JavaScript 入门教程 V 1.0

简介： （1）JavaScript 是运行的客户机上的脚本语言
（2）JavaScript 一般被用来改进网页设计（特效），验证表单，检测浏览器等等

Javascript 基础语法篇

实现： 如何将 JavaScript 嵌入 html 网页中？

```
<html>
    <body>
        <script type=”text/javascript”>
            Document.write(“HELLO WORLD”);
        </script>
    </body>
</html>
```

上边的代码运行的结果：
HELLO WORLD

解释：如果需要把一段 JavaScript 插入 HTML 页面，我们需要使用 <script> 标签（同时使用 type 属性来定义脚本语言）。这样就可以告诉浏览器，js 程序从何处开始<script>,从何处结束</script>。Document.write 是 js 命令，用于向页面输出信息。如果没有<script>标签，那么浏览器会把 Document.write(“HELLO WORLD”)当做纯文本输出。那些不支持 js 的浏览器会把脚本作为页面的内容来显示。为了防止这种情况发生，我们可以使用这样的 HTML 注释标签：

```
<html>
    <body>
        <script type=”text/javascript”>
            <!--
            Document.write(“HELLO WORLD”);
            //> 正斜杠是 JavaScript 的注释符号，它会阻止 JavaScript 编译器对这一行的编译
        </script>
    </body>
</html>
```

放置： （1）在 html 中如何放置 js？

页面中的脚本会在页面载入浏览器后立即执行。我们并不是想所有的脚本都这样。有时，我们希望当页面载入时执行脚本，而另外的时候，我们则希望当用户触发事件时才执行脚本。

（2）位于 head 部分的脚本。

当脚本被调用时，或者当事件被触发时，脚本就会被执行。当你把脚本放置到 head 部分后，就可以确保在需要使用脚本之前，它已经被载入了。

```
<html>
    <head>
        <script type=”text/javascript”>
            Document.write(“HELLO WORLD”);
        </script>
    </head>
</html>
```

（3）位于 body 部分的脚本。

在页面载入时脚本就会被执行。当你把脚本放置于 body 部分后，它就会生成页面的内容。

```
<html>
    <body>
        <script type=”text/javascript”>
            Document.write(“HELLO WORLD”);
        </script>
    </body>
</html>
```

（4）同时在 head 和 body 部分的脚本。

你可以在文档中放置任何数量的脚本，因此你既可以把脚本放置到 body，又可以放置到 head 部分。

```
<html>
    <head>
        <script type=”text/javascript”>
            Document.write(“head 部分”);
        </script>
    </head>

    <body>
        <script type=”text/javascript”>
            Document.write(“body 部分”);
        </script>
    </body>
</html>
```

（5）调用外部 js 脚本。

有时，你也许希望在若干个页面中运行 JavaScript，同时不在每个页面中写相同的脚本。

为了达到这个目的，你可以将 JavaScript 写入一个外部文件之中。然后以 .js 为后缀保存这个文件。

注意：外部文件不能包含 **<script>** 标签。然后把 **.js** 文件指定给 **<script>** 标签中的 **"src"** 属性，就可以使用这个外部文件了：

```
<html>
    <head>
        <script src=”xxxx.js”></script>
    </head>

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

注释： （1）单列注释。

```
<script type=”text/javascript”>

    //这是标题头
    document.write(“<h1>this is a title</h1>”);

        //这是段落
        document.write(“<p>this is a content</p>”);
</script>
```

（2）多列注释。

```
<script type=”text/javascript”>

    /*

    下面的代码将输出一个标题，一个段落

    */

    document.write(“<h1>this is a title</h1>”);
    document.write(“<p>this is a content</p>”);
</script>
```

变量： （1）什么是变量？

变量是存储信息的容器，那么在数学里边也经常使用到变量，我们还记得在数学中我们使用变量是需要先声明一个变量的，那么我们的 js 脚本语言使用变量只是需要先声明，他们区别只在于声明的方式不同。

（2）声明（创建）js 变量和赋值。

在 js 中我们使用 var 声明变量，声明变量的同时我们可以为变量赋一个默认值：

```
Var a;
Var b=”student”;
Var c=3;
```

那么我们看见以上我们声明了 3 个变量，一个没有赋值，一个赋的 String 类型的值，一个是 int 类型的值，那有些同学就要问了，是不是写错了啊？这里，我很明确的告诉大家，js 是一门弱类型的语言，它是不区分类型的，我们也可以不声明，直接使用，那么它会自动为我们声明，不过，鉴于一个编程的严谨性，我建议大家预先声明再使用。当然 js 也可以重复声明，原来的值是不会丢失的，

不过一个优秀的程序员，应该是不会犯这种低级的错误吧。

注意：js 脚本变量的声明是区分大小写的：var a 和 var A 是两个不同的变量

（3）js 变量的运算。

```
Var A=3;
Var B;
Var C;
B=A+3;
C=B+8;
```

正如数学一样，在 js 脚本中，你可以使用变量来做运算。现在上边的例子你猜到 B 和 C 的值了吗？

运算符：　（1）运算符类型。

算术运算符
赋值运算符
比较运算符
逻辑运算符
条件运算符

（2）算术运算符详解。

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	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数 (保留整数)	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

（3）赋值运算符详解。

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

=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

（4）用于字符串的 + 运算符。

"+" 运算符用于把文本值或字符串变量加起来（连接起来）。如需把两个或多个字符串变量连接起来，请使用 + 运算符。

Var a="where are you going?";

Var b="I will going to school.";

Var c=a+b;

那么你猜到 c 的值了么？c="where are you going?I will going to school.";

(5)字符串和数字之间的加法运算。

Var x=5+5;

Document.write(x);

Var x="5"+"5";

Document.write(x);

Var x="5"+5;

Document.write(x);

Var x=5+"5";

Document.write(x);

赶快动手试一下吧。。。

（6）比较运算符。

比较运算符在逻辑语句中使用，以测定变量或值是否相等。给定 x=5，下面的表格解释了比较运算符：

运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

```
Var x=20;
If(x<21){
    Document.write("he is too young!");
}
```

（7）逻辑运算符。

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

（8）条件运算符。

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

```
Var x=1;
Var y=2;
X=(y)?value1:value2
```

声明变量 x,y；x 等于 y 吗？如果是 true 返回 x 的值 value1，否则返回 x 的值 value2.

逻辑判断： (1)条件判断种类。

在您编写代码时，经常需要根据不同的条件完成不同的行为。可以在代码中使用条件语句来完成这个任务。

在 JavaScript 中，我们可以使用下面几种条件语句：

If 语句：在一个指定的条件成立时执行代码。

if...else 语句：在指定的条件成立时执行代码，当条件不成立时执行另外的代码。

if...else if....else 语句：使用这个语句可以选择执行若干块代码中的一个。

switch 语句：使用这个语句可以选择执行若干块代码中的一个。

（2）if 详解。

语法：
```
if(条件){
    执行代码块
}
```
案例：
```
<script type="text/javascript">
    Var a=10;
    Var b=19;
    If(a<=b){
        Document.write("You are too great!");
    }
</script>
```
（3）if...else...详解。

语法：
```
if(条件){
    条件满足时执行的代码块
}else{
    条件不满足时执行的代码块
}
```
案例：<script type="text/javascript">

```
Var a=10;
Var b=19;
If(a<=b){
    Document.write("You are too great!");
}else{
    Document.write("I like this");
}
</script>
```

（4）if...else if...else...详解。

语法：if(条件 1){
条件 1 满足时执行的代码块
}else if(条件 2){
条件 2 满足时执行的代码块
}else{
都不满足时执行的代码块
}

案例：<script type="text/javascript">

```
Var a=10;
Var b=19;
If(a<b){
    Document.write("You are too great!");
}else if(a>b){
    Document.write("I like this");
}else{
    Document.write("It is too bad!");
}
</script>
```

(5) switch 详解。

语法：switch(n){

```
Case 1:
    执行代码块 1
    Break
```

```
        Case 2:
            执行代码块 2
            Break
        Default:
            不满足以上两个 case 的执行的代码块
    }
案例：<script type="text/javascript">
        Var a=10;
        switch(a){
            case 1:
                document.write("first");
                break;
            case 2:
                document.write("second");
                break;
            default:
                document.write("other");
        }
    </script>
```

消息框： (1)消息框类别。

警告框

确认框

提示框

(2)警告框。

Alert("文本");

(3)确认框。

Confirm("文本");

(4)提示框。

Prompt("文本","默认值");

函数： (1)将脚本编写为函数，就可以避免页面载入时执行该脚本。

函数包含着一些代码，这些代码只能被事件激活，或者在函数被调用时才会执行。

你可以在页面中的任何位置调用脚本（如果函数嵌入一个外部的 .js 文件，那么甚至可以从其他的页面中调用）。

函数可以在页面任何地方定义。

案例：<html>

```
<head>
<script type="text/javascript">
Function test(){
Alert("消息框");
}
</script>
</head>
<body>
<from>
<input type="button" value="click me" onclick="test()">
</form>
</body>
</html>
```

（2）如何定义函数

语法：function test(var a,var b){

要执行的程序

}

Var a 和 var b 是函数定义的要转入的值或者变量，大括号定义了函数的开始和结束

注意：无参函数必须在后边加括号

语法：function test(){

要执行的程序

}

（3）return 语句

Return 语句用来规定函数返回的值，因此,需要返回某个值的函数必须使用这个 return 语句

语法：function prod(a,b){

X=a*b;

Return x;

}

所以，当您调用 prod 函数的时候需要传入 ab 两个参数值

（4）javascript 变量的生存周期。

当您在函数内声明了一个变量后，就只能在该函数中访问该变量。当退出该函数后，这个变量会被撤销。这种变量称为本地变量。您可以在不同的函数中使用名称相同的本地变量，这是因为只有声明过变量的函数能够识别其中的每个变量。

如果您在函数之外声明了一个变量，则页面上的所有函数都可以访问该变量。这些变量的生存期从声明它们之后开始，在页面关闭时结束。

循环遍历： (1)for 循环：将一段代码循环执行指定的次数。

语法：for(变量=开始值;变量<=结束值;变量=变量+进步值){

需执行的代码

}

实例：下面的例子定义了一个循环程序，这个程序中 i 的起始值为 0。每执行一次循环，i 的值就会累加一次 1，循环会一直运行下去，直到 i 等于 10 为止。

<html>

<body>

<script type=”text/javascript”>

For(var i=0;i<10;i++){

Document.write(“The number is ”+i);

Document.write(“
”);

}

</script>

</body>

</html>

结果：The number is 0

The number is1

The number is 2

The number is 3

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

（2）while 循环：当指定条件为 true 时循环执行代码。

语法：while(变量<=结束值){

需要执行的代码

}

实例：while(i<10){

Document.write("The number is "+i);

Document.write("
");

}

结果：The number is 0

The number is1

The number is 2

The number is 3

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

（3）do…while…循环语法：do...while 循环是 while 循环的变种。该循环程序在初次运行时会首先执行一遍其中的代码，然后当指定的条件为 true 时，它会继续这个循环。所以可以这么说，do...while 循环为执行至少一遍其中的代码，即使条件为 false，因为其中的代码执行后才会进行条件验证。

语法：do{

需执行的代码

}while(变量<=结束值)

案例：<html>

<body>

<script type=”text/javascript”>

Var i=0;

do{

Document.write(“The number is ”+i);

Document.write(“
”);

}while(i<=1)

</script>

</body>

</html>

结果：The number is 0;

The number is 1;

（4）break 语句详解。

案例：<html>

<body>

<script type=”text/javascript”>

Var i=0;

For(i:i<=10;i++){

If(i==3){

Break;

}

```
                Document.write(“The number is ”+i);
                Document.write(“<br/>”);
            }
        </script>
    </body>
</html>
```

结果：The number is 0;

The number is 1;

The number is 2;

（5）continue 语句详解: continue 命令会终止当前的循环，然后从下一个值继续运行。

案例：

```
<html>
    <body>
        <script   type=”text/javascript”>
            Var i=0;
            For(i:i<=5;i++){
                If(i==3){
                    continue;
                }
                Document.write(“The number is ”+i);
                Document.write(“<br/>”);
            }
        </script>
    </body>
</html>
```

结果：The number is 0;

The number is 1;

The number is 2;

The number is 4;

The number is 5;

(6)for…in 循环详解：声明用于对数组或者对象的属性进行循环操作，循环中的代码每执行一次，

就会对数组或者对象的属性进行一次操作。

语法：for(变量 in 对象){

在此执行代码

}

实例：<html>

<body>

<script type=”text/javascript”>

Var x;

Var listtest=new AyyayList();

Listttest[0]=”a”;

Listttest[1]=”b”;

Listttest[2]=”c”;

For(x in listtest){

Document.write(listtest[x]+”
”);

}

</script>

</body>

</html>

事件： (1)事件是可以被 javascript 侦测到的行为。

网页中的每个元素都可以产生某些可以触发 JavaScript 函数的事件。比方说，我们可以在用户点击某按钮时产生一个 onClick 事件来触发某个函数。事件在 HTML 页面中定义。

事件举例：

- 鼠标点击
- 页面或图像载入
- 鼠标悬浮于页面的某个热点之上
- 在表单中选取输入框
- 确认表单
- 键盘按键

注意：事件通常与函数配合使用，当事件发生时函数才会执行

（2）常用事件案例。

案例 1：onFouces,onBlur 和 onChange 这三个事件通常相互配合来验证表单

下面是一个使用 onChange 事件的例子。用户一旦改变了域的内容，checkMail() 函数就会被调用

<input type="text" size="30" onChange="checkMail()">

案例 2：onSubmit 用于提交表单之前验证表单域，

下面是一个使用 onSubmit 事件的例子。当用户单击表单中的确认按钮时，checkForm() 函数就会被调用。假若域的值无效，此次提交就会被取消。checkForm() 函数的返回值是 true 或者 false。如果返回值为 true，则提交表单，反之取消提交。

<from method="post" action="xxx.html" Onsubmit="return checkFrom()">

案例 3：onMouseOver 和 onMouseOut：

onMouseOver 和 onMouseOut 用来创建"动态的"按钮

下面是一个使用 onMouseOver 事件的例子。当 onMouseOver 事件被脚本侦测到时，就会弹出一个警告框：

<img src="test.jpg" width="100" height="100">

异常捕获：（1）异常捕获类型。

使用 Try….catch….异常捕获（主要适用于 ie5 以上内核的浏览器，也是最床用的异常捕获方式）

使用 **onerror** 事件异常捕获，这种捕获方式是比较古老的一种方式，目前一些主流的浏览器暂不支持

本文基于失效开发的前提，所以这个地方只讲解 try…catch…异常捕获，有兴趣的同学可以自己找找 onerror 事件的捕获方式

（2）try….catch….异常捕获详解：try 部分包含要运行的代码，catch 部分包含错误运行时要执行的代码。

语法：try{

//在此执行的代码

}catch(err){

```
    //在此处理错误的代码
}
```

案例：下面一个例子，由于误写了 alert()，所以错误发生了。不过这一次，catch 部分捕获到了错误，并用一段准备好的代码来处理这个错误。这段代码会显示一个自定义的出错信息来告知用户所发生的事情。

```
<html>
    <head>
        <script type=”text/javascript”>
            Function test(){
                Try{
                    Alerrtt(“welcome  dear”);
                }catch(err){
                    Alert(“此页面出现了一个错误，描述：”+err.description);
                }
            }
        </script>
    </head>
    <body>
        <input type=”button” value=”click me” onClick=”test()”>
    </body>
</html>
```

特殊字符：（1）在 javascript 中我们经常使用反斜杠来插入一些特殊字符，比如在文本字符串中插入省略号、换行符、引号和其他特殊字符。

案例：var txt=”what are you going “to” do?”;

Alert(“txt”);

输出：what are you going

如何解决这样的问题呢？

要解决这个问题，就必须把在 "to" 中的引号前面加上反斜杠 (\)。这样就可以把每个双引号转换为字面上的字符串。

案例：var txt=”what are you going \“to\” do?”;

Alert(“txt”);

输出：what are you going “to” do?

(2)特殊字符的插入是非常简单易懂的，下边这些特殊字符也都可以使用反斜杠添加到文本字符串中。

\'	单引号
\"	双引号
\&	和号
\\	反斜杠
\n	换行符
\r	回车符
\t	制表符
\b	退格符
\f	换页符

以上就是 javascript 基本用法的相关知识点，还是非常简单的吧，相信同学们，看到这里大家一定感慨原来 javascript 是这么简单啊，那么首先，我要先恭喜各位，已经正式进入 javascript 编程，但是，我同时也要告诉大家，这是进入 javascript 编程的基本语法，换句话说，我们还没有进入逻辑语法，高级 javascript 以及对象，图像的处理。不过大家不用着急，本教程写到这里是希望大家把基础掌握熟练了，这对我们接下来的研究很有帮助。谢谢。

Javascript 对象篇

对象简介： （1）javascript 是面向对象的编程语言（oop），对象有自己的属相和方法。

字符串对象： （1）属性：属性指与客户有关的值。

在下面的例子中，我们使用字符串对象的长度属性来计算字符串中的字符数目。

案例：<script type="text/javascript">

Var txt="contratulations";

Document.write(txt.length);

</script>

输出：15

（2）方法：方法指对象可以完成的行为或者功能。

在下面的例子中，我们使用字符串对象的 toUpperCase() 方法来显示大写字母文本。

案例：<script type="text/javascript">

Var txt="contratulations";

Document.write(txt.toUpperCase());

</script>

输出：CONTRATULATIONS

日期对象： （1）日期的定义。

Date 对象用于处理日期和时间。

可以通过 new 关键词来定义 Date 对象。以下代码定义了名为 myDate 的 Date 对象：

Var mydate=new Date();

注意：Date 对象自动使用当前的日期和时间作为其初始值。

(2)日期对象的操作。

在下面的例子中，我们为日期对象设置了一个特定的日期 (2012 年 2 月 2 日)：

案例 1：var mydate=new Date();

Mydate=setFullYear(2012,1,2);

注意：表示月份的参数介于 0 到 11 之间。也就是说，如果希望把月设置为 2 月，则参数应该是 1。

在下面的例子中，我们将日期对象设置为 5 天后的日期：

案例 2：var mydate=new Date();

Mydate.setDate(mydate.getDte()+5);

注意：如果增加天数会改变月份或者年份，那么日期对象会自动完成这种转换

(3)比较日期。

日期对象也可用于比较两个日期。

下面的代码将当前日期与 2012 年 2 月 2 日做了比较：

```
Var mydate=new Date();
Mydate.setFullDate(2012,1,2);
Var today=new Date();
If(today>mydate){
    Alert("today is"+today);
}else{
    Alert("today is"+mydate);
}
```

数组对象： （1）数组的定义和赋值：数组对象用来在单独的变量名中存储一系列的值。

我们使用关键词 new 来创建数组对象。下面的代码定义了一个名为 myArray 的数组对象：

```
Var myarray=new Array();
```

有两种向数组赋值的方法（你可以添加任意多的值，就像你可以定义你需要的任意多的变量一样）。

第一种方式：

```
Var mycars=new Ayyar();
Mycars[0]="Saab";
Mycars[1]="Volvo";
Mycars[2]="BMW";
```

也可以使用一个整数自变量来控制数组的容量：

```
Var mycars=new Ayyar(3);
Mycars[0]="Saab";
Mycars[1]="Volvo";
Mycars[2]="BMW";
```

第二种方式：

Var mycars=new Ayyar(“Saab”,”Volvo”,”BMW”);

注意：如果你需要在数组内指定数值或者逻辑值，那么变量类型应该是数值变量或者布尔变量，而不是字符变量。

(2)访问数组。

通过指定数组名以及索引号码，你可以访问某个特定的元素。

案例：document.write(mycars[1]);

输出：Volvo;

(3)修改已有数组中的值。

只要向指定下标号添加一个新值即可：

Mycars[0]=”Opel”;

输出 document.write(mycars[0]);

Opel;

逻辑对象： （1）创建 Boolean 对象

Boolean（逻辑）对象用于将非逻辑值转换为逻辑值（true 或者 false）

使用关键词 new 来定义 Boolean 对象。下面的代码定义了一个名为 myBoolean 的逻辑对象：

Var myboolean=new Boolean();

注意：如果逻辑对象无初始值或者其值为 0、-0、null、""、false、undefined 或者 NaN，那么对象的值为 false。否则，其值为 true（即使当自变量为字符串 "false" 时）！

下面的所有的代码行均会创建初始值为 false 的 Boolean 对象。

var myBoolean=new Boolean();

var myBoolean=new Boolean(0);

var myBoolean=new Boolean(“”);

var myBoolean=new Boolean(null);

var myBoolean=new Boolean(false);

var myBoolean=new Boolean(NaN);

下面的所有的代码行均会创建初始值为 false 的 Boolean 对象。

var myBoolean=new Boolean(1);

```
var myBoolean=new Boolean(true);
var myBoolean=new Boolean("true");
var myBoolean=new Boolean("false");
```

算数对象：(1) 算数值。

Math（算数）对象的作用是：执行普通的算数任务。

Math 对象提供多种算数值类型和函数。无需在使用这个对象之前对它进行定义。

JavaScript 提供 8 种可被 Math 对象访问的算数值：

- 常数
- 圆周率
- 2 的平方根
- 1/2 的平方根
- 2 的自然对数
- 10 的自然对数
- 以 2 为底的 e 的对数
- 以 10 为底的 e 的对数

这是在 Javascript 中使用这些值的方法：（与上面的算数值一一对应）

- Math.E
- Math.PI
- Math.SQRT2
- Math.SQRT1_2
- Math.LN2
- Math.LN10
- Math.LOG2E
- Math.LOG10E

（2）算数方法。

除了可被 Math 对象访问的算数值以外，还有几个函数（方法）可以使用。

实例 1：下面的例子使用了 Math 对象的 round 方法对一个数进行四舍五入

```
Document.write(Math.round(4.7));
```

输出：5;

实例 2：下面的例子使用了 Math 对象的 random() 方法来返回一个介于 0 和 1 之间的随机数：

Document.write(Math.random());

输出：0.12212xxxxx(或者 0-1 之间的任何小数);

实例 3：下面的例子使用了 Math 对象的 floor() 方法和 random() 来返回一个介于 0 和 10 之间的随机数：

Document.write(Math.floor(Math.random())*11);

输出：8(或者 0-10 之间的任意整数);

正则表达式对象： （1）定义 RegExp。

RegExp 对象用于存储检索模式。

来定义 RegExp 对象。以下代码定义了名为 patt1 的 RegExp 对象，其模式是 "e"：

Var myreg=new RegExp("e");

当您使用该 RegExp 对象在一个字符串中检索时，将寻找的是字符 "e"。

（2）RegExp 对象的方法。

RegExp 对象有三种方法 test(),exec()以及 complie()。

Test():检索字符串中的指定值。返回值是 true 或 false。

案例：Var myreg=new RegExp("e");

Document.write(myreg.test("this is a test example!"));

由于上边已经定义检索字符是"e",所以这个地方的输出：true;

Exec():检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

案例：Var myreg=new RegExp("e");

Document.write(myreg.exec("this is a test example!"));

由于上边已经定义检索字符是"e",并且检索字符串中存在"e",所以这个地方的输出：e

扩展学习：您可以向 RegExp 对象添加第二个参数，以设定检索。例如，如果需要找到所有某个字符的所有存在，则可以使用 "g" 参数 ("global")。在使用 "g" 参数时，exec() 的工作原理如下：

- 找到第一个 "e"，并存储其位置
- 如果再次运行 exec()，则从存储的位置开始检索，并找到下一个 "e"，并存储其位置

案例：Var myreg=new RegExp(“e”,”g”);

Do{

Result=myreg.exec(“this is a test example”);

Document.write(result);

}while(result!=null)

由于这个字符串中 3 个 "e" 字母，代码的输出将是：eeenull;

Compile():既可以改变检索模式，也可以添加或删除第二个参数。

案例：Var myreg=new RegExp(“e”);

Document.write(myreg.exec(“this is a test example!”));

Var myreg=new RegExp(“d”);

Document.write(myreg.compile(“this is a test example!”));

由于字符串中存在 "e"，而没有 "d"，以上代码的输出是：truefalse;

以上就是 javascript 的基础对象篇，那么学到这里，大家是否能完全掌握了呢？当然，我木有完全的把所有相关的对象都列举出来，那也是不现实的，编写此教程的目的在于培养大家的编码思想，对象的属性和方法等知识点其实也难死记硬背的，所以只要你掌握了编码的思想，你就可以举一反三，根据客户的需求，查相关的帮助文档，我希望大家把这一篇的知识好好的巩固和学习一下，下一篇我们将深入 javascript 高级研究。

Javascript 高级

浏览器检测： （1）概念描述

本教程中几乎所有的代码均可在任何支持 JavaScript 的浏览器中运行。不过个别的代码无法运行于特定的浏览器，特别是老式的浏览器。

所以，有些时候对访问者的浏览器类型及版本进行检测是很有帮助的，然后可在此基础上为访问者提供合适的信息。

要做到这一点，最好的办法是使你的网页变得足够聪明，这样的话它就可以不同的方式对待不同类型的浏览器。

JavaScript 包含一个名为 Navigator 的对象，它就可以完成上述的任务。

Navigator 包含了有关访问者浏览器的信息，包括浏览器类型、版本等等

（2）Navigator 对象详解

AppName:保存浏览器类型

AppVersion:保存浏览器的版本信息

实例 1：

```
<html>
    <body>
        <script  type="text/javascript">
            Var browser= navigator.appname;
            Var messversion= navigator.appVersion;
            Var version=parseFloat(messversion);
            Document.write("browser  name"+ browser+"<br/>");
            Document.write("browser version"+version);
        </script>
    </body>
</html>
```

上面例子中的 browser 变量存有浏览器的名称，比如，"Netscape" 或者 "Microsoft Internet Explorer"。

上面例子中的 appVersion 属性返回的字符串所包含的信息不止是版本号而已，但是现在我们只关注版本号。我们使用一个名为 parseFloat() 的函数会抽取字符串中类似十进制数的一段字符并将之返回，这样我们就可以从字符串中抽出版本号信息了。

重要事项：在 IE 5.0 及以后版本中，版本号是不正确的！在 IE 5.0 和 IE 6.0 中，微软为 appVersion 字符串赋的值是 4.0。怎么会出现这样的错误呢？无论如何，我们需要清楚的是，JavaScript 在 IE6、IE5 和 IE4 中的获得的版本号是相同的。

下面的脚本会根据访问者的浏览器类型显示不同的警告。

实例 2：

```
<html>
    <body onload="test()">
        <script  type="text/javascript">
            Function test(){
                Var browser= navigator.appname;
                Var messversion= navigator.appVersion;
                Var version=parseFloat(messversion);
                If(browser=="Netscape"|| browser=="Internet Explorer "&&version>=4){
                    Alert("Your browser  is good enough!");
                }else{
                    Alert("It is time to upgrade your browser!");
```

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

Cookies： （1）什么是 cookie?

cookie 是存储于访问者的计算机中的变量。每当同一台计算机通过浏览器请求某个页面时，就会发送这个 cookie。你可以使用 JavaScript 来创建和取回 cookie 的值。

（2）cookie 的使用范围

名字 cookie：当访问者首次访问页面时，他或她也许会填写他/她们的名字。名字会存储于 cookie 中。当访问者再次访问网站时，他们会收到类似 "Welcome John Doe!" 的欢迎词。而名字则是从 cookie 中取回的。

密码 cookie：当访问者首次访问页面时，他或她也许会填写他/她们的密码。密码也可被存储于 cookie 中。当他们再次访问网站时，密码就会从 cookie 中取回。

日期 cookie：当访问者首次访问你的网站时，当前的日期可存储于 cookie 中。当他们再次访问网站时，他们会收到类似这样的一条消息："Your last visit was on Tuesday August 11, 2005!"。日期也是从 cookie 中取回的。

（3）创建和存储 cookie

在这个例子中我们要创建一个存储访问者名字的 cookie。当访问者首次访问网站时，他们会被要求填写姓名。名字会存储于 cookie 中。当访问者再次访问网站时，他们就会收到欢迎词。

首先，我们会创建一个可在 cookie 变量中存储访问者姓名的函数：

```
Function setCookie(c_name,value,expiredays){
    Var exdate=new Date();
    Exdate.setDate(exdate.getDate()+expiredays);

Document.cookie=c_name+”=”+escape(value)+((expiredays==null?””: “;expires=”+exdate.toGMTString()));
}
```

上面这个函数中的参数存有 cookie 的名称、值以及过期天数。

在上面的函数中，我们首先将天数转换为有效的日期，然后，我们将 cookie 名称、值及其过期日期存入 document.cookie 对象。

之后，我们要创建另一个函数来检查是否已设置 cookie：

```
Function getCcookie(c_name){
    If(document.getCookie.length>0){
        C_start=document.cookie.indexof(c_name+”=”);
```

```
            If(C_start!=-1){
                C_start=C_start+c_name+length;
                C_end=document.cookie.indexof(“;”,C_start);
                If(C_end==-1){
                    C_end=document.cookie.length;
                    Return
        unespace(document.cookie.substring(C_start,C_end))
        ;
                }
            }
            Return “”;
        }
    }
```

上面的函数首先会检查 document.cookie 对象中是否存有 cookie。假如 document.cookie 对象存有某些 cookie，那么会继续检查我们指定的 cookie 是否已储存。如果找到了我们要的 cookie，就返回值，否则返回空字符串。

最后，我们要创建一个函数，这个函数的作用是：如果 cookie 已设置，则显示欢迎词，否则显示提示框来要求用户输入名字。

```
function checkCookie(){
  username=getCookie('username')
  if (username!=null && username!=""){
    alert('Welcome again '+username+'!')
  }else{
    username=prompt('Please enter your name:',"")
    if (username!=null && username!="") {
      setCookie('username',username,365)
    }
  }
}
```

这是所有的代码：

```
<html>
<head>
<script type="text/javascript">
    function getCookie(c_name){
    if (document.cookie.length>0){
        c_start=document.cookie.indexOf(c_name + "=")
    if (c_start!=-1){
        c_start=c_start + c_name.length+1;
        c_end=document.cookie.indexOf(";",c_start);
        if (c_end==-1){
            c_end=document.cookie.length;
                return
unescape(document.cookie.substring(c_start,c_end));
            }
```

```
    }
    return "";
}

function setCookie(c_name,value,expiredays){
    var exdate=new Date();
    exdate.setDate(exdate.getDate()+expiredays);
    document.cookie=c_name+ "=" +escape(value)+
    ((expiredays==null) ? "" :
";expires="+exdate.toGMTString());
}

function checkCookie(){
    username=getCookie('username');
    if (username!=null && username!=""){
    alert('Welcome again '+username+'!');
}else {
  username=prompt('Please enter your name:',"");
     if (username!=null && username!="") {
            setCookie('username',username,365);
    }
  }
}
</script>
</head>

<body onLoad="checkCookie()">
</body>
</html>
```

表单验证：