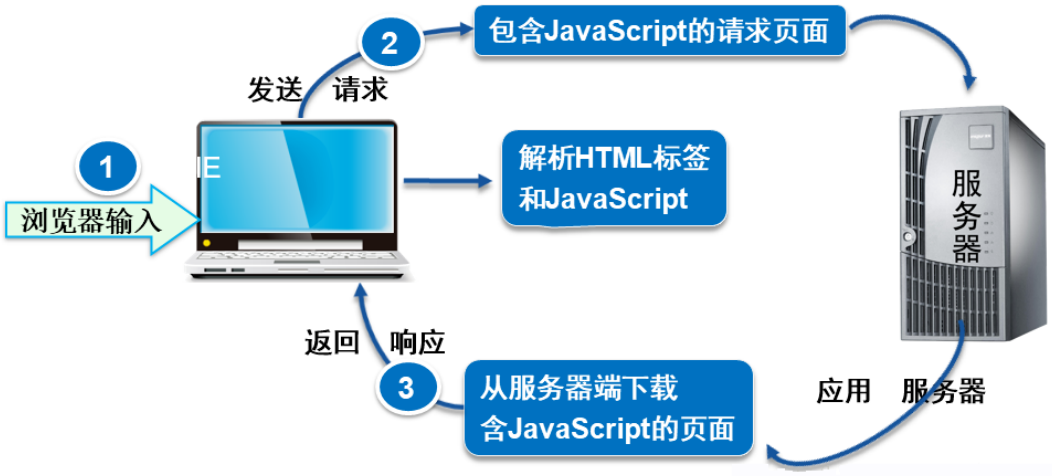
更多嵌入式 Linux 学习资料，请关注：一口 Linux 回复关键字:1024

个人微信


公众号


## 前言

JavaScript 诞生于 1995 年，它的出现主要是用于处理网页中的前端验证。所谓的前端验证，就是指检查用户输入的内容是否符合一定的规则。




比如：用户名的长度，密码的长度，邮箱的格式等。但是，有的同学可能会有疑问，这些验证，后端不也可以进行验证吗？

确实，后端程序的确可以进行这些验证，但你要清楚，在 1995 年那个年代，网速是非常慢的，向后端发送一个请求，浏览器很久才能得到响应，那这无疑是一种非常不好的用户体验。

为了解决前端验证的问题，当时的浏览器巨头 NetScape（网景）公司就开发出一种脚本语言，起初命名为 LiveScript，后来由于 SUN 公司的介入更名为了 JavaScript。

Java 和 JavaScript 是没有什么关系的，只不过当时 Java 非常流行，为了蹭热度，才将 LiveScript 更名为 JavaScript，它们的关系就像雷锋和雷峰塔的关系一样，没啥关系。

但是，浏览器开发商不止网景一家，还有一个大家都知道的公司，微软公司，它们的主打产品是 IE（Internet Explorer）浏览器，当网景公司的 Netscape Navigator 浏览器推出 JavaScript 语言时，微软就急了啊，好家伙，人网景都推出了专门用于前端验证的语言，不仅大大减少了后端程序的压力，还提高了用户的体验。我微软这么大的公司不也得整一个，在 1996 年，微软公司在其最新的 IE3 浏览器中引入了自己对 JavaScript 的实现 JScript。

于是在市面上存在两个版本的 JavaScript，一个网景公司的 JavaScript 和微软的 JScript，虽然当时浏览器的巨头是网景，但是网景的浏览器是收费的，虽然微软的 IE 浏览器在全球的市场份额远远不及网景，但是微软的拳头产品是 Windows 操作系统，每一个操作系统都自带一个 IE 浏览器并且免费，那么，未来的发展大家可能也想到了，网景让微软给干倒闭了，1998 年 11 月，网景被美国在线（AOL）收购。

老大哥就是老大哥，为了抢先获得规则制定权，网景最先将 JavaScript 作为草案提交给欧洲计算机制造商协会，也就是 ECMA 组织，希望能将 JavaScript 做

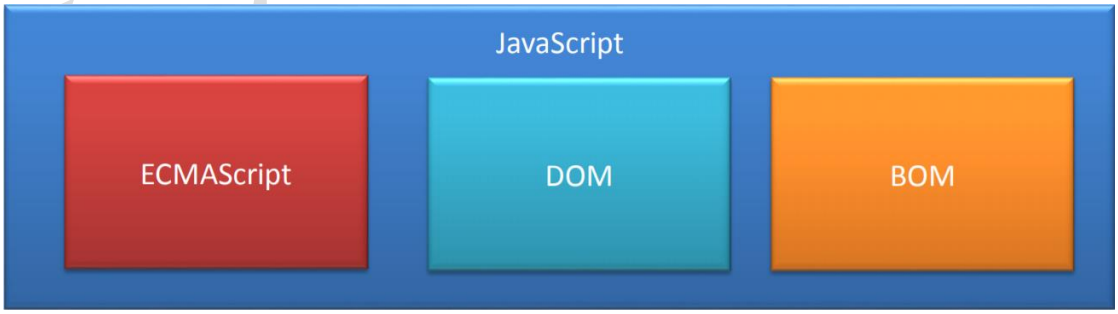
成行业标准，最终在网景、SUN 以及微软等公司的参与下，由一众程序员和相关组织人员组成的第 39 技术委员会也就是 TC39 发布了 ECMA-262 标准，这个标准定义了名为 ECMAScript 的全新脚本语言，为啥又来了个 ECMAScript？
因为 Java 是 SUN 的商标，SUN 授权了 NetScape 可以叫 JavaScript，但是 ECMA 没有 SUN 的授权就不能叫 JavaScript，哪怕 NetScape 成员特别希望 ECMA 把它叫做 JavaScript，但是 ECMA 也有成员并不希望这个标准就叫 JavaScript，总之经过几轮磋商和博弈，ECMAScript 这个名字就定下来。

# 一、什么是 JavaScript

JavaScript 是一种基于对象(Object)和事件驱动(Event Driven)并具有安全性能的脚本语言。
使用它的目的是与 HTML 超文本标记语言、Java 脚本语言（Java 小程序）一起实现在一个 Web 页面中链接多个对象，与 Web 客户交互作用。从而可以开发客户端的应用程序等。
它是通过嵌入或调入在标准的 HTML 语言中实现的。
它的出现弥补了 HTML 语言的缺陷，它是 Java 与 HTML 折衷的选择，具有以下几个基本特点： JavaScript 特点：

- 是一种解释性脚本语言（代码不进行预编译）。
- 主要用来向 HTML（标准通用标记语言下的一个应用）页面添加交互行为。
- 可以直接嵌入 HTML 页面，但写成单独的 js 文件有利于结构和行为的分离。
- 跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如 Windows、Linux、Mac、Android、iOS 等）。

一个完整的 JavaScript 实现应该由以下三个部分构成：



# 二、JavaScript 基本数据结构

## 1. JavaScript 代码的加入

将 JavaScript 脚本加入文档语法

```
<Script Language ="JavaScript">
 JavaScript 语言代码;
 JavaScript 语言代码;
 …….
</Script>
```

说明:

1. 通过标识<Script>...</Script>指明 JavaScript 脚本源代码将放入其间。
2. 通过属性 Language ="JavaScript"说明标识中是使用的何种语言，这里是 JavaScript 语言,表示在 JavaScript 中使用的语言。
3. 可将<Script>...</Script> 标识放入<head>.. </Head> 或 <Body> ...</Body> 之间。
   将 JavaScript 标识放置<Head>... </Head>在头部之间，使之在主页和其余部分代码之前装载， 从而可使代码的功能更强大;
   可以将 JavaScript 标识放置在<Body>... </Body>主体之间以实现某些部分动态地创建文档。
4. 我们也可以把 JavaScript 代码保存在独立的文件中(扩展名为.js),在 Html 中引入这个文件。

### 实例 1

下面是将 JavaScript 脚本嵌入到 Web 文档中的例子:

```
<HTML>

<Head>
<Script Language ="JavaScript">
document.write("document.write()的说明:<br>document 是指当前的 html 文档,使用
JavaScript 脚本中的 document 对象的 write 方法,<br>可以向 html 文档中添加信息!");
document.close();
alert("我的第一 JavaScript!麻烦你点一下确定按钮!");
</Script>
```

```
</Head>

<body>

<p>上面的文字是用 JavaScript 加入的.</p>

</body>
</HTML>
```

说明：Document. write()是文档对象的输出函数，其功能是将括号中的字符或变量值输出到口；document. close()是将输出关闭。

alert()是 JavaScript 的窗口对象方法，其功能是弹出一个具有 OK 对话框并显示（）中的字符串.

### 实例 2 嵌入式 js

importJs.html

```
<html>
<head>
</head>
<body>
<script src="importJs.js">
</script>
<p>
上面的一段话是 importJs.js 文件写入的信息!
</p>
</body>
</html>
```

importJs.js

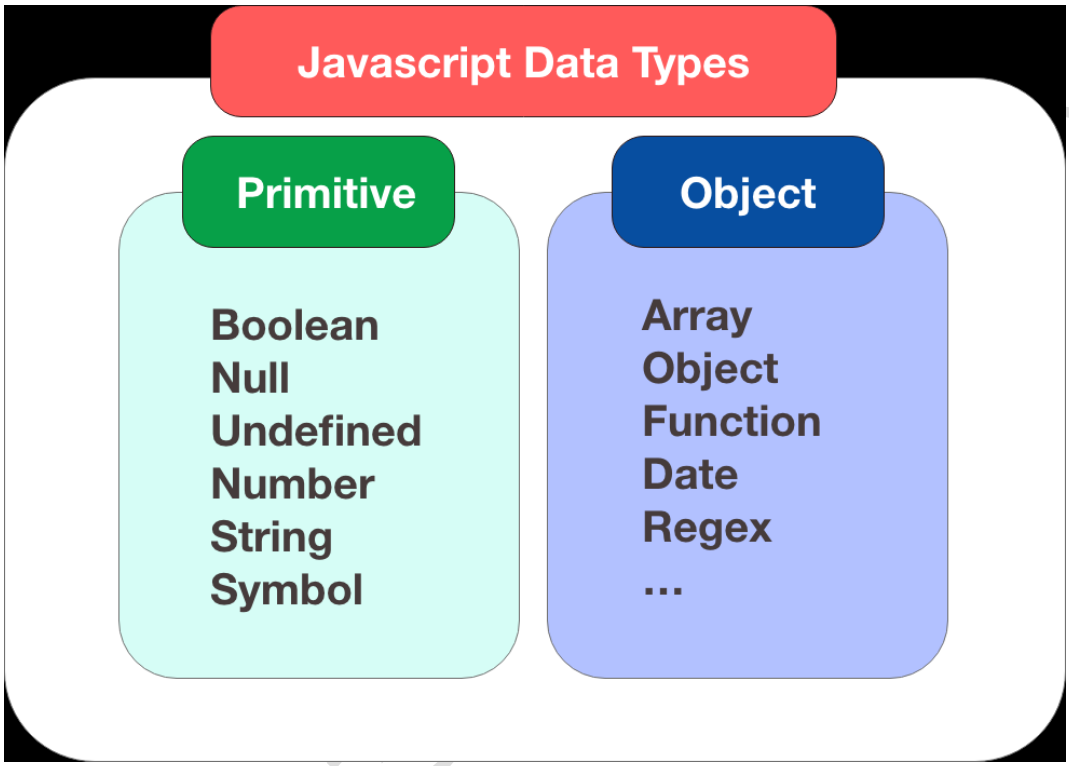
```
//嵌入的Js 代码
document.write("<h4>");
document.write("人活的是一种心态!");
document.write("</h4>");
```

注意：在 js 代码中就不要使用<script></script>去包含 Js 代码,因为在引用的 Html 中已经包含了插入 Js 的标记.

JavaScript 注释有两种：单行注释和多行注释。。单行注释用双反斜杠“//”表示。多行注释是用“/”和“/”括起来的一行到多行文字。

## 2. 基本数据类型

JavaScript 是弱类型脚本语言，声明变量时无需指定变量的数据类型。
JavaScript 变量的数据类型是解释时动态决定的。但是 JavaScript 的值保存在内存中，也是数据类型的。
值类型(基本类型)：**字符串（String）、数字(Number)、布尔(Boolean)、对空（Null）、未定义（Undefined）、Symbol**。
引用数据类型（对象类型）：**对象(Object)、数组(Array)、函数(Function)，还有两个特殊的对象：正则（RegExp）和日期（Date）**。




本文只对基本数据类型做一些介绍，引用数据类型不作介绍。

## JavaScript 拥有动态类型

JavaScript 拥有动态类型。
这意味着相同的变量可用作不同的类型：

```
var x;               // x 为 undefined
var x = 5;           // 现在 x 为数字
var x = "John";      // 现在 x 为字符串
```

变量的数据类型可以使用 typeof 操作符来查看

```
typeof "John"                // 返回 string
typeof 3.14                  // 返回 number
typeof false                 // 返回 boolean
```

```
typeof [1,2,3,4]             // 返回 object
typeof {name:'John', age:34} // 返回 object
```

## JavaScript 字符串

字符串是存储字符（比如 "yikoulinux"）的变量。
字符串可以是引号中的任意文本。您可以使用单引号或双引号：

```
var carname="Volvo XC60";
var carname='Volvo XC60';
```

您可以在字符串中使用引号，只要不匹配包围字符串的引号即可：

```
var answer="It's alright";
var answer="He is called 'Johnny'";
var answer='He is called "Johnny"';
```

## JavaScript 数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

```
var x1=34.00;      //使用小数点来写
var x2=34;         //不使用小数点来写
```

极大或极小的数字可以通过科学（指数）计数法来书写：

```
var y=123e5;      // 12300000
var z=123e-5;     // 0.00123
```

## JavaScript 布尔

布尔（逻辑）只能有两个值：true 或 false。

```
var x=true;
var y=false;
```

## JavaScript 数组

下面的代码创建名为 cars 的数组：

```
var cars=new Array();
cars[0]="Saab";
```

```
cars[1]="Volvo";
cars[2]="BMW";
```

或者 (condensed array):

```
var cars=new Array("Saab","Volvo","BMW");
```

数组下标是基于零的，所以第一个项目是 [0]，第二个是 [1]，以此类推。

### JavaScript 对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。空格和折行无关紧要。声明可横跨多行：

```
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};
```

对象属性有两种寻址方式：

```
name=person.lastname;
name=person["lastname"];
```

### Undefined 和 Null

Undefined 这个值表示变量不含有值。

可以通过将变量的值设置为 null 来清空变量。

```
cars=null;
person=null;
```

## 3. 变量

变量的主要作用是存取数据、提供存放信息的容器。对于变量必须明确变量的命名、变 量的类型、变量的声明及其变量的作用域。 当您声明新变量时，可以使用关键词 "new" 来声明其类型：

# 变量命名

JavaScript 中的变量命名同其计算机语言非常相似：

- 1.必须是一个有效的变量，即变量以字母开头，中间可以出现数字如 test1、text2 等。 除下划线（一）作为连字符外，变量名称不能有空格、（＋）、（－）、（，）或其它符号。
- 2.不能使用 JavaScript 中的关键字作为变量。 在 JavaScript 中定义了 4 0 多个类键字，这些关键是 JavaScript 内部使用的，不能作为 变量的名称。如 Var、int、double、true 不能作为变量的名称。

| break | do | instanceof | typeof | case |
|---|---|---|---|---|
| else | new | var | catch | finally |
| return | void | continue | for | switch |
| while | default | if | throw | delete |
| in | try | function | this | with |
| debugger | false | true | null | |

保留关键字:

| class | enum | extends | super | const | export |
|---|---|---|---|---|---|
| import | implements | let | private | public | yield |
| interface | package | protected | static | | |

其他不建议的关键字:

| abstract | double | goto | native | static | boolean |
|---|---|---|---|---|---|
| enum | implements | package | super | byte | export |
| import | private | synchronize | char | extends | int |
| protected | throws | class | final | interface | public |
| transient | const | float | long | short | volatile |
| arguments | encodeURI | Infinity | Number | RegExp | undefined |
| isFinite | Object | String | Boolean | Error | RangeError |
| parseFloat | SyntaxError | Date | eval | JSON | ReferenceError |
| TypeError | decodeURI | EvalError | Math | URIError | decodeURIComponent |
| Function | NaN | isNaN | parseInt | Array | encodeURICOmponent |

- 3.数据类型

JavaScript 中一共有 5 种基本数据类型：

- 字符串型（String）
- 数值型（Number）
- 布尔型（Boolean）
- undefined 型（Undefined）
- null 型（Null）

JavaScript 中，变量以可以不作声明，而在使用时再根据数据的类型来确其变量的类型

```
var carname=new String;
var x=      new Number;
var y=      new Boolean;
var cars=   new Array;
var person= new Object;
```

JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。
JavaScript 变量可以在使用前先作声明，并可赋值。通过使用 var 关键字对变量作声明。
对变量作声明的最大好处就是能及时发现代码中的错误；因为 JavaScript 是采用动态编译的， 而动态编译是不易发现代码中的错误，特别是变量命名的方面。

## 变量的声明及其作用域

对于变量还有一个重要性——那就是变量的作用域。
在 JavaScript 中同样有全局变量和局部变量。全局变量是定义在所有函数体之外，其作用范围是整个函数；而局部变量是定义在函数体之内，只对其该函数是可见的，而对其它函数则是不可见的。

# 4. 表达式和运算符

- (1)算术运算符

```
+（加） 、-（减）、 *（乘）、 /（除）、 %（取模） 、|（按位或）、&(按位与)、<<（左移）、
>>（右移）、 >>>（右移，零填充）。
-（取反）、~（取补）、++（递加 1）、--（递减 1）。
```

- (2)比较运算符

比较运算符它的基本操作过程是，首先对它的操作数进行比较，尔后再返回一个 true 或 false 值，有 8 个比较运算符:
<(小于)、>(大于)、<=(小于等于)、>=(大于等于)、= =(等于)、!=(不等于)。

- (3)布尔逻辑运算符

&&(与),|| (或),!(非)

- (4)位运算符

~(取补), &（与）,|(或), ^（异或）

- (5)三元操作符 主要格式 如下:

操作数? 结果 1 : 结果 2

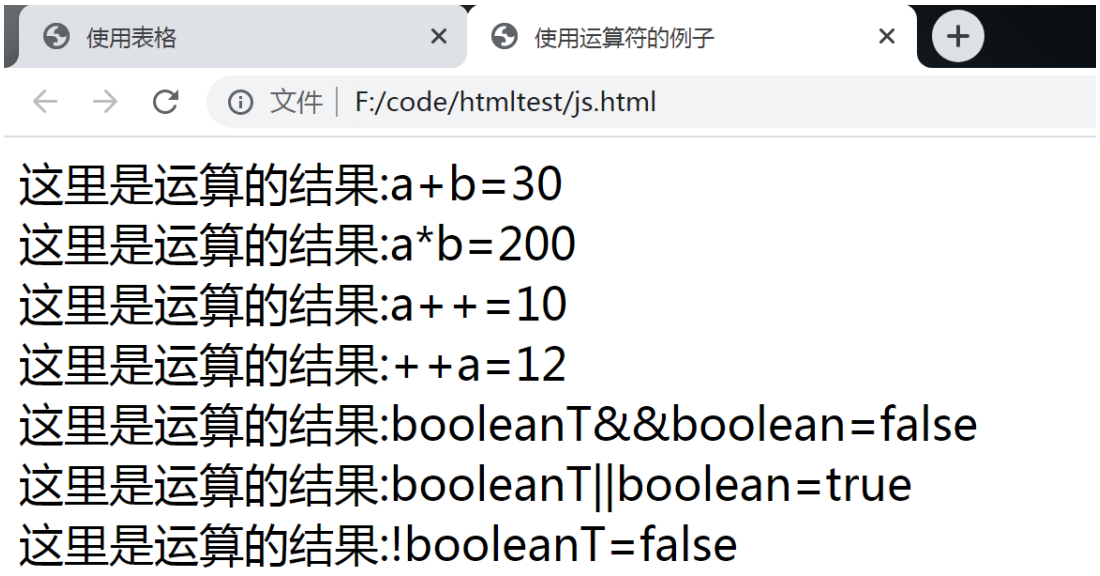
若操作数的结果为真，则表述式的结果为结果 1，否则为结果 2

## 5. 综合例子

```
<html>
<head>
<title>使用运算符的例子</title>
<script Language="JavaScript">
//定义全局整型变量
var a=10;
var b=20;
//定义全局字符串
var c="这里是运算的结果:";
//定义全局布尔变量
var booleanT=true;
var booleanF=false;
//定义一个计算函数
function calculate(){
//定义局部变量
var temp;
var d=a+b;
e=a*b;
f="a+b=";
g="a*b=";
temp=c+f;
document.write(temp);
document.write(d+"<br>");
temp=c+g;
document.write(temp);
```

```
document.write(e);
document.write("<br>");
document.write(c);
document.write("a++=");
document.write(a++);
document.write("<br>");
document.write(c);
document.write("++a=");
document.write(++a);
document.write("<br>");
document.write(c);
document.write("booleanT&&boolean=");
document.write(+booleanT&&booleanF);
document.write("<br>");
document.write(c);
document.write("booleanT||boolean=");
document.write(booleanT||booleanF);
document.write("<br>");
document.write(c);
document.write("!booleanT=");
document.write(!booleanT);
//关闭文档
document.close();
}
//调用计算函数
calculate();
</script>
</head>
<body>
</body>
</html>
```

运行结果:




# 三、JavaScript 程序构成

JavaScript 脚本语言的基本构成是由控制语句、函数、对象、方法、属性等，来实现编程的。
使用方法类似于 C 语言。

## 1. 流程控制语句

### if else

实例： 如果时间小于 10:00，则生成问候 "早上好"，如果时间大于 10:00 小于 20:00，则生成问候 "今天好"，否则生成问候 "晚上好"：

```
<html>
<body>
<script type="text/javascript">
var d = new Date()//获得日期
var time = d.getHours()//获得小时
if (time<10)
{
    document.write("<b>早上好</b>");
}
else if (time>=10 && time<20)
```

```
{
    document.write("<b>今天好</b>");
}
else
{
    document.write("<b>晚上好!</b>");
}
document.write("<p>" + d + "</p>");
</script>
<p>
这是一个最简单的 if 语句!
</p>
<p>
如时间在 10 点以前会显示"早上好"!
</p>
</body>
</html>
```

**今天好**

Mon Apr 04 2022 11:42:54 GMT+0800 (中国标准时间)

这是一个最简单的if 语句!

如时间在10 点以前会显示"早上好"!

for

```
<html>
 <head>
 <title>for 循环的例子</title>
  <script type="text/javascript">
   for (i = 1; i <= 6; i++)
   {
    document.write("<h" + i + ">This is header " + i )
    document.write("</h" + i + ">" )
   }
  </script>
 </head>
 <body>
```

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

# This is header 1

## This is header 2

### This is header 3

#### This is header 4

##### This is header 5

###### This is header 6

while

```
<html>
<head>
<title>while 循环的例子</title>
<script type="text/javascript">
i = 0;
while (i <= 5)
{
document.write("The number is " + i);
document.write("<br>");
i++;
}
</script>
```

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

## do while

```
<html>
<body>
    <script type="text/javascript">
        i = 0
        do
        {
            document.write("The number is " + i)
            document.write("<br>")
            i++
        }
        while (i <= 5)
    </script>
<p>do...while 循环的例子,至少要执行一次</p>
</body>
</html>
```

## switch

```
<!DOCTYPE html>
<html>
<head>
<title>一口 Linux</title>
</head>
<body>

<p>点击下面的按钮来显示今天是周几：</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>
<script>
function myFunction(){
 var x;
 var d=new Date().getDay();
 switch (d){
    case 0:x="今天是星期日";
     break;
```

```
  case 1:x="今天是星期一";
       break;
   case 2:x="今天是星期二";
       break;
       case 3:x="今天是星期三";
    break;
   case 4:x="今天是星期四";
    break;
   case 5:x="今天是星期五";
       break;
   case 6:x="今天是星期六";
    break;
  }
 document.getElementById("demo").innerHTML=x;
}
</script>

</body>
</html>
```

点击下面的按钮来显示今天是周几：

点击这里

今天是星期一

## 四、函数

JavaScript 函数的定义

```
function 函数名（参数）{
    函数体;.
    return 表达式;
}
```

说明：

- 函数由关键字 function 定义(关键字为小写)
- 参数列表，是传递给函数使用或操作的值，其值可以是常量，变量或其它表达式
- 通过指定函数名（实参）来调用一个函数

- 使用 Return 将值返回
- 函数名对大小写是敏感的

JavaScript 对大小写敏感。关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

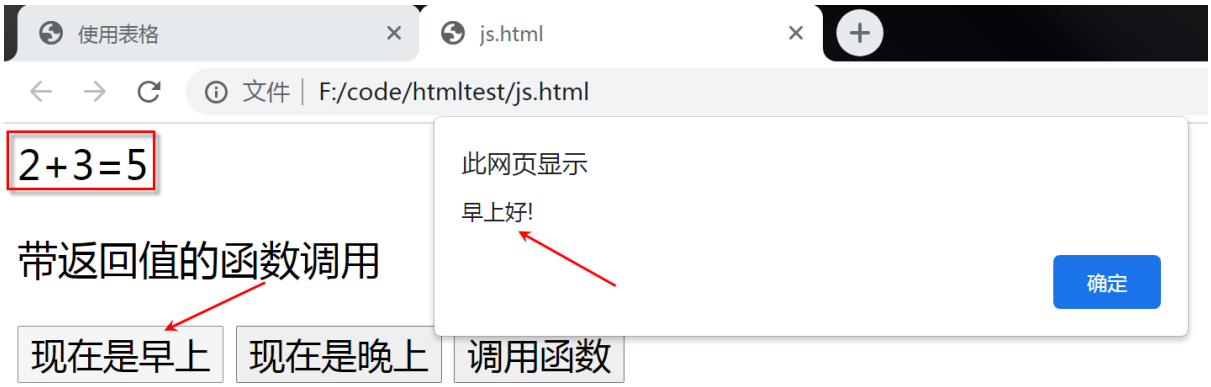
## arguments .Length

在函数的定义中，我们看到函数名后有参数列表，这些参数变量可能是一个或几个。 那么怎样才能确定参数变量的个数呢？在 JavaScript 中可通过 arguments .Length 来检查参数的个数。

## 1. 函数综合实例

```
<html>
<head>
 <script type="text/javascript">
  //普通传参
  function welcome(txt)
  {
    alert(txt)
  }
  //传参带返回值
  function total(numberA,numberB)
  {
    return numberA+ numberB
  }
  //测试参数个数
  function TestArgLen(a,b,c,d)
  {
    len= TestArgLen.arguments.length;
    alert("len="+len);
  }
 </script>
</head>
<body>
 <script type="text/javascript">
  document.write("2+3=");
  document.write(total(2,3))
 </script>
```

```
<p>带返回值的函数调用</p>
 <form>
  <input type="button" onclick="welcome('早上好!')" value="现在是早上">
  <input type="button" onclick="welcome('晚上好!')" value="现在是晚上">
  <input type="button" onclick="TestArgLen(5,7,9)" value="调用函数">
 </form>
</body>
</html>
```




页面加载后，首先执行 body 中的 script 标签里的代码，调用函数 total()，然后打印结果； 点击对应的按钮会调用对应的函数。

## 五、事件驱动及事件处理

通常鼠标或热键的动作我们称之为事件（Event），而由鼠标或热键引发的一连串程序的动作，称之为事件驱动（Event Driver）。
而对事件进行处理程序或函数，我们称之为事件处理程序（Event Handler）。其基本格式与函数全部一样，可以将前面所介绍的所有函数作为事件处理程序。格式如下：

```
function 事件处理名（参数列表）
{
        事件处理语句集;
        ……
}
```

下面是一些常见的HTML事件的列表:

| 事件 | 描述 |
| --- | --- |
| onchange | HTML 元素改变 |
| onclick | 用户点击 HTML 元素 |
| onmouseover | 用户在一个HTML元素上移动鼠 |
| onmouseout | 用户从一个HTML元素上移开鼠 |
| onkeydown | 用户按下键盘按键 |
| onload | 浏览器已完成页面的加载 |

前面几个实例点击 button 的事件用的是 onclick，可以做参考。
除此之外，还有很多事件本篇暂不讨论。

# 六、基于对象的 JavaScript 语言

JavaScript 语言是基于对象的（Object-Based），而不是面向对象的（object-oriented）。之所以说它是一门基于对象的语言，主要是因为它没有提供象抽象、继承、重载等有关面向对象语言的许多功能。而是把其它语言所创建的复杂对象统一起来，从而形成一个非常强大的对象系统。

## A. Javascript 对象的基础知识

### 1. JavaScript 对象的基本结构

JavaScript 中的对象是由属性(properties)和方法(methods)两个基本的元素构成的。

- 属性(properties) 是对象在实施其所需要行为的过程中，实现信息的装载单位，从而与变量相关联；
- 方法(methods) 对象能够按照设计者的意图而被执行，从而与特定的函数相联。

## 2. 引用对象的途径

- 引用 JavaScript 内部对象 由浏览器环境中提供的对象
- 创建新对象
- 利用现存的对象

## 3. 有关对象操作语句

JavaScript 不是一种纯面向对象的语言，但它提供了面向对象语言的一些功能，因此 JavaScript 设计者把它称为“基于对象”而不是面向对象的语言，在 JavaScript 中提供了几个用于操作对象的语句和关键字及运算符。

- 1 for...in 语句 格式如下:

```
for（对象属性名 in 已知对象名）
```

说明： 该语句的功能是用于对已知对象的所有属性进行操作的控制循环。它是将一个已知对象的所有属性反复置给一个变量；而不是使用计数器来实现的,该语句的优点是无需知道对象中属性的个数即可进行操作。 例：下列函数是显示数组中的内容：

```
function showData(object)
{
 for (var i=0; i<30;i++)
     document.write(object[i])
}
```

该函数是通过数组下标顺序值，来访问每个对象的属性，使用这种方式首先必须知道数组的下标值，否则若超出范围，则就会发生错误。 如使用 for...in 语句，则根本不需要知道对象属性的个数，如下：

```
function showData(object){
 for(var prop in object)
  document.write(object[prop]);
}
```

使用该函数时，在循环体中，for 自动将对象的属性取出来，直到最后为此。

- 2 with 语句 使用该语句的意思是：在该语句体内，任何对变量的引用被认为是这个对象的属性，以省一些代码。

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

所有在 with 语句后的花括号中的语句，都是在后面 object 对象的作用域的。

- 3 this 关键字 this 是对当前的引用，在 JavaScript 由于对象的引用是多层次，多方位的，

往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，最后自己已不知道现在引用的那一个对象，
为此 JavaScript 提供了一个用于将对象指定当前对象的语句 this。

- 4 New 运算符 虽然在 JavaScript 中对象的功能已经是非常强大的了。但更强大的是设计人员可以按照需求来创建自己的对象，以满足某一特定的要求。使用 New 运算符可以创建一个新的对象。 其创建对象使用如下格式：

```
Newobject=NEW Object(Parameters table);
```

其中 Newobject 创建的新对象：object 是已经存在的对象； parameters table 参数表； new 是 JavaScript 中的命令语句。
如创建一个日期新对象

```
newData=New Date()
birthday=New Date (December 12.1998)
```

之后就可使 newData、birthday 作为一个新的日期对象了。

## 对象属性的引用

对象属性的引用可由下列三种 方式之一实现：

- a) 使用点（.）运算符

```
university.name="云南省"
university.city="昆明市"
university.date="1999"
```

其中 university 是一个已经存在的对象，name、city、date 是它的三个属性，并通过操 作对其赋值。

- b) 通过对象的下标实现引用

```
university[0]="云南"
university[1]="昆明市"
university[2]="1999"
```

通过数组形式的访问属性，可以使用循环操作获取其值。

```
function showunievsity(object){
for (var j=0;j<2; j++)
document.write(object[j])
}
```

若采用 For...in 则可以不知其属性的个数后就可以实现：

```
function showmy(object){
for (var prop in this)
 docament.write(this[prop]);
}
```

- c) 通过字符串的形式实现

```
university["name"]=“云南”
university["city"]=“昆明市”
university["date"]="1999"
```

## 4. 对象的方法的引用

在 JavaScript 中对象方法的引用是非常简单的。

```
ObjectName.methods()
```

如引用 university 对象中的 showmy（）方法，则可使用：

```
document.write (university.showmy())
```

如引用 math 内部对象中 cos()的方法 则使用：

```
with(Math)
document.write(cos(35));
document.write(cos(80));
```

若不使用 with 则引用时相对要复杂些：

```
document.write(Math.cos(35))
document.write(Math.sin(80)
```

# B. String 对象

String 字符串对象。 声明一个字符串对象最简单,常用的方法就是直接赋值。

## 1. 属性 length

String 对象的 length 属性声明了该字符串中的字符数。

举例：

```
<html>
<body>
 <script type="text/javascript">
  var str="你好!"
  document.write("<p>" + str + "</p>");
  document.write("字符串长度为:");
  document.write(str.length);
  str="hello boy!"
  document.write("<p>" + str + "</p>");
  document.write("字符串长度为:");
  document.write(str.length)
 </script>
</body>
</html>
```

## 2. String 方法

| 方法 | 含义 |
| --- | --- |
| 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。 |

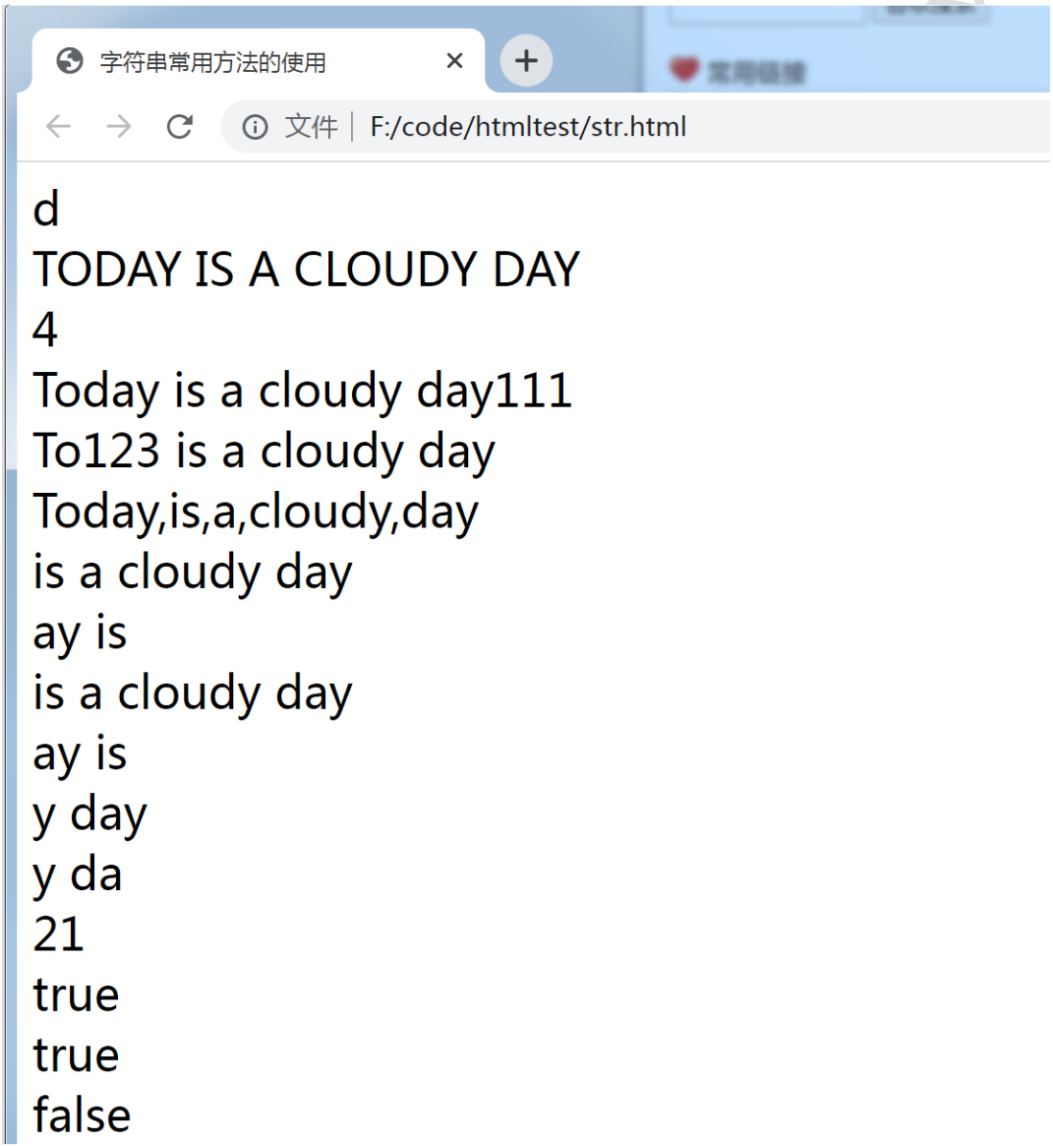
| 方法 | 含义 |
| --- | --- |
| fontsize() | 使用指定的尺寸来显示字符串。 |
| fromCharCode() | 从字符编码创建一个字符串。 |
| indexOf() | 检索字符串。 |
| italics() | 使用斜体显示字符串。 |
| lastIndexOf() | 从后向前搜索字符串。 |
| link() | 将字符串显示为链接。 |
| localeCompare() | 用本地特定的顺序来比较两个字符串。 |
| match() | 找到一个或多个正则表达式的匹配。 |
| replace() | 替换与正则表达式匹配的子串。 |
| 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。 |

需要注意的是，JavaScript 的字符串是不可变的（immutable），String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase() 这样的方法，返回的是全新的字符串，而不是修改原始字符串。

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta >
    <title>字符串常用方法的使用</title>
    <script>
        var s ="Today is a cloudy day";
        //charAt() 获取指定下标位置的字符  下标从0开始
        document.write(s.charAt(2)+"<br/>");
        //toUpperCase() 字母全部转换成大写
        //tiLowerCase()转换成小写
        document.write(s.toUpperCase()+"<br/>");
        //indexOf(substring)找到第一次匹配的元素的下标
        document.write(s.indexOf("y","i")+"<br/>");
        //comcat() 拼接，在原有的字符串后面加上括号里面的内容，不影响原来字符串的内容
        document.write(s.concat("111")+"<br/>");
        //replace() 替换   替换第一个匹配的指定元素为新的指定元素
        document.write(s.replace("day","123")+"<br/>");
        //split() 按照指定字符串或者正则字符串拆分，返回一个数组
        document.write(s.split(" ")+"<br/>");

        //substring()截取指定下标的元素
        //"Today is a cloudy day"
        document.write(s.substring(6)+"<br/>");
        //substring()截取指定下标到指定下标的元素
        //包含3  不包含8  实际截取3-7
        document.write(s.substring(3,8)+"<br/>");
        //slice()
        document.write(s.slice(6)+"<br/>");
        document.write(s.slice(3,8)+"<br/>");
        //slice()可以写负数 即倒数  最后一个从-1开始
        document.write(s.slice(-5)+"<br/>");//y day
        document.write(s.slice(-5,-1)+"<br/>");//y da
        document.write(s.length+"<br>");//长度

        var s1 = "11";
        var s2 = 11;
        var s3 = "11";
        var s4 = "hello";
        document.write(s1==s2);
```

```
        document.write("<br/>");
        document.write(s1==s3);
        document.write("<br/>");
        document.write(s1==s4);

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

执行结果:




## C. Math 对象

| 方法 | 含义 |
| --- | --- |
| bs(x) | 返回 x 的绝对值。 |
| acos(x) | 返回 x 的反余弦值（余弦值等于 x 的角度），用弧度表示。 |
| asin(x) | 返回 x 的反正弦值。 |
| atan(x) | 返回 x 的反正切值。 |
| atan2(x, y) | 返回复平面内点(x, y)对应的复数的幅角，用弧度表示，其值在 -π 到 π 之间。 |
| ceil(x) | 返回大于等于 x 的最小整数。 |
| cos(x) | 返回 x 的余弦。 |
| exp(x) | 返回 e 的 x 次幂 (ex)。 |
| floor(x) | 返回小于等于 x 的最大整数。 |
| log(x) | 返回 x 的自然对数 (ln x)。 |
| max(a, b) | 返回 a, b 中较大的数。 |
| min(a, b) | 返回 a, b 中较小的数。 |
| 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。 |
| sin(x) | 返回 x 的正弦。 |
| sqrt(x) | 返回 x 的平方根。 |
| tan(x) | 返回 x 的正切。 |

```
<html>
<body>
<script type="text/javascript">
document.write("Math.round(7.56)=");
document.write(Math.round(7.56)+"<br>");
document.write("产生 0-1 之间的随机数:");
document.write(Math.random()+"<br>");
document.write("产生 0-10 之间的随机数:");
document.write(Math.round(Math.random()*10)+"<br>");
document.write("求最大值:");
document.write(Math.max(2,40)+"<br>");
```

```
document.write("求最小值:");
document.write(Math.min(2,40)+"<br>")
</script>
</body>
</html>
```

执行结果:

> Math.round(7.56)=8
> 产生 0-1 之间的随机数:0.06364650956023743
> 产生 0-10 之间的随机数:0
> 求最大值:40
> 求最小值:2

## D. Date 对象

Date 对象可以储存任意一个日期，从 0001 年到 9999 年，并且可以精确到毫秒数（1/1000 秒）。

在内部，日期对象是一个整数，它是从 1970 年 1 月 1 日零时整开始计算到日期对象所指的日期的毫秒数。如果所指日期比 1970 年早，则它是一个负数。

所有日期时间，如果不指定时区，都采用“UTC”（世界时）时区，它与“GMT”（格林威治时间）在数值上是一样的。

这个方法使 d 成为日期对象，并且已有初始值：当前时间。如果要自定初始值，可以用

```
1）new Date("month dd,yyyy hh:mm:ss");
2）new Date("month dd,yyyy");
3）new Date(yyyy,mth,dd,hh,mm,ss);
4）new Date(yyyy,mth,dd);
5）new Date(ms);
```

各种参数的含义如下：

```
month:用英文 表示月份名称，从 January 到 December
mth:用整数表示月份，从 0（1 月）到 1 1 （1 2 月）
dd:表示一个 月中的第几天，从 1 到 31
yyyy:四位数表示的年份
hh:小时数，从 0（午夜）到 23（晚 11 点）
mm: 分钟数，从 0 到 59 的整数
```

```
ss:秒数，从 0 到 59 的整数
ms:毫秒数，为大于等于 0 的整数
```

例如：

```
new Date("January 12,2006 22:19:35");
new Date("January 12,2006");
new Date(2006,0,12,22,19,35);
new Date(2006,0,12);
new Date(1137075575000);
```

上面的各种创建形式都表示 2006 年 1 月 12 日这一天。

## 1. Date 属性

YYYY-MM-DDThh:mm:ssTZD
日期或时间。下面解释了其中的成分：

- YYYY - 年 (例如 2011)
- MM - 月 (例如 01 表示 January)
- DD - 天 (例如 08)
- T - 必需的分隔符，若规定时间的话
- hh - 时 (例如 22 表示 10.00pm)
- mm - 分 (例如 55)
- ss - 秒 (例如 03)
- TZD - 时区标识符 (Z 表示祖鲁，也称为格林威治时间)

## 2. Date 方法

常用的 Date 方法，【加粗的是常用的】
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)。
**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)。

## 举例 1：js 获取某年某月的哪些天是周六和周日

```
<html>
<body>
<div id="text"></div>
<script type="text/javascript">
    function time(y,m){
    var tempTime = new Date(y,m,0);
    var time = new Date();
    var saturday = new Array();
    var sunday = new Array();
    for(var i=1;i<=tempTime.getDate();i++){
        var ss = time.setFullYear(y,m-1,i);
        var day = time.getDay();
        if(day == 6){
            saturday.push(i);
        }else if(day == 0){
            sunday.push(i);
        }
    }
```

```
    var text = y+"年"+m+"月份"+"<br />"
                +"周六: "+saturday.toString()+"<br />"
                +"周日: "+sunday.toString();
    document.getElementById("text").innerHTML = text;
}

time(2022,4);
</script>
</body>
</html>
```

> 2022年4月份
> 周六：2,9,16,23,30
> 周日：3,10,17,24

## 实例 2：一个走动的时间

```
<html>
<head>
 <script type="text/javascript">
  var timer = null
  function stop()
  {
   clearTimeout(timer)
  }

  function start()
  {
   var time = new Date()
   var hours = time.getHours()
   var minutes = time.getMinutes()
   minutes=((minutes < 10) ? "0" : "") + minutes
   var seconds = time.getSeconds()
   seconds=((seconds < 10) ? "0" : "") + seconds
   var clock = hours + ":" + minutes + ":" + seconds
   document.forms[0].display.value = clock
   timer = setTimeout("start()",1000)
  }
 </script>
</head>
<body onload="start()" onunload="stop()">
```

```
<form>
<input type="text" name="display" size="20">
</form>
</body>
</html>
```

执行结果:

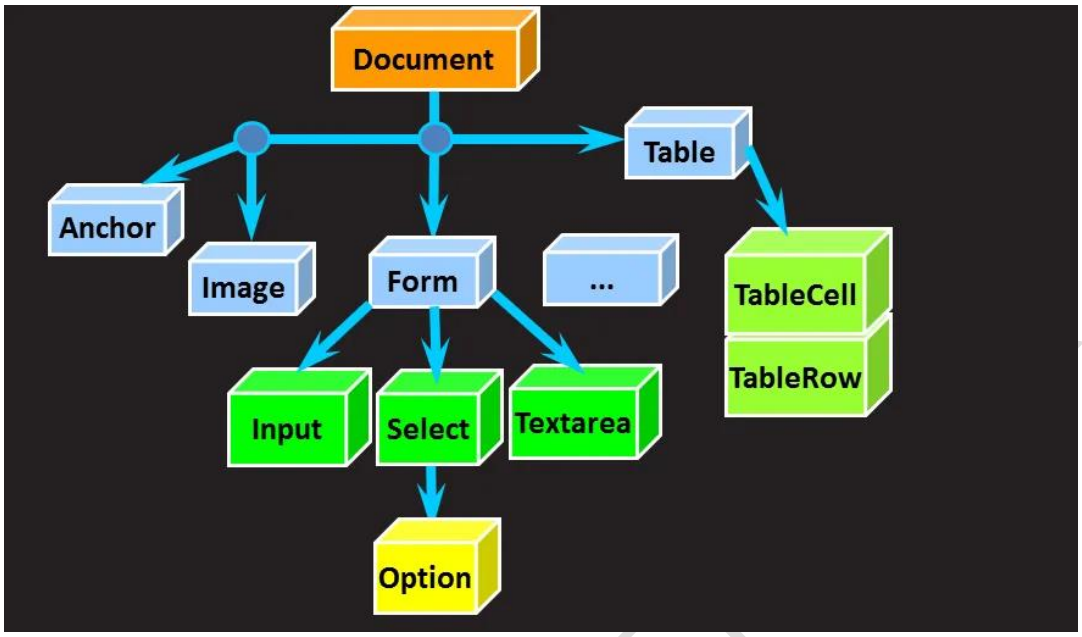

# 七、系统函数

全局函数不属于任何其他对象的属性和方法。但对于 Global 来说，所有在全局作用域中定义的属性和方法，都是 Global 对象的属性。

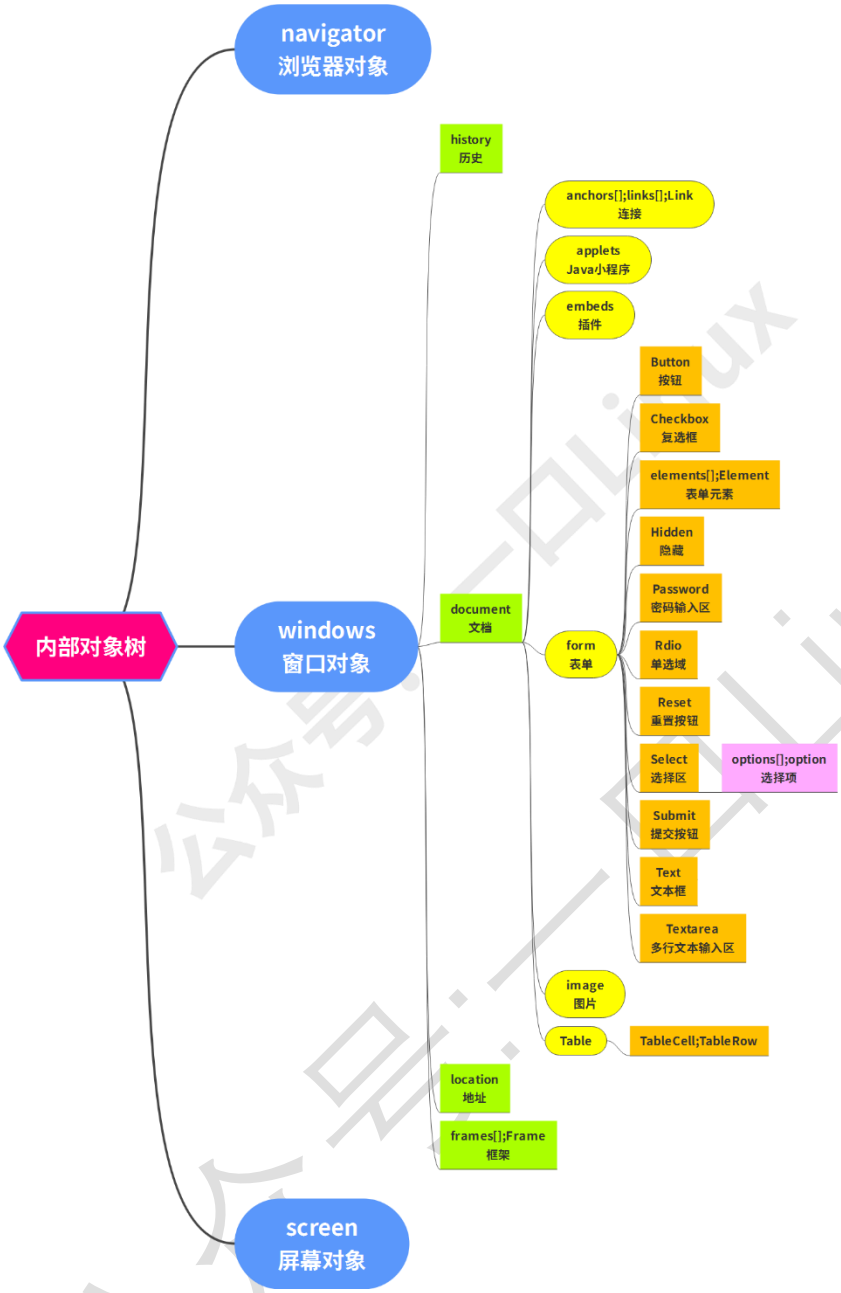
**JavaScript 全局函数**

| 函数 | 描述 |
| --- | --- |
| decodeURI() | 解码某个编码的 URI。 |
| decodeURIComponent() | 解码一个编码的 URI 组件。 |
| encodeURI() | 把字符串编码为 URI。 |
| encodeURIComponent() | 把字符串编码为 URI 组件。 |
| escape() | 对字符串进行编码。 |
| eval() | 计算 JavaScript 字符串，并把它作为脚本代码来执行。 |
| isFinite() | 检查某个值是否为有穷大的数。 |
| isNaN() | 检查某个值是否是数字。 |
| Number() | 把对象的值转换为数字。 |
| parseFloat() | 解析一个字符串并返回一个浮点数。 |
| parseInt() | 解析一个字符串并返回一个整数。 |
| String() | 把对象的值转换为字符串。 |
| unescape() | 对由 escape() 编码的字符串进行解码。 |

# 八、内部对象




常用的内部对象树：




# 1. navigator 浏览器对象

navigator 对象主要反映了当前使用的浏览器的资料

## 1）属性

| 属性 | 描述 |
| --- | --- |
| ppCodeName | 返回浏览器的“码名”,流行的 IE 返回 'Mozilla' |

| 属性 | 描述 |
| --- | --- |
| appName | 返回浏览器名。IE 返回 'Microsoft Internet Explorer' |
| appVersion | 返回浏览器版本，包括了大版本号、小版本号、语言、操作平台等信息 |
| platform | 返回浏览器的操作平台，对于 Windows 9x 上的浏览器，返回 'Win32'（大小写可能有差异） |
| userAgent | 返回以上全部信息。例如，IE5.01 返回 'Mozilla/4.0 (compatible; MSIE 5.01; Windows 98)' |

实例：

```
<html>
<body>
<script type="text/javascript">
document.write("<p>浏览器: ")
document.write(navigator.appName + "</p>")
document.write("<p>版本: ")
document.write(navigator.appVersion + "</p>")
document.write("<p>Code: ")
document.write(navigator.appCodeName + "</p>")
document.write("<p>操作平台: ")
document.write(navigator.platform + "</p>")
document.write("<p>可否使用 cookie: ")
document.write(navigator.cookieEnabled + "</p>")
document.write("<p>浏览器代理头信息: ")
document.write(navigator.userAgent + "</p>")
</script>
</body>
</html>
```

执行结果：

浏览器: Netscape

版本: 5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36

Code: Mozilla

操作平台: Win32

可否使用 cookie: true

浏览器代理头信息: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.99 Safari/537.36

## 2. screen 屏幕对象

screen 对象反映了当前用户的屏幕设置。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| width | 返回屏幕的宽度（像素数） |
| height | 返回屏幕的高度 |
| availWidth | 返回屏幕的可用宽度（除去了一些不自动隐藏的类似任务栏的东西所占用的宽度） |
| availHeight | 返回屏幕的可用高度 |
| colorDepth | 返回当前颜色设置所用的位数 - 1: 黑白; 8: 256 色; 16: 增强色; 24/32: 真彩色 |

## 3. window 窗口对象

window 窗口对象是最大的对象，它描述的是一个浏览器窗口。一般要引用它的属性和方法时，不需要用“window.xxx”这种形式，而直接使用“xxx”。一个框架页面也是一个窗口。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 窗口的名称，由打开它的连接（<a target="...">）或框架页（<frame name="...">）或某一个窗口调用的 open() 方法（见下）决定。一般我们不会用这个属性 |
| status | 指窗口下方的“状态栏”所显示的内容。通过对 status 赋值，可以改变状态栏的显示 |
| opener | 用法 window.opener 返回打开本窗口的窗口对象。注意：返回的是一个窗口对象。如果窗口不是由其他窗口打开的，在 Netscape 中这个属性返回 null; 在 IE 中返回“未定义”(undefined)。undefined 在一定程度上等于 null。 |

| 属性 | 描述 |
| --- | --- |
| | 注意：undefined 不是 JavaScript 常数 |
| self | 指窗口本身，它返回的对象跟 window 对象是一模一样的。最常用的是“self.close()”，放在<a>标记中：“<a href="javascript:self.close()">关闭窗口</a>” |
| arent | 返回窗口所属的框架页对象 |
| top | 返回占据整个浏览器窗口的最顶端的框架页对象 |

## 2）方法

### open

```
open(<URL 字符串>, <窗口名称字符串>, <参数字符串>);
```

<URL 字符串>：描述所打开的窗口打开哪一个网页。如果留空（’’），则不打开任意网 页。
<窗口名称字符串>：描述被打开的窗口的名称（window.name），可以使用’_top’、’_blank’ 等内建名称。这里的名称跟“<a href=”...” target=”...”>”里的“target”属性是一样的。
<参数字符串>：描述被打开的窗口的样貌。如果只需要打开一个普通窗口，该字符串留（’’），如果要指定样貌，就在字符串里写上一到多个参数，参数之间用逗号隔开。
例如：打开一个 400 x 100 的干净的窗口

```
open('','_blank','width=400,height=100,menubar=no,toolbar=no,
location=no,directories=no,status=no,scrollbars=yes,resizable=yes')
```

参数含义：

```
top=# 窗口顶部离开屏幕顶部的像素数
left=# 窗口左端离开屏幕左端的像素数
width=# 窗口的宽度
height=# 窗口的高度
menubar=... 窗口有没有菜单，取值 yes 或 no
toolbar=... 窗口有没有工具条，取值 yes 或 no
location=... 窗口有没有地址栏，取值 yes 或 no
directories=... 窗口有没有连接区，取值 yes 或 no
scrollbars=... 窗口有没有滚动条，取值 yes 或 no
```

```
status=... 窗口有没有状态栏，取值 yes 或 no
resizable=... 窗口给不给调整大小，取值 yes 或 no
```

open() 方法有返回值，返回的就是它打开的窗口对象。 所以，

```
var newWindow =open('','_blank');
```

这样把一个新窗口赋值到“newWindow”变量中，以后通过“newWindow”变量就可以控 制窗口了

## close

close() 关闭一个已打开的窗口

```
window.close() 或 self.close()：关闭本窗口；
```

<窗口对象>.close()：关闭指定的窗口。 如果该窗口有状态栏，调用该方法后浏览器会警告：“网页正在试图关闭窗口，是否关 闭？”然后等待用户选择是否；如果没有状态栏，调用该方法将直接关闭窗口

## blur

使焦点从窗口移走，窗口变为“非活动窗口”。

## focus()

使窗口获得焦点，变为“活动窗口”。

## scrollTo()

用法

```
[<窗口对象>.]scrollTo(x, y);
```

使窗口滚动，使文档从左上角数起的(x, y)点滚动到窗口的左上角。

## scrollBy()

用法

```
[<窗口对象>.]scrollBy(deltaX, deltaY);
```

使窗口向右滚动 deltaX 像素，向下滚动 deltaY 像素。如果取负值，则向相反的方向滚动。

## resizeTo()

用法

```
[<窗口对象>.]resizeTo(width, height);
```

使窗口调整大小到宽 width 像素，高 height 像素。

## resizeBy()

用法

```
[<窗口对象>.]resizeBy(deltaWidth, deltaHeight);
```

使窗口调整大小，宽增大 deltaWidth 像素，高增大 deltaHeight 像素。如果取负值，则减少。

## alert()

用法

```
alert(<字符串>);
```

弹出一个只包含“确定”按钮的对话框，显示<字符串>的内容，整个文档的读取、Script 的运行都会暂停，直到用户按下“确定”。

## confirm()

用法

```
confirm(<字符串>);
```

弹出一个包含“确定”和“取消”按钮的对话框，显示<字符串>的内 容，要求用户做出选择，整个文档的读取、Script 的运行都会暂停。 如果用户按下“确定”， 则返回 true 值，如果按下“取消”，则返回 false 值。

## prompt()

用法

```
prompt(<字符串>[, <初始值>]);
```

弹出一个包含“确认”“取消”和一个文本框的对话框，显示<字符串>的内容，要求用户在文本框输入一些数据，整个文档的读取、Script 的运行都会暂停。

如果用户按下“确认”，则返回文本框里已有的内容，如果用户按下“取消”，则返回 null 值。

如果指定<初始值>，则文本框里会有默认值。 事件

```
onload; onunload; onresize; onblur; onfocus; onerror
```

## 4. history

历史对象指浏览器的浏览历史。

### 1）属性

#### length

历史的项数。JavaScript 所能管到的历史被限制在用浏览器的“前进”“后退”键可以去到的范围。本属性返回的是“前进”和“后退”两个按键之下包含的地址数的和。

### 2）方法

#### back()

后退，跟按下“后退”键是等效的。

#### forward()

前进，跟按下“前进”键是等效的。

go()

用法

```
history.go(x);
```

在历史的范围内去到指定的一个地址。如果 x < 0，则后退 x 个地址，如果 x> 0，则前进 x 个地址，如果 x= =0，则刷新现在打开的网页。
history.go(0) 跟 location.reload()是等效的

## 5. location 地址对象

它描述的是某一个窗口对象所打开的地址。 要表示当前窗口的地址，只需要使用“location”就行了； 若要表示某一个窗口的地址，就使用“<窗口对象>.location”。

### 1） 属性

| 属性 | 描述 |
| --- | --- |
| protocol | 返回地址的协议，取值为 'http:','https:','file:' 等等。 |
| hostname | 返回地址的主机 名 ，例如,一个 “ http://www.microsoft.com/china/ ” 的地址 ， location.hostname == 'www.microsoft.com'。 |
| port | 返回地址的端口号，一般 http 的端口号是 '80'。 |
| host | 返回主机名和端口号，如：'www.a.com:8080'。 |
| pathname | 返回路径名，如“http://www.a.com/b/c.html”，location.pathname = = 'b/c.html'。 |
| search | 返回“?”以及以后的内容，如“http://www.a.com/b/c.asp?selection=3&jumpto=4”，location.search == '?selection=3&jumpto=4'；如果地址里没有“?”，则返回空字符串。 |
| href | 返回以上全部内容，也就是说，返回整个地址。在浏览器的地址栏上怎么显示它就怎么返回。如果想一个窗口对象打开某地址，可以使用“location.href = '...'”，也可以直接用“location= '...'”来达到此目的。 |

### 2） 方法

| 方法 | 描述 |
| --- | --- |
| reload() | 相当于按浏览器上的“刷新”(IE)或“Reload”(Netscape)键。 |
| replace() | 打开一个 URL，并取代历史对象中当前位置的地址。用这个方法打开一个 URL 后， |
| frames[] | Frame 框架对象 |

实例：刷新页面显示当前时间时:分:秒

```
<html>
<head>
<script type="text/javascript">
function refresh()
{
  window.location.reload()
}
</script>
</head>
<body>
<script type="text/javascript">
var d = new Date()
document.write("当前时间是:");
document.write(d.getHours())
document.write(".")
document.write(d.getMinutes())
document.write(".")
document.write(d.getSeconds())
</script>
<form>
 <input type="button" value="刷新页面" onclick="refresh()">
</form>
</body>
</html>
```

## 6. document 文档对象

描述当前窗口或指定窗口对象的文档。它包含了文档从<head>到</body>的内容。

用法

```
document （当前窗口）
```

或

<窗口对象>.document （指定窗口）

| 属性 | 描述 |
| --- | --- |
| cookie | Cookie 是一些数据，存储于你电脑上的文本文件中。Cookie 的作用就是用于解决 "如何记录客户端的用户信息":当用户访问 web 页面时，他的名字可以记录在 cookie 中；在用户下一次访问该页面时，可以在 cookie 中读取用户访问记录。 |
| lastModified | 当前文档的最后修改日期，是一个 Date 对象。 |
| referrer | 如果当前文档是通过点击连接打开的，则 referrer 返回原来的 URL。 |
| title | 指<head>标记里用<title>...</title>定义的文字. |
| fgColor | 指<body>标记的 text 属性所表示的文本颜色。 |
| bgColor | 指<body>标记的 bgcolor 属性所表示的背景颜色。 |
| linkColor | 指<body>标记的 link 属性所表示的连接颜色。 |
| alinkColor | 指<body>标记的 alink 属性所表示的活动连接颜色。 |
| vlinkColor | 指<body>标记的 vlink 属性所表示的已访问连接颜色 |

## 7. anchors[]; links[]; link 连接对象

用法

```
document.anchors[[x]];
document.links[[x]];
<anchorId>;
<linkId>
```

document.anchors

是一个数组，包含了文档中所有锚标记（包含 name 属性的<a>标记），按照在文档中的次序，从 0 开始给每个锚标记定义了一个下标。

document.links 也是一个数组，包含了文档中所有连接标记（包含 href 属性的<a>标记和<map>标记段里的<area>标记），按照在文档中的次序，从 0 开始给每个连接标记定义了一个下标。

如果一个<a>标记既有 name 属性，又有 href 属性，则它既是一个 Anchor 对象，又是一个 link 对象。
anchors 和 links 作为数组，有数组的属性和方法。
单个 Anchor 对象没有属性； 单个 Link 对象的属性见下。
**属性** protocol; hostname; port; host; pathname; hash; search; href 与 location 对象相同。 target 返回/指定连接的目标窗口（字符串），与<a>标记里的 target 属性是一样的。
**事件**

```
onclick; onmouseover; onmouseout; onmousedown; onmouseup
```

## 8. embeds[] 插件对象

它是一个数组，包含了文档中所有的插件（标记）。因为每个插件的不同，每个 Embed 对象也有不同的属性和方法。

## 9. forms[]; Form 表单对象

document.forms[] 是一个数组，包含了文档中所有的表单（<form>）。
要引用单个表单，可以用 document.forms[x]，但是一般来说，人们都会这样做：在<form>标记中加上“name="..."”属性，那么直接用“document.<表单名>”就可以引用了。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 返回表单的名称，也就是 属性。 |
| action | 返回/设定表单的提交地址，也就是属性。 |
| method | 返回/设定表单的提交方法，也就是属性。 |
| target | 返回/设定表单提交后返回的窗口，也就是属性。 |
| encoding | 返回/设定表单提交内容的编码方式，也就是属性。 |

| 属性 | 描述 |
| --- | --- |
| length | 返回该表单所含元素的数目。 |

## 2）方法

| 方法 | 描述 |
| --- | --- |
| reset() | 重置表单。这与按下“重置”按钮是一样的。 |
| submit() | 提交表单。这与按下“提交”按钮是一样的。 |

## 3）事件

```
onreset
onsubmit
```

## 4）实例：document 操作指定 form

```
<html>
<head>
<script type="text/javascript">
 function formReset()
 {
  var x=document.forms.myForm;
  x.reset();
  alert("表单已经重置");
 }
 function formSubmit()
 {
  var x=document.forms.myForm;
  x.submit();
  alert("表单已经提交")
 }
</script>
</head>
<body>
<form name="myForm">
 <p>控制表单的提交和重置</p>
 <input type="text" size="20"><br>
```

```
 <input type="text" size="20"><br>
 <br>
 <input type="button" onclick="formReset()" value="重置">
 <input type="button" onclick="formSubmit()" value="提交">
</form>
</body>
</html>
```

## 10. Button 按钮对象

由“<input type="button">”指定。
引用一个 Button 对象，可以使用“<文档对象>.<表单对象>.<按钮名称>”。<按钮名称>指在<input>标记中的“name="..."”属性的值。引用任意表单元素都可以用这种方法。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用<input name="...">指定的元素名称。 |
| value | 返回/设定用<input value="...">指定的元素的值。 |
| form | 返回包含本元素的表单对象。 |

### 2）方法

| 方法 | 描述 |
| --- | --- |
| blur() | 从对象中移走焦点。 |
| focus() | 让对象获得焦点。 |
| click() | 模拟鼠标点击该对象。 |

### 3）事件

```
onclick; onmousedown; onmouseup
```

## 11. Checkbox 复选框对象

由“<input type="checkbox">”指定。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用<input name="...">指定的元素名称 |
| value | 返回/设定用<input value="...">指定的元素的值。 |
| form | 返回包含本元素的表单对象。 |
| checked | 返回/设定该复选框对象是否被选中。这是一个布尔值。 |
| defaultChecked | 返回/设定该复选框对象默认是否被选中。这是一个布尔值 |

### 2）方法

| 方法 | 描述 |
| --- | --- |
| blur() | 从对象中移走焦点。 |
| focus() | 让对象获得焦点。 |
| click() | 模拟鼠标点击该对象。 |

### 3）事件

```
onclick
```

### 实例：document 操作指定复选框

```
<html>
<head>
<script type="text/javascript">
function check()
{
/*可以通过form的名字查找，也可以通过数组forms[]查找，根据form顺序，下标依次0、1、2*/
 className=document.forms.myForm.className
 classSec=document.forms[0].classSec
 txt=""
 for (i=0;i<className.length;++i)
 {
  if (className[i].checked)
  {
   txt=txt + className[i].value + ","
  }
  if(txt != "")
  {
   classSec.value="选择的课程:" + txt
  }else{
   classSec.value="没有选择课程"
  }

 }
}
</script>
</head>
<body>
<form name="myForm">
选择你要培训的课程<br><br>
<input type="checkbox" name="className" value="Linux">Linux 课程<br>
<input type="checkbox" name="className" value="Arm">ARM 课程<br><br>
<input type="button" name="test" onclick="check()" value="我要培训">
<br><br>
<input type="text" name="classSec" size="50">
</form>
</body>
</html>
```

## 12. elements[]; Element 表单元素对象

<表单对象>.elements 是一个数组，包含了该表单所有的对象。一般我们不用该数组，而直接引用各个具体的对象。

## 13. Hidden 隐藏对象

由“<input type="hidden">”指定。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用<input name="...">指定的元素名称。 |
| value | 返回/设定用<input value="...">指定的元素的值。 |
| form | 返回包含本元素的表单对象 |

## 14. Password 密码输入区对象

由“<input type="password">”指定。

### 1)属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用<input name="...">指定的元素名称。 |
| value | 返回/设定密码输入区当前的值。 |
| defaultValue | 返回用<input value="...">指定的默认值。 |
| form | 返回包含本元素的表单对象。 |

### 2)方法

| 方法 | 描述 |
| --- | --- |
| blur() | 从对象中移走焦点。 |

| 方法 | 描述 |
| --- | --- |
| focus() | 让对象获得焦点。 |
| select() | 选中密码输入区里全部文本。 |

### 3)事件

onchange

## 15. Radio 单选域对象

由“<input type="radio">”指定。一组 Radio 对象有共同的名称（name 属性），这样的话，document.formName.radioName 就成了一个数组。要访问单个 Radio 对象就要用：document.formName.radioName[x]。

### 1）属性

单个 Radio 对象的属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用 指定的元素名称。 |
| value | 返回/设定用 指定的元素的值。 |
| form | 返回包含本元素的表单对象。 |
| checked | 返回/设定该单选域对象是否被选中。这是一个布尔值。 |
| defaultChecked | 返回/设定该对象默认是否被选中。这是一个布尔值。 |

### 2）方法

| 方法 | 描述 |
| --- | --- |
| blur() | 从对象中移走焦点。 |

| 方法 | 描述 |
| --- | --- |
| focus() | 让对象获得焦点。 |
| click() | 模拟鼠标点击该对象。 |

### 3）事件

`onclick`

## 16. Reset 重置按钮对象

由“`<input type="reset">`”指定。
因为 Reset 也是按钮，所以也有 Button 对象的属性和方法。至于“onclick”事件，一般用 Form 对象的 onreset 代替。

## 17. Select 选择区（下拉菜单、列表）对象

由“`<select>`”指定。

### 1）属性

| 属性 | 描述 |
| --- | --- |
| name | 返回/设定用[redacted]指定的元素名称。 |
| length | 返回 Select 对象下选项的数目。 |
| selectedIndex | 返回被选中的选项的下标。这个下标就是在 options[] 数组中该选项的位置。如果 Select 对象允许多项选择，则返回第一个被选中的选项的下标。 |
| form | 返回包含本元素的表单对象。 |

### 2）方法

| 方法 | 描述 |
| --- | --- |
| blur() | 从对象中移走焦点。 |
| focus() | 让对象获得焦点。 |

### 3）事件

`onchange`

## 18. Select 的 options[]; Option 选择项对象

options[] 是一个数组，包含了在同一个 Select 对象下的 Option 对象。Option 对象由“<select>”下的“<options>”指定。

### 1） options[] 数组的属性

length; selectedIndex 与所属 Select 对象的同名属性相同。

### 2） 单个 Option 对象的属性

| 属性 | 描述 |
| --- | --- |
| text | 返回/指定 Option 对象所显示的文本 |
| value | 返回/指定 Option 对象的值，与<options value="...">一致。 |
| index | 返回该 Option 对象的下标。对此并没有什么好说，因为要指定特定的一个 Option 对象，都要先知道该对象的下标。 |
| selected | 返回/指定该对象是否被选中。通过指定 true 或者 false，可以动态的改变选中项。 |
| defaultSelected | 返回该对象默认是否被选中。true / false。 |

### 3） 实例：document 操作下拉菜单选项

```
<html>
<head>
<script type="text/javascript">
function put()
{
 txt=document.forms[0].myClass.options[document.forms[0].myClass.selectedIndex].tex
t
 /*如果选中的是最后提示选项则不现实内容*/
 if(document.forms[0].myClass.selectedIndex != document.forms[0].myClass.options.in
validopt.index)
 {
 document.forms[0].classSec.value=document.forms[0].myClass.options[document.forms[
0].myClass.selectedIndex].index + ":" + txt
 }else{
  document.forms[0].classSec.value=""
 }
}
</script>
</head>
<body>
<form name="myForm">
选择你要培训的课程<br><br>

<select name="myClass" onchange="put()">
<option>C 语言</option>
<option>ARM</option>
<option>Linux</option>
<option>数据结构</option>
<option selected=true name="invalidopt">点击选择课程</option>
</select>

<br><br>
你选择的课程是: <input type="text" name="classSec" size="20">
</form>

</body>
</html>
```

## 19. Submit 提交按钮对象

由“<input type="submit">指定。因为 Submit 也是按钮，所以也有 Button 对象的属性和方法。至于“onclick”事件，一般用 Form 对象的 onsubmit 代替。

## 20. Text 文本框对象

由“<input type="text">”指定。Password 对象也是 Text 对象的一种，所以 Password 对象所有的属性、方法和事件，Text 对象都有。

## 21. Textarea 多行文本输入区对象

由“<textarea>”指定。Textarea 对象所有的属性、方法和事件和 Text 对象相同，也就是跟 Password 对象一样。

## 22. images[]; Image 图片对象

document.images[] 是一个数组，包含了文档中所有的图片（<img>）。
要引用单个图片，可以用 document.images[x]。
如果某图片包含“name”属性，也就是用“<img name="...">”
这种格式定义了一幅图片，就可以使用“document.images['...']”这种方法来引用图片。
在 IE 中，如果某图片包含 ID 属性，也就是用“<img id="...">”这种格式定义了一幅图片，就可以直接使用“<imageID>”来引用图片。

### 1）单个 Image 对象的属性

```
name; src; width; height; vspace; hspace; border
```

这些属性跟<img>标记里的同名属性是一样的。这些属性最有用的就是 src 了，通过对 src 属性赋值，可以实时的更改图片。

### 2）事件

```
onclick
```

# 23. 综合项目：创建登录注册页面，检查用户名密码

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

<title>物联网仓储管理系统 - 登陆</title>

<script language="javascript">
function isValidate(form)
{
 username=form.username.value;
 userpass=form.userpass.value;
   alert("用户名不能为空,请输入两位以上用户名");
 if(username == "")
 {
  alert("用户名不能为空,请输入两位以上用户名");
  form.username.focus();
  return false;
 }

 if(!minLength(username,2))
 {
  alert("用户名长度小于 2 位！");
  form.username.focus();
  return false;
 }

 if(!maxLength(username,10))
 {
  alert("用户名长度大于 10 位！");
  form.username.focus();
  return false;
 }
 if(userpass == "")
 {
  alert("密码不能为空");
  form.userpass.focus();
  return false;
 }
 if(!minLength(userpass,2))
 {
  alert("口令长度小于 2 位！");
```

```
  form.userpass.focus();
  return false;
 }
 if(!maxLength(userpass,20))
 {
  alert("口令长度大于 20 位！ ");
  form.userpass.focus();
  return false;
 }
 return true;
 }
 function minLength(str,length)
 {
  if(str.length>=length)
   return true;
  else
   return false;
 }
 function maxLength(str,length)
 {
  if(str.length<=length)
   return true;
  else
   return false;
 }
</script>
</head>
<body>

<table width="100%" border="0" cellspacing="0" cellpadding="0">

 <form onsubmit="return isValidate(myform)" name="myform" method="post"  action="cg
i-bin/login.cgi">
 <tr>
  <td width="92" class="txt_bt">name:</td>
  <td ><input name="username" type="text" id="username" value="root"></td>
 </tr>
 <tr>
  <td>password</td>
  <td><input name="userpass" type="password" id="userpass" value="root"></td>
 </tr>

 <tr>
  <td height="35">登录</td>
```

```
  <td><input type="submit" name="register"  value="提交" ></td>
 </tr>
 </form>
</table>
</html>
```