第1章 预备知识(2)
JavaScript语言

课程名称：XML语言

1. JavaScript 简介

第1章 预备知识(2) JavaScript语言

JavaScript 的起源和发展

- 1995 年 Netscape & Sun联合推出, JavaScript的发明者是布兰登·艾奇 (Brendan Eich)。

Brendan Eich

- JavaScript 语言的前身叫作 LiveScript
- 90年代末ECMA发布语言标准ECMA-262
- ISO接受了该标准并命名为ISO-16262

JavaScript 的特点

- 基于对象的语言：预定义对象、自定义对象
- 事件驱动的语言：事件响应程序
- 动态性：直接对用户输入作出反应
- 简单性：在浏览器中解释执行
- 节省交互时间：在客户端执行，无需流量
- 跨平台性：只依赖浏览器
- 安全性：沙箱机制

JavaScript的作用

- 1. 校验用户输入的表单内容
- 2. 动态添加网页内容
- 3. 为静态网页增加一些特效（如动画效果）
- 4. 操纵HTML元素（改变元素的属性和内容）

网页中插入JavaScript脚本的方法

- 行内式
  - 直接将脚本嵌入到HTML标记的事件中
  - 格式：on事件名="代码段"

例：行内式插入脚本

```
<!DOCTYPE html>
<html>
    <head>
    <meta charset="utf-8"/>
    <title>行内式插入脚本</title>
    </head>
    <body>
        <p onClick="alert('Hello, the WEB world!');">
            Click Here
        </p>
    </body>
</html>
```

- 嵌入式
  - 使用<script>标记将脚本嵌入到HTML的<head>和<body>中
  - 格式：<script>代码段</script>

例：嵌入式插入脚本

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8"/>
    <title>嵌入式插入脚本</title>
    <script>
    function msg () {  //单行注释:建立函数
        /*多行注释: 函数体，弹出消息警告框*/
        alert("Hello, the WEB world!");
    }
    </script>
    </head>
    <body>
        <p onClick="msg()">Click Here</p> <!-- 调用函数 -->
    </body>
</html>
```

链接式

- 通过<script>标记的src属性链接外部脚本文件
- 格式：<script src="脚本文件名">代码段</script>

文件”3-3.js”的内容：

```
//在脚本文件里不需<script>标签
function msg () { //建立函数
    alert (“Hello,the WEB world!”);
}
```

例：链接式插入脚本文件

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset=“utf=8”/>
        <title>链接式插入脚本文件</title>
        <script src="3-3.js" ></script>
        </head>
    <body>
        <p onClick="msg()">Click Here</p>
    </body>
</html>
```

开发和调试JavaScript的工具

- Firefox：【工具】→【Web开发者】→【Web控制台】
- IE 11：【工具】→【开发人员工具】
- 具有代码提示功能的工具：Firebug (Firefox插件)

2. JavaScript 语言基础

第1章 预备知识(2) JavaScript语言

2.1变量

- 用“var”关键字，并可以将其初始化为任何值
  - var name =“Wang Xiaoming";
  - var age＝28;
  - var male=true;

变量的命名原则

- 首字符必须是字母、下划线（_）或美元符（$）
- 余下的字母可以是下划线、美元符号、任意字母或者数字；
- 变量名不能是关键字或保留字
- 变量名对大小写敏感
- 变量名中不能有空格、回车符或其他标点字符

JavaScript 保留字

Reserved Words in JavaScript			
abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

2.2运算符

- 1. 算术运算符
  - +、-、*、/、%、++、- -
- 2. 比较运算符
  - >、<、>=、<= 、==、===、!=、!==
- 3. 逻辑运算符
  - && 、 || 、!
- 4. 赋值运算符
  - =、 +=、-= 、*= 、/= 、%=

2.2运算符

- 5. 连接运算符
  - +
- 6. 三元运算符
  - exp？res1 : res2

- 表达式
  - 表达式是运算符和操作数的组合

2.3数据类型

- 1. 字符串（String）
  - var course=“JavaScript Programming"
  - (1) length属性: 字符串长度
  - (2) charAt方法: 返回指定位置的字符
  - (3) indexOf方法: 用于查找和定位子串
  - (4) substr方法: 截取指定位置和长度的子串
  - (5) split方法: 根据指定的符号将字符串分割成一个数组。

2. 转义字符

- \r　　回车
- \n　　换行
- \'　　单引号
- \t　　制表符
- \"　　双引号
- \b　　空格
- \f　　换页符
- \\　　反斜杠

- 3. 数值型（number）
  - var num1=23.45
  - var num2=76
  - var num3=-9e5　//-900000
- 4. 布尔型（boolean）
  - var married = true;
  - 布尔型数据的取值只有两个：true和false。
  - 布尔型数据不能用引号引起来，否则就变成字符串了。

2.4 数组（array）

- 1. 数组的定义（三种方法）
  - 第1种定义方法
    - var rank =new Array(12);//数组有12个元素，下标从0开始
  - 第2种定义方法
    - var map =new Array(“China”，“USA”，“Britain”);
  - 第3种定义方法
    - var map =["China", "USA", "Britain"];

2. 数组的常用属性和方法

- （1）length属性：用来获取数组的长度
- （2）concat方法：连接两个或更多的数组，并返回合并后的新数组
- （3）join方法：将数组的内容连接起来，返回字符串
- （4）sort方法：对数组中的元素排序（默认按字母顺序排序）

2.5 条件语句

- 在JavaScript中提供了三种条件判断语句
  - if语句
  - if else语句
  - switch语句

2.5 条件语句

1. if语句

```
if (条件) {
    执行语句
}
```

2.5 条件语句

- 2. if…else语句

```
if (条件) {
    执行语句1
} else {
    执行语句2
}
```

2.5 条件语句

- 3. switch语句

```
switch (表达式) {
    case 常量1: 语句组1
                break;
    case常量2: 语句组2
                break;
        ...
    case常量n: 语句组n
                break;
    [default: 语句组n+1]
}
```

2.6 循环语句

- 在JavaScript中提供了三种循环语句
  - while语句
  - for语句
  - for…in语句

2.6 循环语句

1. while循环

```
while (条件) {
        执行语句
}
```

2.6 循环语句

2. for 循环

```
for (初始化部分；条件部分；更新部分) {
        执行部分
}
```

2.6 循环语句

3. for…in循环

```
for(属性或数组下标 in 对象或数组) {
        执行部分
}
```

for…in循环例

```
<script>
var mycars = new Array();
mycars[0] = "Audi";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars) {
   document.write(mycars[x] + "<br />");
}
</script>
```

2.6 循环语句

break: 语句用于跳出循环

continue: 用于跳过循环中的一个迭代

2.7 函数

函数定义：

function fnName(para$_1$, para$_2$,..., para$_n$) {

    statements

    [return[expression]]

}

调用：

fnName(arg$_1$,arg$_2$,...,arg$_n$);

2.8 注释

- 单行注释以双斜线“//”开头到行末结束

```
//定义乘法函数，x为乘数、y为被乘数
//结果返回两个数的积
function a(x, y) {
    return(x*y)
}
```

- 多行注释以“/*”开头、以“*/”结束

```
/*函数名：addUser
  参数  name：用户姓名
        tel：用户电话号码
*/
```

3. 对象

第1章 预备知识(2) JavaScript语言

3.1对象

- 自定义对象
  - 用new创建对象
  - 用json表示法创建对象
- 内置对象
  - 字符串对象：String（前面已介绍）
  - 日期对象： Date
  - 数学对象： Math
  - 数组对象： Array （前面已介绍）

3.1对象

- 浏览器对象: 提供了用于访问浏览器和窗口的API
  - window
    - 提供对浏览器窗口进行操作的方法和属性
  - document
    - 提供对HTML页面进行操作的方法和属性
  - location
    - 提供对URL进行操作的方法和属性
  - navigator
    - 提供获取浏览器相关信息的属性
  - history
    - 提供对访问历史进行操作的方法和属性

3.2 自定义对象

- 对象包含两个要素：
  - 属性用来描述对象特性的一组数据，也就是若干变量；
  - 方法用来操作对象特性的若干动作，也就是若干函数。
- 访问对象的属性的两种方式：
  - 对象.属性名: 属性名只能是字符串
  - 对象[属性名]: 属性名可以是字符串或字符串类型的变量

3.2 自定义对象

- 用new创建自定义对象

```
var person=new Object();  //创建对象
person.firstname="Brendan";  //增添属性
person.lastname="Eich";  //增添属性

alert(person.firstname);  //访问对象的属性
//alert(字符串)是显示警告窗的函数
```

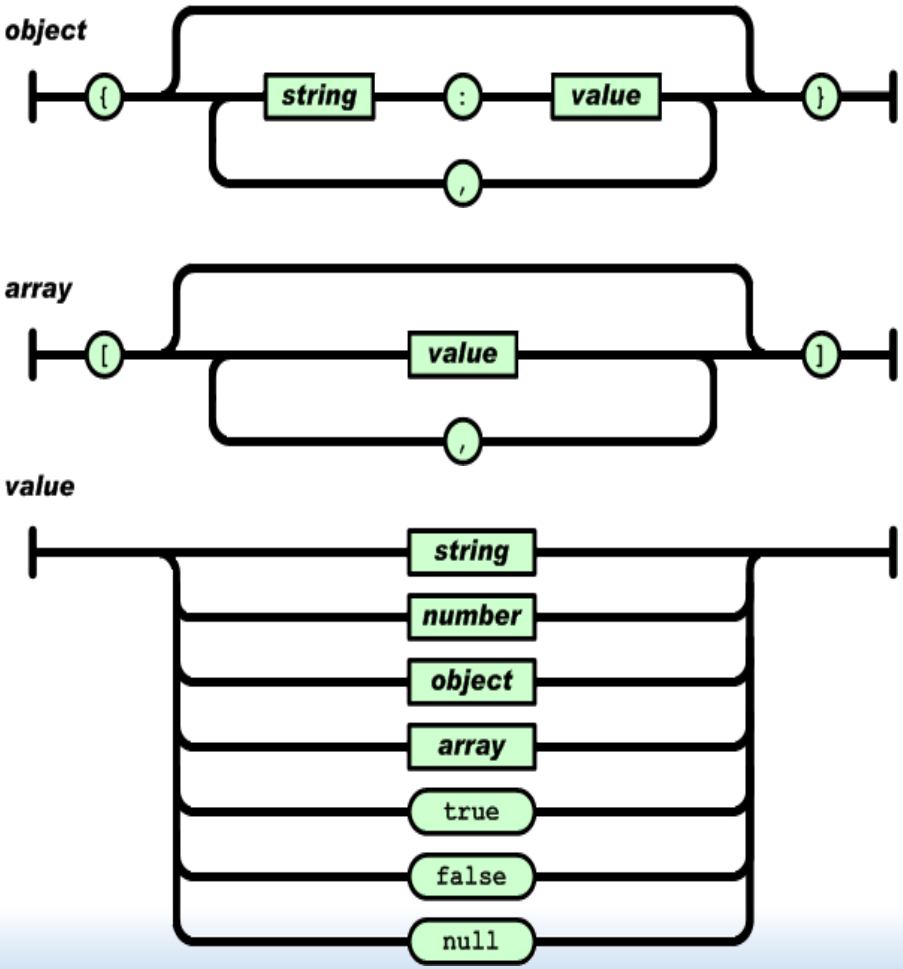
3.2 自定义对象

- 用JSON创建对象

```
var person = { //创建对象
   “firstname”:“Brendan”, //创建属性
   “lastname”:“Eich” //创建属性
    //最后一个属性后面不要加逗号“，”
}
alert(person.lastname); //访问属性
```

3.2 自定义对象

- JSON语法规则
  - 花括号“{ }”保存对象
  - 方括号“[ ]”保存数组
  - 数据由逗号“, ”分隔
  - 数据在“名称/值”数据对中
- JSON中的值可以是：
  - 数字（整数或浮点数）
  - 字符串（在双引号中）
  - 逻辑值（true 或 false）
  - 数组（在方括号中）
  - 对象（在花括号中）
  - null



JSON语法图

3.2 自定义对象

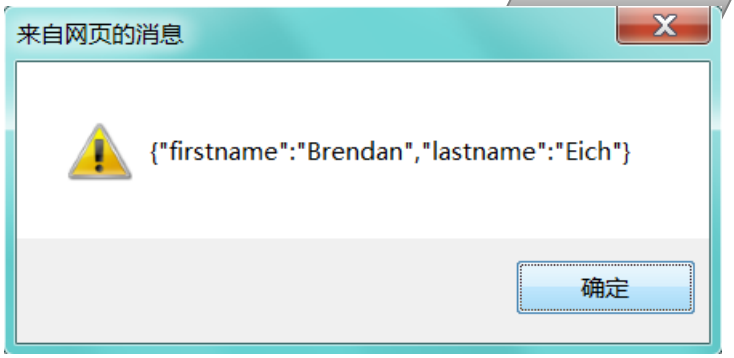
- 用JSON创建对象数组：

```
var employees = [
    { "firstName":"Brendan", "lastName": "Eich" },
    { "firstName":"Bill", "lastName":"Gates" },
    { "firstName": "Steve", "lastName": "Jobs" }
];
alert(employees[0].lastName);
```

3.2 自定义对象

- 对象和JSON字符串的转换
  - 服务器和浏览器之间以JSON字符串的形式交换数据，而在脚本中又是以对象的形式处理数据
  - 对象转换成JSON字符串
    - strJson = JSON.stringify(obj);
  - JSON字符串转换成对象
    - obJson = JSON.parse(str);

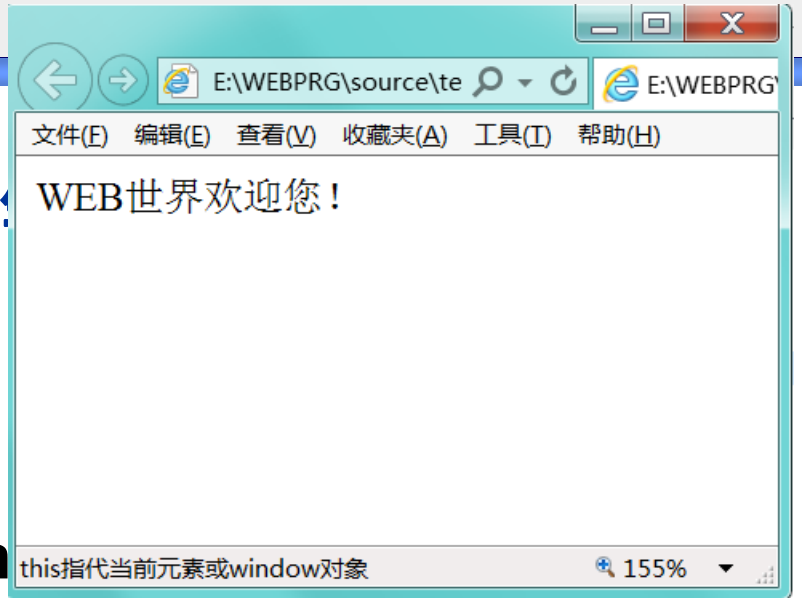
```
//创建对象
var oPerson = {
        "firstname":"Brendan",
        "lastname":"Eich"
    }
//将对象转换成JSON字符串
var strPerson=JSON.stringify(oPerson);
alert(strPerson);
//将JSON字符串转换成对象
var oPersonC=JSON.parse(strPerson);
alert(oPersonC.lastname);
```



用for…in语句遍历对象的属性

```
<script>
var oPerson = {
    "firstname":"Brendan","lastname":"Eich"
}
var strMsg= "";
for (x in oPerson) {
     strMsg += x+ ":" + oPerson[x] + "\r\n";
}
alert(strMsg);
</script>
```

3.3 this关



- 1. this指代当前元素

```
<div id="div2"
    onmouseover="this.align
    onmouseout="this.align='left';" >
```

- 2. 作为普通函数直接调用时，this指代window对象(可省略)

```
function doSomething(){
    this.status="在这里this指代window对象";
}
```

3.4内置对象

- Date对象：提供日期、时间方面的详细信息。
  - (1) var 变量名 = new Date( )：当前日期和时间
  - (2) var 变量名 = new Date(日期参数)
- 日期参数形式有以下几种：
  - new Date("month dd,yyyy hh:mm:ss");
  - new Date("month dd,yyyy");
  - new Date(yyyy,mth,dd,hh,mm,ss);
  - new Date(yyyy,mth,dd);
- 例如：
  - var d = new Date(2016, 5, 1) //2016年5月1日
  - var now = new Date(“Jan 20,2215 9:30:00”)

3.4内置对象

- Date对象常用的获得/设置日期和时间的方法
  - g/setTime()：返回/设置时间戳，以毫秒为单位。
  - g/setFullYear()：返回/设置年份，用四位数表示。
  - g/setMonth()：返回/设置月份。
  - g/setDate()：返回/设置日期。
  - g/setDay()：返回/设置星期，0 表示星期天。
  - g/setHours()：返回/设置小时数，24小时制。
  - g/setMinutes()：返回/设置分钟数。
  - g/setSeconds()：返回/设置秒钟数。
  - toLocalString()：将日期时间转化为本地格式。

3.4内置对象

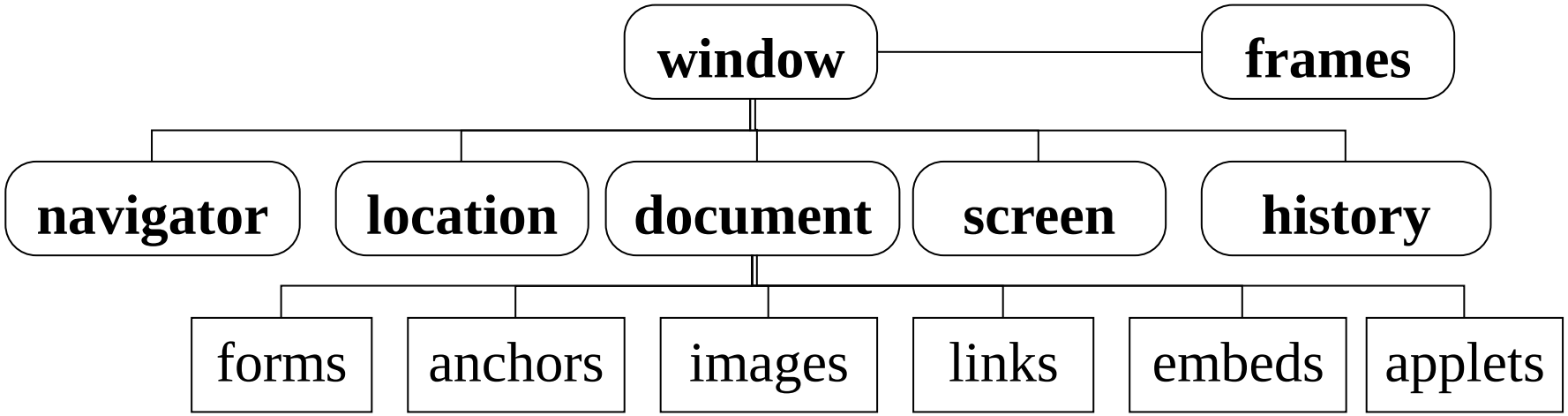
- Math对象
  - 提供除加、减、乘、除以外的一些运算，如对数，平方根等。
  - 使用时不需要生成对象实例，只要直接使用Math作为对象名即可。
- Math对象常用方法
  - ceil(x)返回大于等于 x 的最小整数。
  - floor(x)返回小于等于 x 的最大整数。
  - pow(n, m)返回 n 的 m 次幂 (nm)。
  - random()返回大于 0 小于 1 的一个随机数。
  - round(x)返回 x 四舍五入后的值。
-

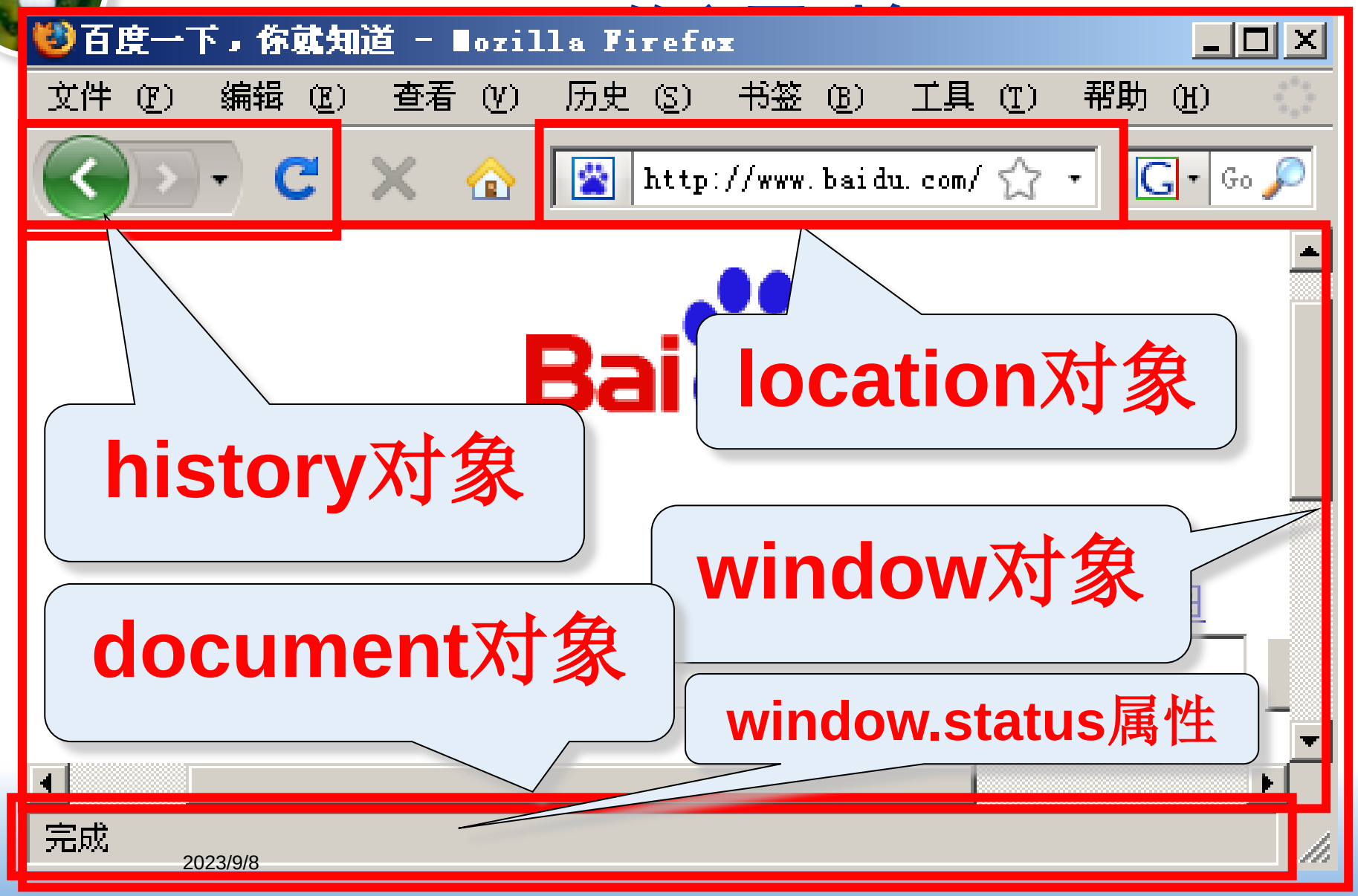
4. 浏览器对象模型

第1章 预备知识(2) JavaScript语言

浏览器对象模型BOM

- JavaScript是运行在浏览器中的，BOM提供对浏览器进行访问的API：




百度一下，你就知道 - Mozilla Firefox
文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)
http://www.baidu.com/
Go
Bai
location对象
history对象
window对象
document对象
window.status属性
完成

window对象

- window对象提供了操作浏览器窗口的API。如：

（1）调整窗口的大小和位置；

（2）打开新窗口；

（3）系统提示框；

（4）状态栏控制；

（5）定时操作。

window对象的方法

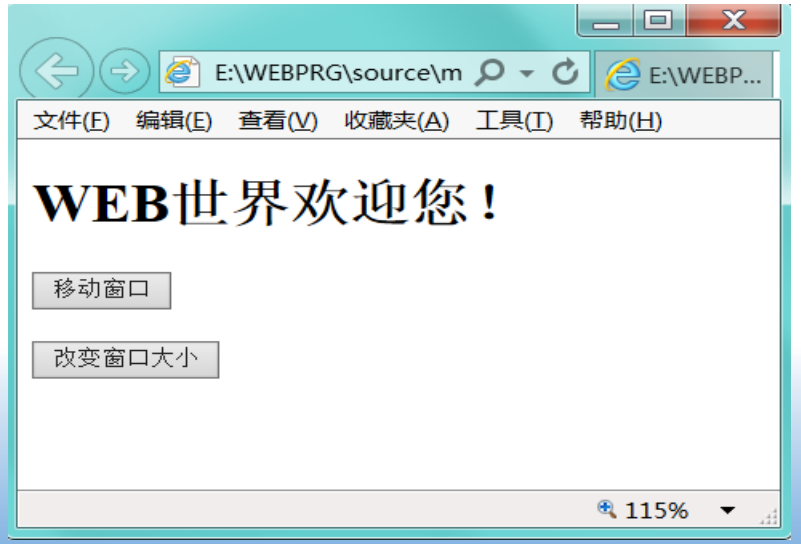
- 调整窗口的位置或大小。

（1）window.moveBy(dx, dy)

（2）window.moveTo(x, y)

（3）window.resizeBy(dw, dh)

（4）window.resizeTo(w, h)



window.open方法

- 打开新窗口:
  - window.open([url] [, 窗口名] [,窗口属性])
  - 窗口属性的各种设置间用逗号分割,可能的取值有：

（1）height, width：窗口的高度和宽度的像素值

（2）left, top：窗口边缘位置的像素值

（3）menubar, scrollba, status, toolbar, resizable: 值为yes或no, 分别表示窗口上是否出现菜单栏, 滚动条, 状态栏, 工具栏, 可否改变窗口大小

例 打开一个新窗口并控制其外观

```
<html>
<head>
<script typ
function o
    window.                                                    k",
    "width=4
}
</script>
</head>

<body>
<form>
<input typ                                                     _win()">
</form>
</body>

</html>
```

系统对话框

（1）window.alert([message])

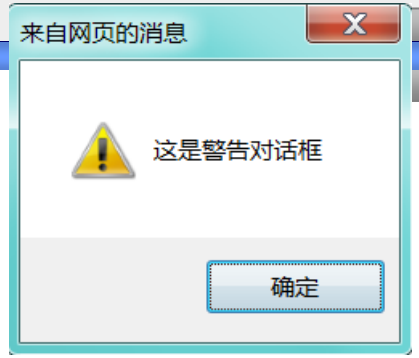
显示带有“确定”按钮的警告对话框
显示的内容。

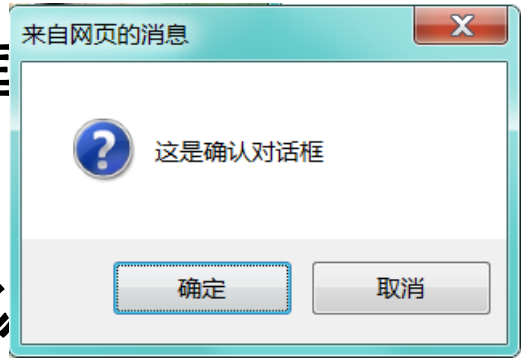
（2）window.confirm([message])

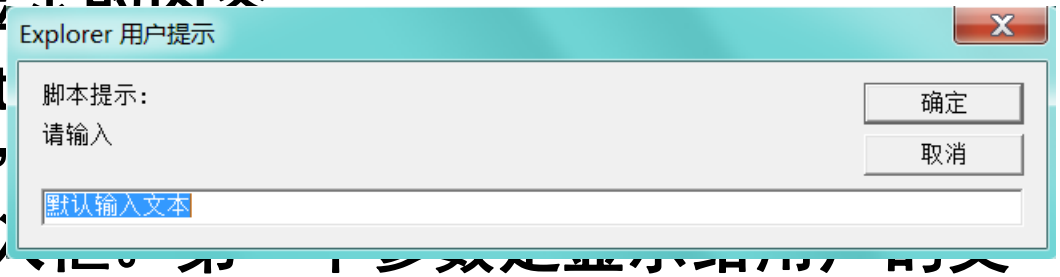
显示带有“确定”和“取消”按钮的确认
message 是要显示的内容

（3）window.prompt

显示带有“确定”
包含一个文本输入
本，第二个参数为文本输入框中的默认文本（可为
空）。







location对象

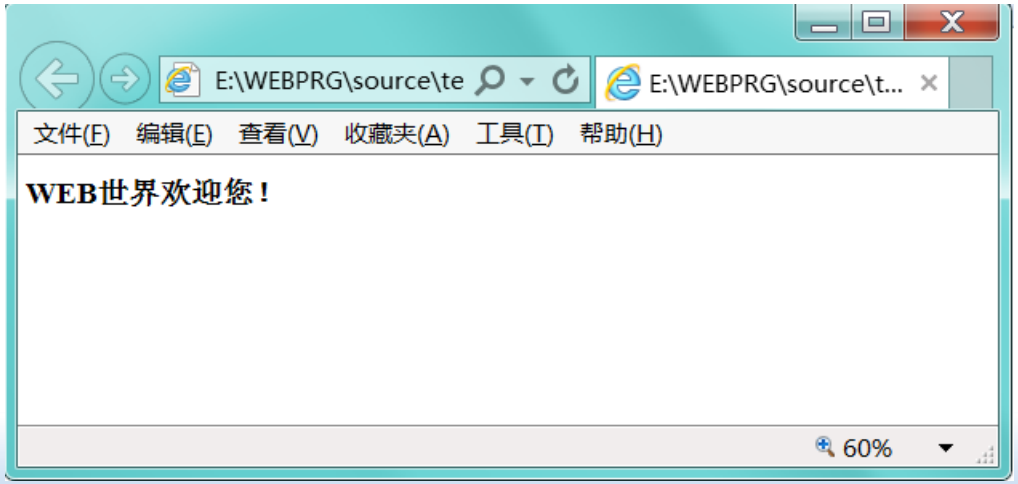
- Location对象也是window的一个属性，提供与window相关的URL信息。
- 通过location可进行网页的跳
  - location.href=URL或loc


```
<html>
<head><title>网页跳转</title
<body>
<input type="button" value=
   onclick='location="http://www.w3school.com.cn"'>
</body>
</html>
```

document对象

- document是表示HTML文档的对象(DOM).
- document.write 方法是在当前页面中输出字符串或HTML代码片段。

```
document.write(
"<h1>", "WEB世界欢迎您！", "</h1>");
```



5. 文档对象模型

第1章 预备知识(2) JavaScript语言

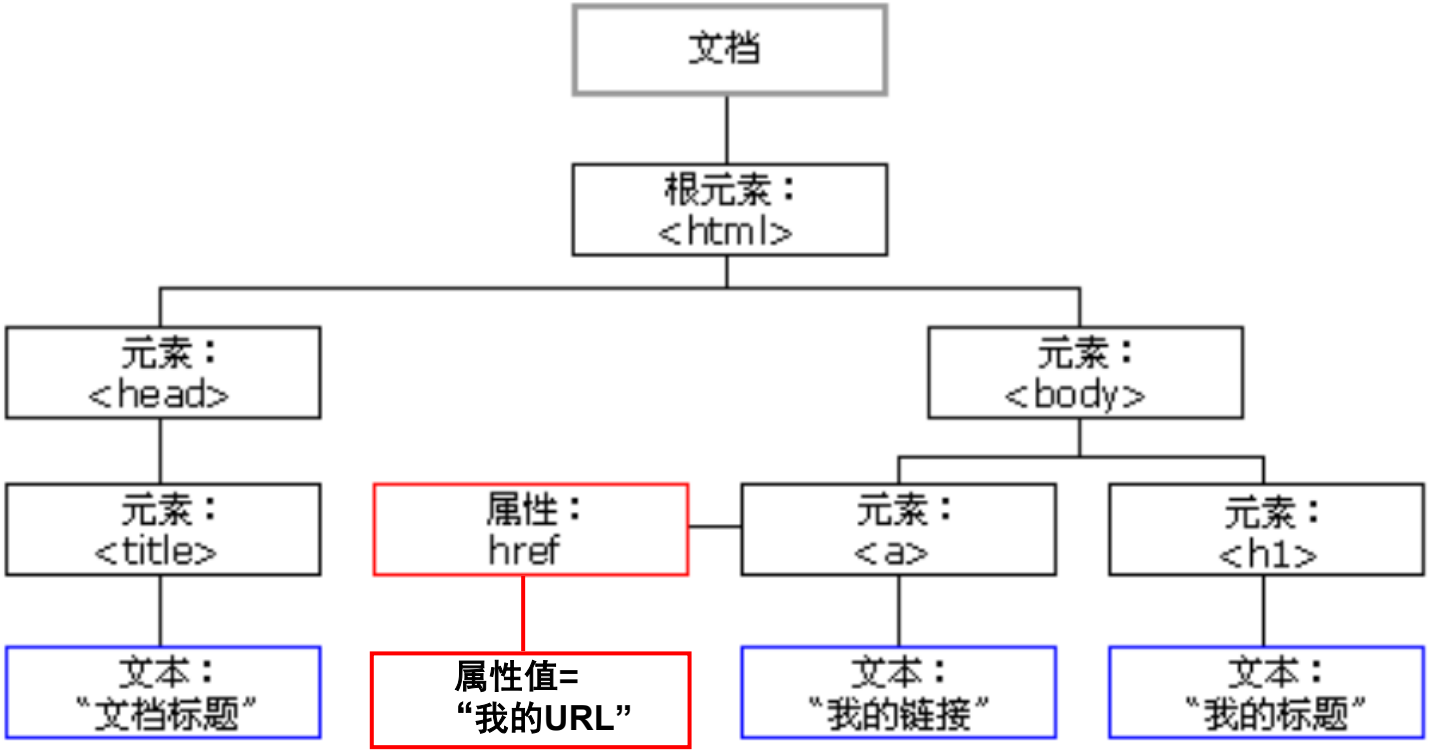
5.1 文档对象模型
(DOM: Document Object Module)

- DOM提供了访问和处理HTML页面元素的标准API
- 通过浏览器内置的document对象访问DOM 。
- 每个HTML元素都对应DOM中的一个节点(node)。每个节点都有一系列属性和方法。
- 通过节点的innerHTML属性获取或更改该节点对应的HTML元素的内容。

例 一个HTML文档

```
<html>
    <head>
        <title>文档标题</title>
    </head>
    <body>
        <a href=“我的URL”>我的连接</a>
        <h1>我的标题</h1>
    </body>
</html>
```

对应的HTML文档对象模型



- 注意：文本是元素型节点的子节点，而不是元素型节点的值。

5.2. 访问指定节点的方法

- document.getElementById(id)
  - 在页面中获得具有指定id属性值的元素型节点

使用getElementsById的示例：

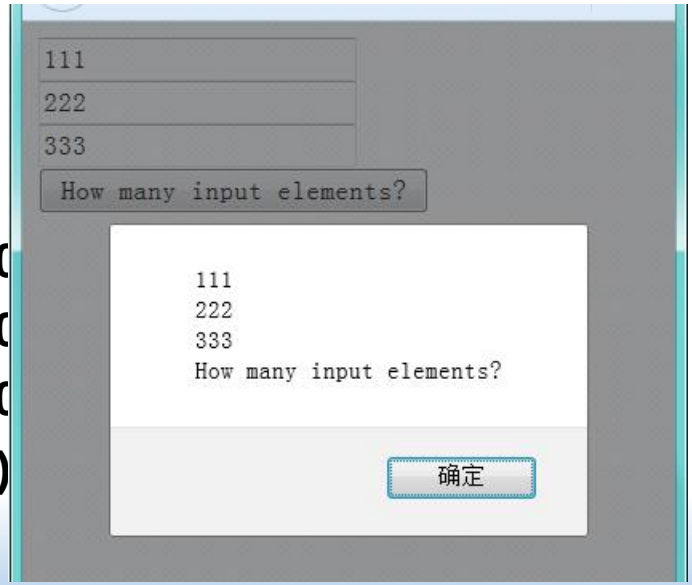
```
<html>
<head>
<script type="text/javascript">
function getValue() {
    var x=document.getElementById("myHeader")
    alert(x.innerHTML)
}
</script>
</head>
<body>
<h1 id="myHeader" onclick="getV
    </h1>
<p>点击标题，会提示出它的值。</p>
</body></html>
```

- X.getElementsByTagName(tagName)
  - 在X的子元素中获得所有具有相同标签名的元素型节点，并返回由这些节点组成的数组

- 其中，X为document对象或元素型节点

使用getElmentsByTagName的示例：

```
<html>
<head>
<script type="text/javascript">
function getElements() {
    var inputs=document.getElementsByTagName("input");
    strInputs="";
    for(i=0;i<inputs.length;i++) strInputs+=inputs[i].value+"\n";
    alert(strInputs);}
</script>
</head>
<body>
<input name="myInput" type="text"  size="20
<input name="myInput" type="text"  size="20
<input name="myInput" type="text"  size="20
<input type="button" onclick="getElements()
    value="How many input elements?" />
</body></html>
```



- X.getElementsByName(name)
  - 在X的子元素中获得所有具有相同name属性值的元素型节点，并返回由这些节点组成的数组

- 其中，X为document对象或元素型节点

使用getElmentsByName的示例：

```
<html>
<head>
<script type="text/javascript">
function getElements() {
    var inputs=document.getElementsByName(" myInput");
    strInputs="";
    for(i=0;i<inputs.length;i++) strInputs+=inputs[i].value+"\n";
    alert(strInputs);}
</script>
</head>
<body>
<input name="myInput" type="text"  size="2
<input name="myInput" type="text"  size="2
<input name="myInput" type="text"  size="2
<input type="button" onclick="getElements(
    value="How many input elements?" />
</body></html>
```

访问相关节点

- 根据已知的节点，寻找和它存在关系的节点，如父节点、子节点、兄弟节点等。
- （1）获取html和body元素对应的节点分别是：
  - var oHtml=document.documentElement
  - var oBody=document.body
- （2）访问子节点
  - 节点的childNodes属性表示子节点数组
  - var oChildren = X.childNodes
  - 其中，X为document对象或元素型节点

5.3 访问元素属性

- 获取元素属性值
  - var attrValue=node.属性名
- 设置元素属性值
  - node.属性名=新属性值;
- 删除元素属性
  - node.属性名=null;


```
<html>
<head>
<title>设置元素属性</title>
<script type="text/javascript">
function changePic(n){
    var myImg = document.getElementById("picbox");
    myImg.src="images/pic"+n+".jpg"; //更新元素的属性值

}
</script></head>
<body>
<div>
<img src="images/pic1.jpg" id="picbox" style="float:left;"/>
<ul>
  <li onmouseover="changePic(1)">沙漠古堡</li>
  <li onmouseover="changePic(2)">天山冰湖</li>
  <li onmouseover="changePic(3)">自然村落</li>
</ul>
</div></body></html>
```

5.4 访问和设置元素的内容

- 通过 “*innerHTML*” 属性可以将元素的内容（起始标记和结束标记之间的部分）改变成其他任何内容（如文本或HTML元素）
  - var a=document.getElementById("a");
  - alert(a.innerHTML);//读取元素中的HTML内容
  - a.innerHTML=“看见变化了吗？”;//设置元素中的HTML内容

5.5 控制表单元素

- 在表单中定义输入元素:

  <input name= =“名称” id=“名称” type=“…” />

- 在脚本中通过以下方式获得表单输入元素:

  var oInput = document.getElementById(“名称”);

  var value = oInput.value;

  var name = oInput.name;

```
<html><head>
<title>检查表单</title>
<script>
function checkValue() {
    if (document.getElementById("username").value == "" ) {
        alert("用户名不能为空！ ");
        return(false);
    }
    return(true);
}
</script></head>
<body>
<form method = "post"  action = ""
      onsubmit=“return(checkValue())”>
      username: <input type="text"  id="usernam“/>
      password: <input type="text"  id="password"/>
      <input type=“submit" value="登 录" />
</form>
</body></html>
```

6. 事件处理

第1章 预备知识(2) JavaScript语言

6.1 事件处理

- 事件是用户和Web页面交互时产生的各种操作。
- 浏览器运行的大部分时间都是在等待事件的发生，并在事件发生时调用相应的事件处理函数，完成事件处理。
- 发生事件的地方称为事件源，发生事件时调用的处理函数称为事件处理器。
- 事件是JavaScript和DOM之间进行交互的桥梁，当某个事件发生时，通过它的处理函数执行相应的JavaScript代码。

6.2 事件处理器的两种注册方法

- 将事件处理器连接到事件源的过程称为注册
  - 在HTML标签中使用属性绑定事件处理器：

```
<input type="button" id="myButton"
 onclick=“myfunc( );" />
```

  - 在脚本中使用对象的属性动态地绑定事件处理器：

```
document.getElementById("myButton")
.onclick = myfunc;
```

6.3 浏览器中的常用事件

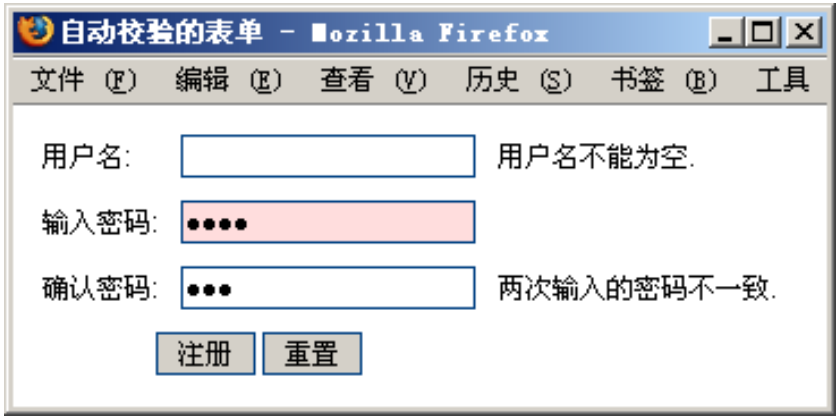
- 窗口事件(仅在 body元素中有效)
  - onload :当文档载入时执行脚本
  - onunload :当文档卸载时执行脚本
- 表单元素事件(仅在表单元素中有效)
  - onchange :当元素改变时执行脚本
  - onsubmit :当表单被提交时执行脚本
  - onreset :当表单被重置时执行脚本
  - onselect :当元素被选取时执行脚本
  - onblur :当元素失去焦点时执行脚本
  - onfocus :当元素获得焦点时执行脚本

6.3 浏览器中的常用事件

- 键盘事件
  - onkeydown :当键盘被按下时执行脚本
  - onkeypress :当键盘被按下后又松开时执行脚本
  - onkeyup :当键盘被松开时执行脚本
- 鼠标事件
  - onclick :当鼠标被单击时执行脚本
  - ondblclick :当鼠标被双击时执行脚本
  - onmousedown :当鼠标按钮被按下时执行脚本
  - onmouseup :当鼠标按钮被松开时执行脚本
  - onmousemove :当鼠标指针移动时执行脚本
  - onmouseover :当鼠标指针悬停于某元素之上时执行脚本
  - onmouseout :当鼠标指针移出某元素时执行脚本

6.4 事件的应用举例

- 利用onBlur事件自动校验表单



```
<html>
<head>
<title>利用onBlur事件自动校验表单</title>
<script src="checkInput.js"></script>
<style type="text/css">
input{border:1px solid #0066CC;}
form {font-size:12px;}
table {border:0; margin:0; padding:0}
</style></head>
<body>
<form name="register">
<table>
    <tr>
        <td>用户名:</td><td><input type="text" name="User"></td>
        <td><span id="UserResult"></span></td>
    </tr>
    <tr>
        <td>输入密码:</td><td><input type="password" name="passwd1"></td>
        <td></td>
    </tr>
    <tr>
        <td>确认密码:</td><td><input type="password" name="passwd2"></td>
        <td> <span id="pwdResult"></span></td>
    </tr>
    <tr>
        <td colspan="2" align="center">
            <input type="submit" value="注册">
            <input type="reset" value="重置">
        </td>
        <td></td>
    </tr>
</table>
</form>
</body>
</html>
```

```
function myFocus(){                    //文件名: checkinput.js
   this.style.backgroundColor="#ffdddd";
}
function myBlur(){
   this.style.backgroundColor="#ffffff";
   startCheck(this);                   //这一句是新增的验证表单的代码
}
function startCheck(oInput){
   if(oInput.name=="User"){            //如果是用户名的输入框
      if(!oInput.value){               //如果值为空
         oInput.focus();               //聚焦到用户名的输入框
         UserResult.innerHTML = "用户名不能为空";
         return;
      } else
         UserResult.innerHTML = "";
   }
   if(oInput.name=="passwd2"){
      var pass1=document.getElementsByName("passwd1")[0].value;
      var pass2=document.getElementsByName("passwd2")[0].value;
      if(pass1 != pass2)               //如果两个密码框值不相等
         pwdResult.innerHTML = "两次输入的密码不一致";
      else
         pwdResult.innerHTML = "";
   }
}
window.onload = function(){
      var elements = document.getElementsByTagName("input");
      for (var i = 0;i < elements.length;i++) {
            var type = elements[i].type;
            if (type == "text" || type == "password") {
                  elements[i].onfocus = myFocus;
                  elements[i].onblur =myBlur;
            }
      }
}
```

本章要点

- JavaScript语法
- HTML中引入JavaScript脚本的方法
- JavaScript基本数据类型
- 流程控制语句: 注意for..in的特别之处
- 自定义对象: new(), JSON
- 内置对象: 字符串, 日期, 数组, 数学计算
- 浏览器对象模型: window, location, document
- 文档对象模型: 获取指定节点对象(集合), 获取子节点集合, 属性的访问和设置
- 事件: 窗口事件, 表单事件, 鼠标事件