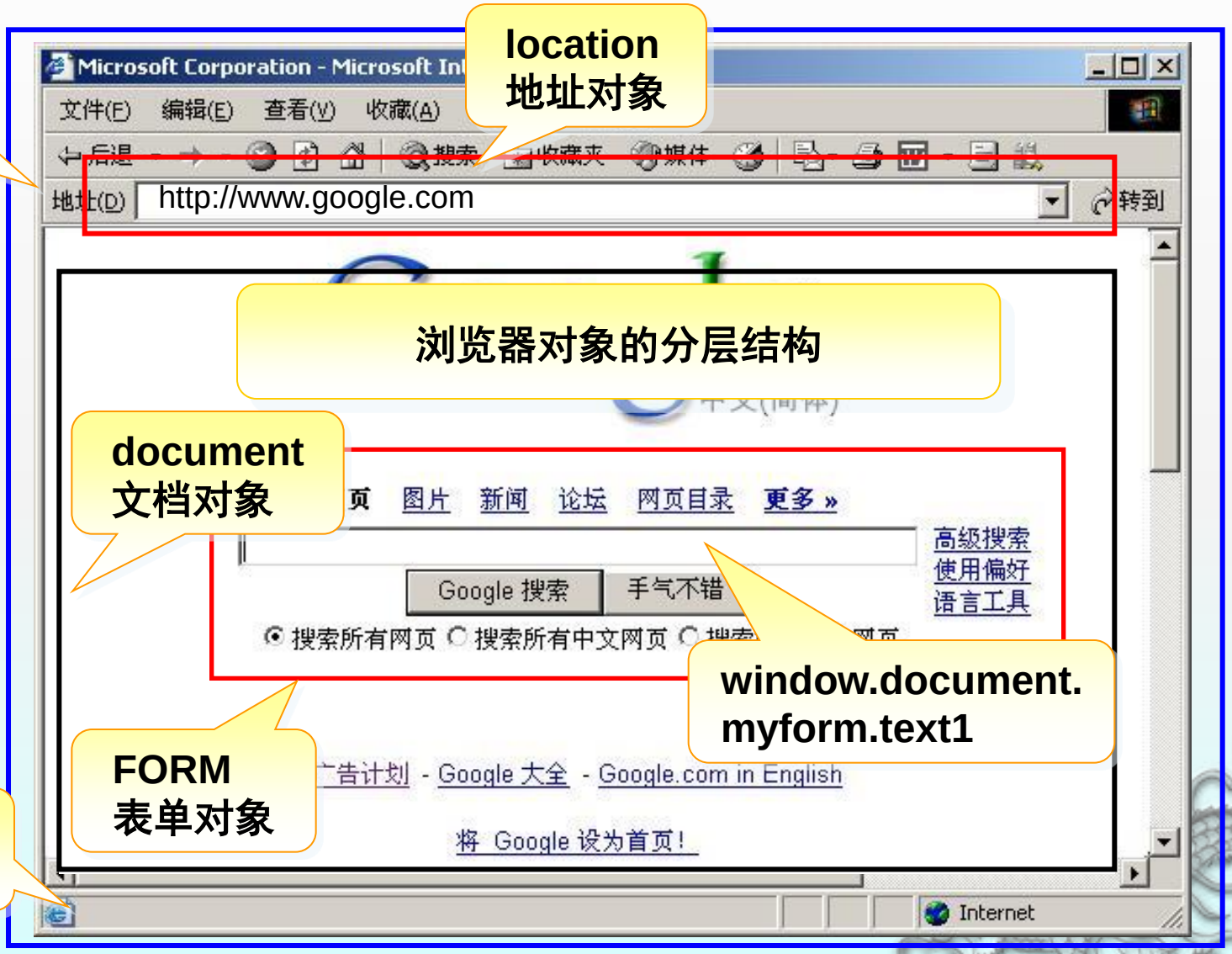
4、Math对象：提供一些数学常数以及运算函数

- 创建：直接使用，不需创建
- 属性：
  - ✓PI、E、LN10、LN2、SQRT1-2 、SQRT2
- 方法
  - ✓ 绝对值函数：abs()
  - ✓ 正弦余弦函数：sin(),cos()
  - ✓ 反正弦反余弦函数：asin(), acos()
  - ✓ 正切反正切函数：tan(),atan()
  - ✓ 四舍五入函数：round()
  - ✓ 求平方根函数：sqrt()
  - ✓ 基于几方次的值：Pow(base, exponent)

示例：使用Math对象

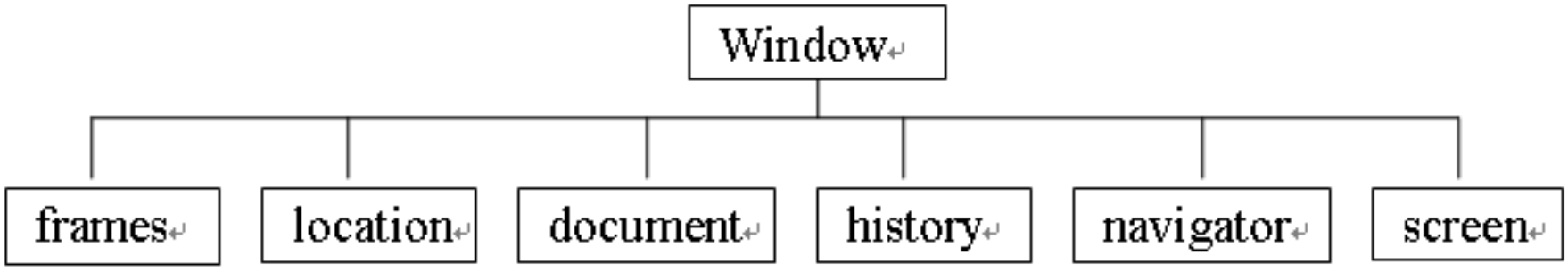
```
var pi_val = Math.PI;                    // 获得PI值
var sin_val = Math.sin(pi_val);          // 计算sin(PI)
var sqrt_val = Math.sqrt(5);             // 求5的平方根
var rnd_val = Math.round(sqrt_val);      // 对5的平方根取整
document.write ("PI=" + pi_val + "<br/>");
document.write ("sin(PI)=" + sin_val + "<br/>");
document.write ("sqrt(5)=" + sqrt_val + "<br/>");
document.write ("round(sqrt(5))=" + rnd_val + "<br/>");
```

3.3.2 浏览器对象模型


window
窗口对象
location
地址对象
浏览器对象的分层结构
document
文档对象
window.document.
myform.text1
FORM
表单对象
status
状态栏对象
Microsoft Corporation - Microsoft Int
文件(F) 编辑(E) 查看(V) 收藏(A)
后退
搜索
收藏夹
媒体
地址(D)
http://www.google.com
转到
图片 新闻 论坛 网页目录 更多 »
高级搜索
使用偏好
语言工具
Google 搜索
手气不错
搜索所有网页
搜索所有中文网页
广告计划 - Google 大全 - Google.com in English
将 Google 设为首页!
Internet

浏览器宿主对象模型

浏览器作为JavaScript的运行环境，提供了一系列的宿主对象；通过这些对象，JavaScript可以获取浏览器的信息，并控制浏览器执行指定的操作。.



浏览器宿主对象关系

1、Window对象：

Window 对象表示浏览器中打开的窗口。

Window作为顶层对象，在访问它的属性和方法时，无需指定对象名。

- Window对象的常用属性
  - ✓ **status：指定浏览器状态栏中显示的临时消息**
  - ✓ **screen： 有关客户端的屏幕和显示性能的信息**
  - ✓ **history：有关客户访问过的URL的信息**
  - ✓ **location：有关当前 URL 的信息**
  - ✓ **document：表示浏览器窗口中的HTML文档**
  - ✓ **Opener：代表使用open方法打开当前窗口的窗口**
  - ✓ **Self：代表当前窗口**
  - ✓ **Top：代表所有框架中的顶层窗口**
  - ✓ **Frames：集合对象，代表当前窗口中的框架集，从而可以获取并操纵所有的子窗口**

- Window对象的常用方法
  - ✓ alert (“提示信息”)：显示一个带有提示信息和确定按钮的对话框
  - ✓ confirm (“提示信息”)：显示一个带提示信息、确定和取消按钮的对话框
  - ✓ open (“url”, “name”)：打开具有指定名称的新窗口，并加载给定 URL 所指定的文档；如果没有提供 URL，则打开一个空白文档
  - ✓ close( )：关闭当前窗口
  - ✓ moveTo (ix，iy)：移动到指定位置
  - ✓ resizeTo (iwidth, iheight)：调整到指定大小
  - ✓ setTimeout (“函数名”，毫秒数)：经过指定毫秒后执行某函数
  - ✓ setInterval (“函数名”，毫秒数)：以指定的时间间隔反复执行某函数
  - ✓ clearInterval (intervalID)：清除定时设置
  - ✓ navigate (URL)：导航到新页面


```
<SCRIPT lang                    t">
function openw      dow( )
{  window.status="系统当前状态                户.....";
   if (window.screen.width == 10     && window.screen.height == 768)
      window.open("register.html");
   else
      window.alert("请设置分辨率为1024x768，然后再打开");
}
function closewindow( )
{  if(window.confirm("您确认要退出系统吗？"))
      window.close( );
}
</SCRIPT
<INPUT ty       utton" name="regButton" value=" 用户注册 "
          onclick="openwindow( )">
<INPUT type="button" name="exitButton" value=" 退 出 "
          onclick="closewindow( )">
```

在窗口状态栏中设置文本

使用open方法打开新窗口

设置窗口的高度

弹出警告对话框

弹出确认对话框

关闭当前窗口

添加单击事件

Open方法的一般用法：

open（”打开窗口的url”，”窗口名”，”窗口特征”）

窗口的特征如下，可以任意组合：

- ✓ height： 窗口高度；
- ✓ width： 窗口宽度；
- ✓ top： 窗口距离屏幕上方的象素值；
- ✓ left：窗口距离屏幕左侧的象素值；
- ✓ toolbar： 是否显示工具栏，yes为显示；
- ✓ menubar，scrollbars 表示菜单栏和滚动栏。
- ✓ resizable： 是否允许改变窗口大小，yes或1为允许
- ✓ location： 是否显示地址栏，yes或1为允许
- ✓ status：是否显示状态栏内的信息，yes或1为允许；

示例

```
<SCRIPT language="javascript">
functi                {
  wind                前状态：您正在注册用户......";
  if (wind   w.screen.width == 1024 && window.screen.height == 768)
    open("register.html", "注册窗口", "toolbars=0, location=0,
statusbars=0, menubars=0,width=700,height=550,scrollbars=1");
  else
       window.alert("请设置分辨率为1024x768，然后再打开");  }
function closewindow( )
{
   if(window.confirm("您确认要退出系统吗？"))
       window.close( );
}
</SCRI
<INPUT              tton" name="regButton" value=" 用户注册 "
onclick="openwindow( )">
     <INPUT type="button" name="exitButton" value=" 退 出 "
onclick="closewindow( )">
```

使用 Open 方法
打开注册新窗口

添加单
击事件

2、location对象：代表浏览器所打开的网页地址

- 常用属性
  - ✓ **href**：返回或设置页面的URL，可导航到新页面
  - ✓ **protocol**：返回或设置网址中的协议部分
  - ✓ **host**：返回或设置网址中的主机号
  - ✓ **port**：返回或设置网址中的端口号
  - ✓ **pathname**：返回或设置网址中的路径名
  - ✓ **search**：返回或设置网址中的请求字符串
- 方法
  - ✓ **reload( )**：重新加载页面，等同于工具栏上的刷新按钮
  - ✓ **assign()**：加载一个新的页面
  - ✓ **replace()**：打开另一个URL，替代当前网址

示例：使用location对象获取详细地址信息

```
document.write ("当前位置： " + location.href + "<br/>");
document.write ("主机名称： " + location.host + "<br/>");
document.write ("请求路径： " + location.pathname + "<br/>");
document.write ("主机端口： " + location.port + "<br/>");
document.write ("请求字符串： " + location.search + "<br/>");
```

示例：调用location对象的方法

```
location.assign (“http://localhost/login.aspx”);
location.replace (“http://localhost/index.html”);
location.reload ();
```

说明：由于这里操作的都是当前窗口的location对象，所以省略了窗口名；
若要访问其它窗口的位置，则应使用“窗口名.location”的格式

```
var newwin = window.open (“http://localhost/login.htm”, “_blank”) ;
document.write( newwin.location ) ;
```

3、history对象：浏览器的浏览历史

- 常用属性
  - ✓ **Length**：返回浏览历史中地址的总数
- 方法
  - ✓ **back( )**：后退，和工具栏上的后退按钮等效
  - ✓ **forward()**：前进，和工具栏上的前进按钮等效
  - ✓ **go(location)**：跳转到一个指定的地址

    若location为字符串，则代表浏览历史中的某URL

    若location为正数，则前进n次

    若location为负数，则后退n次

示例

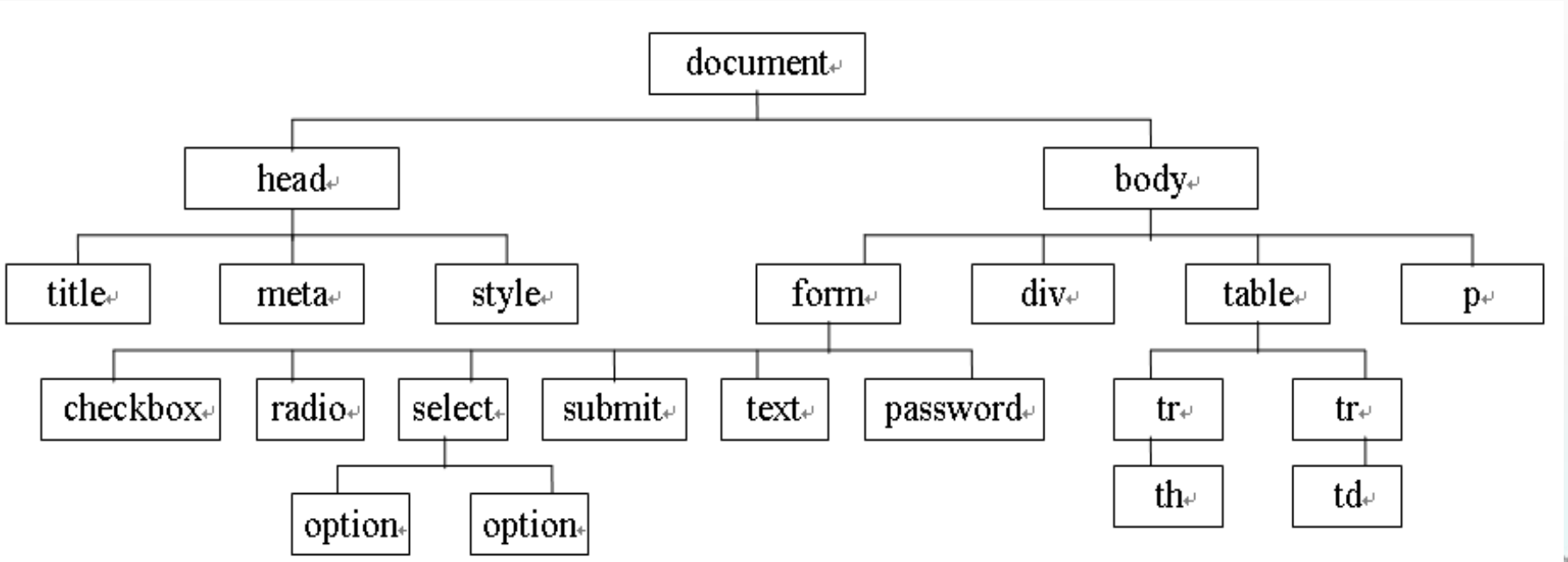
```
<FORM name="form1" method="post" action="">
……
<TD width="4%"><A href="javascript: history.back( )">返回 </A></TD>
<TD width="4%"><A href="javascript: history.forward( )">前进</A></TD>
<TD width="4%"><A href="javascript: location.reload( )"
<TD width="6%"><A href="../index.html">首页</A></TD>
跳转到其他版块
<SELECT name="selTopic"  id="selPTopic" onChange="javascript:
location=this.value">
    <OPTION value="news.html" selected="selected">新闻贴图</OPTION>
     <OPTION value="gard.html">网上谈兵</OPTION>
          value="IT.html">IT茶馆</OPTION>
          value="education.html" selected >教育大家谈</OPTION>
```

添加选项改变事件

获取被选中的下拉菜单项value的值

3.3.3 文档对象模型

HTML文档的逻辑结构：



用节点树的形式表述文档的逻辑结构

什么是DOM： Document Object Model 是W3C国际组织的一套Web标准，它定义了访问XML文档或HTML文档对象的一套属性、方法和事件 。

```
<html>
<head>
<script type="text/javascript"
function changeLink()
{ document.getElementById('myAnchor').innerHTML="搜狐" ;
document.getElementById('myAnchor').href="http://www.sohu.com" ; }
</script>
</head>
<body>
<a id="myAnchor" href="http://www.taobao.com">淘宝</a>
<input type="button" onclick="changeLink()" value="使用DOM改变链接">
</body>
</html>
```

示例

定位链接元素（对象）

修改内容

修改属性

HTML文档的每个节点都是对象，都具备属性、方法和事件

（1）使用DOM方法查找HTML元素

在新的DOM标准中，推荐使用对象ID或名称引用文档中的对象

- ✓ **document.getElementById ("objectID");**
- ✓ **document.getElementByName ("objectName");**

例如：

```
var aNode = document. getElementById (“mydiv”);
var nodeList = document. getElementsByName (sname) ;
```

（2）动态改变元素内容

当检索到某个元素后，可以通过其innerHTML属性获取或改变元素内容，例如：

```
<h1  id="header">Old Header</h1>
 <script>
 var element = document.getElementById("header");
element.innerHTML = "New Header";
 </script>
```

示例：使用Javascript动态改变超链接元素的显示文本和链接目标

```
<html>
<head>
<script type="text/javascript">
   function change()
   {
                  var ah = document.anchors[0];
                  ah.innerHTML = "新的链接";
                  ah.href = "http://mail.163.com";
          }
</script>
</head>
<body>
   <a href="#" >旧的链接</a>
   <input type="button" onclick="change()" value="测试"  />
</body>
</html>
```

（3）动态改变元素样式

如果要动态改变HTML元素的显示样式，可以使用如下语法：

document.getElementById(id).style.property = "属性值"

例如：

```
<p id="p2">Hello World!</p>
<script>
document.getElementById("p2").style.color="blue";
</script>
```

（4）对DOM事件作出响应

- 针对HTML元素可以捕获鼠标单击事件、鼠标移动事件、内容改变事件等一系列事件，并为每个事件编写处理函数。借助DOM技术，我们还可以将一个事件处理函数动态地分配给某个HTML元素的相应事件。

示例：

```
<html>
<head>
<script type="text/javascript">
   function test ( ) {
      var name = document.form1.tbxname.value;
      var gd = document.form1.rbngd[0].checked ? "男" : "女";
      alert("你的输入是：  " + name  + "\t" + gd);
   }
</script>
</head>
<body>
<form action="" name="form1">
    姓名：<input name="tbxname" type="text" /><br/>
    性别：<input name="rbngd" type="radio" />男
    <input name="rbngd" type="radio" />女
    <input type="button" name="btntest" onclick="test()" value="测试"  />
</form>
</body>
</html>
```

（5）动态添加或删除元素

向页面添加新元素的一般步骤是：

1 创建新元素

2 检索要将其加入的父元素

3 在父元素之中加入新元素

例　向页面中动态添加HTML元素

```
<html>
<body>
<div id="div1"> <p id="p1">这是一个段落。</p> </div>
<script>
    var para = document.createElement("p");
    var node = document.createTextNode("这是新段落");
    para.appendChild(node);
    var element=document.getElementById("div1");
    element.appendChild(para);
</script>
</body>
</html>
```

3.4 实例：使用JavaScript实现客户端数据验证

设计说明

用户在表单中输入数据后，点击提交按钮。这时，在数据提交之前，首先在客户端取得表单数据，并进行检测，只要有一项数据不合法，就放弃提交，并给用户以提示，只有所有表单数据都通过验证后才提交到服务器端进行处理。

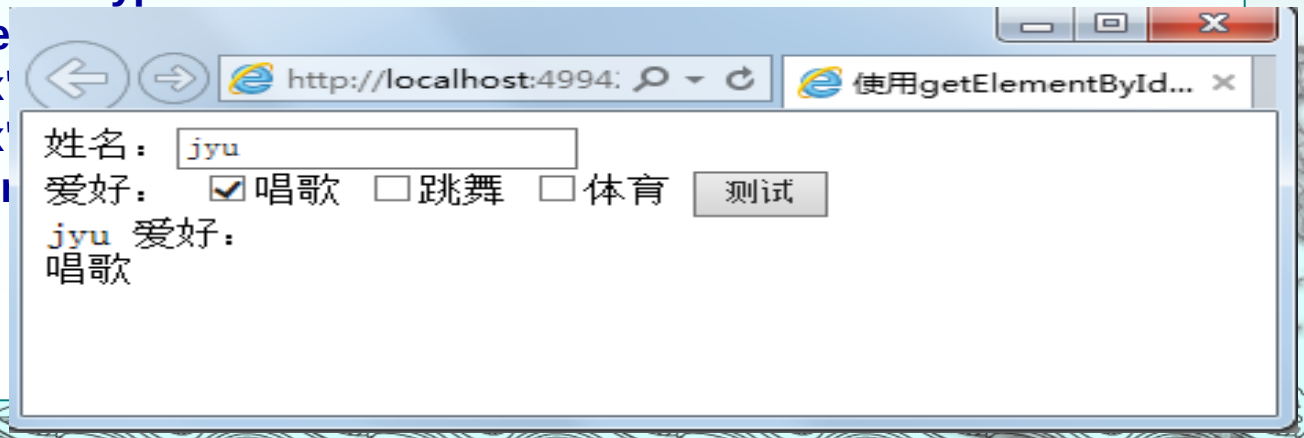
客户端处理的关键有如下两点：

（1）如何获取用户在表单中输入的数据

（2）如何对表单中的数据进行验证

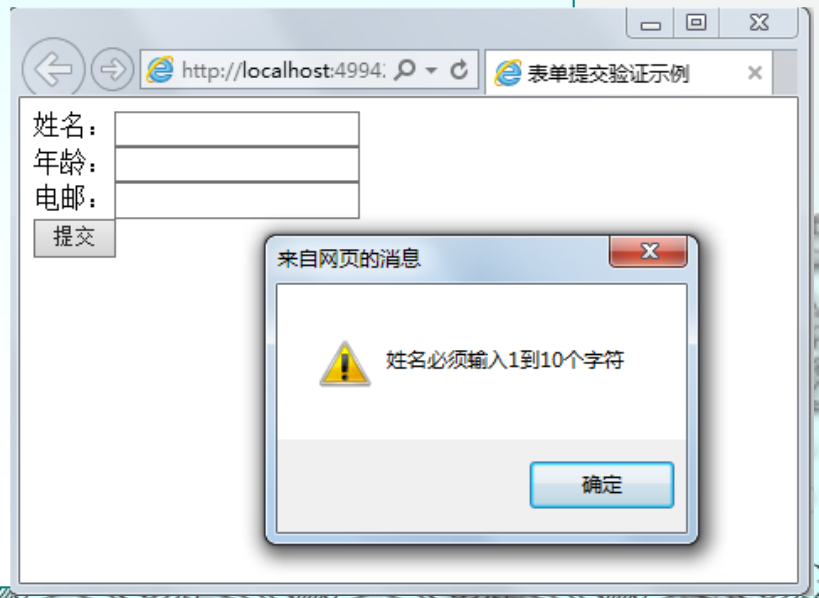
（1）通过DOM方法获取表单中输入的数据

```
<html>
<head>
<script type="text/javascript">
  function test()  {
    var name = document.getElementById ("tbxname"). value;
    var favs = document.getElementsByName ("cbxfav");
    var fstr = name + " 爱好：<br/>";
    For (var i=0; i<favs.length; i++ ){
      If (favs[i]. checked==true) {   fstr += favs[i].value;   fstr += "<br />";   }
    }
    document.getElementById ("divmsg"). innerHTML = fstr;
  }
</script>
</head>
<body>
姓名：<input id="tbxname" type="text" /><br/>
爱好：<input type="che
<input type="checkbox"
<input type="checkbox"
<input type="button" on
<div id="divmsg" />
</body>
</html>
```



（2）验证表单中输入的数据

```
<html>
<head><meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<script>
function validate() {
  var at=document.getElementById("email").value.indexOf("@") ;
  var age=document.getElementById("age").value ;
  var fname=document.getElementById("fname").value ;
  submitOK="true" ;
  if(fname.length==0||fname.length>10) {
                alert("姓名必须输入1到10个字符") ;
                submitOK="false" ;
  }
 else if (isNaN(age)||age<1||age>100) {
                alert("年龄必须是1与100之间的数字") ;
                submitOK="false" ;
  }
  else if (at==-1) {
                alert("不是有效的电子邮件地址") ;
                submitOK="false" ;
  }
  if (submitOK=="false") {
                return false ;    }
}
</script></head>
<body>
<form action="submitpage.html" onsubmit="return validate()">
姓名：<input type="text" id="fname" size="20"><br />
年龄：<input type="text" id="age" size="20"><br />
电邮：<input type="text" id="email" size="20"><br />
<input type="submit" value="提交">
</form></body>
</html>
```



作业3

- 1.用JavaScript脚本语言编写计算1+2+3+...+100的程序，并输出结果。
- 2.输入年、月、日，输出这一天是全年中的第几天。