## JS是什么

Js是一门运行在客户端的编程语言

## JavaScript 对大小写敏感

## 空格

JavaScript 会忽略多余的空格。可以向脚本添加空格，来提高其可读性。

var name="Hello";

var name = "Hello";

上面两行代码等效。

## 对代码行进行折行

可以在文本字符串中使用反斜杠对代码行进行换行。下面的例子会正确地显示：

document.write("Hello \

World!");

不能像这样折行：

document.write \

("Hello World!");

## for...in 循环

for (对象中的变量)

{

要执行的代码

}

注：for...in 循环中的代码块将针对每个属性执行一次。

实例

var person={fname:"Bill",lname:"Gates",age:56};

for (x in person)

{

txt=txt + person[x];

}

## 判断语句（if else）

If(条件)

{条件成立时执行的代码}

else{条件不成立时执行的代码}

## JavaScript 表单验证

JS可用来在数据被送往服务器前对 HTML 表单中的这些输入数据进行验证。

# JS组成部分

（1）ECMAScript：ECMAScript不是一门语言，而是一个标准。符合这个标准的比较常见的有：JavaScript、Action Script（Flash中用的语言）

（2）DOM（Document Object Model 的简称）：JavaScript操作网页上元素的API

（3）BOM（Browser Object Model 的简称）：JavaScript操作浏览器的部分功能的API

# JS引入方式

## <head> 或 <body> 中的 JavaScript

您可以在 HTML 文档中放入不限数量的脚本。

脚本可位于 HTML 的 <body> 或 <head> 部分中，或者同时存在于两个部分中。

通常的做法是把函数放入 <head> 部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，不会干扰页面的内容。

## 外部的 JavaScript

可以把脚本保存到外部文件中。外部文件通常包含被多个网页使用的代码。

外部 JavaScript 文件的文件扩展名是 .js。

如需使用外部文件，请在 <script> 标签的 "src" 属性中设置该 .js 文件

<!DOCTYPE html>

<html>

<body>

<script src="myScript.js"></script>

</body>

</html>

在 <head> 或 <body> 中引用脚本文件都是可以的。实际运行效果与在 <script> 标签中编写脚本完全一致。

注：外部脚本不能包含 <script> 标签。

# 弱类型脚本语言（解释型语言，解析执行与编译执行）

## 解析执行与编译执行

编译执行：把代码编译成CPU认识的语言(文件)，然后整体的执行。

解析执行：一行一行解析，解析一行执行一行。

## 弱类型脚本语言

脚本语言是：不需要编译，直接运行时边解析边执行的语言。弥补编译语言的不足而存在的，作为补充语言。

弱类型语言：简单理解定义一个变量，可以有多种数据类型。（var temp）

## JavaScript 是脚本语言（解释执行）

Js代码不通过编译，直接通过js引擎执行代码，浏览器会在读取代码时，逐行地执行脚本代码。而对于传统编程来说，会在执行前对所有代码进行编译。

# 操作符

## 操作符种类

1. 算数运算符（+—\*/...）
   1. 一元运算符：正号、负号、++、--、平方等一个变量就能运算
   2. 二元运算符：+-\*/%等两个变量才能运算
   3. 三元运算符： 值1？值2：值3；

语法：条件表达式?结果1:结果2

含义：问号前面的位置是判断的条件，判断结果为boolean型，为true时执行结果1，为false时执行结果2。

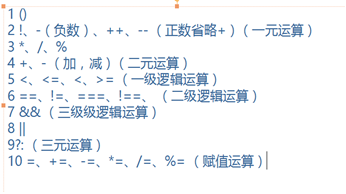
1. 逻辑运算符（ ||&& ! ）（或且非）
2. 比较运算符（<、>、==、>=...）
3. 赋值运算符（=、+=、-=、\*=、/=、%=）

等号运算符

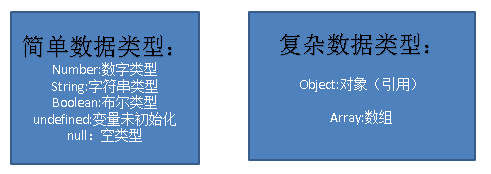
|  |  |
| --- | --- |
| **运算符** | **含义** |
| == | 等于，比较的是内容 |
| === | 全等，比较的是内容和数据类型 |
| != | 不等于，判断的是内容 |
| !== | 不全等于，判断的是内容和数据类型 |

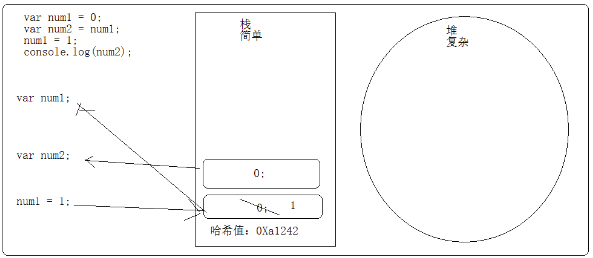
## 操作符之间的优先级（高到低）

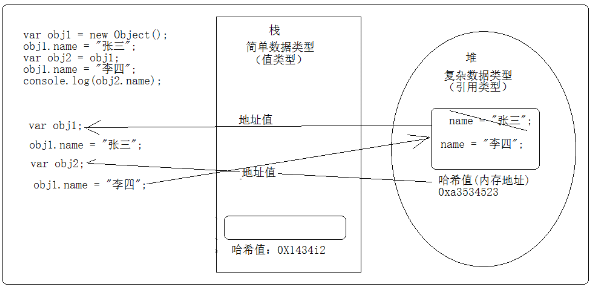
算术操作符>比较操作符>逻辑操作符> “=”赋值符号



# JS数据类型

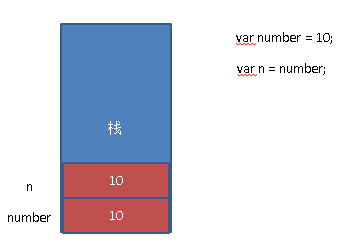






判断数据类型用typeof()

## 数学类型： Number



1: 最基本的数据类型   
2: 不区分整型数值和浮点型数值

浮点数值的最高精度是 17 位小数，但在进行算术计算时其精确度远远不如整数

var result = 0.1 + 0.2; // 结果不是 0.3，而是：0.30000000000000004

console.log(0.07 \* 100);

永远不要测试某个特定的浮点数值(不要判断两个浮点数是否相等)

3: 由于内存的限制，ECMAScript 并不能保存世界上所有的数值

最小值：Number.MIN\_VALUE，这个值为： 5e-324（±5 乘以10的-324次方）

最大值：Number.MAX\_VALUE，这个值为： 1.7976931348623157e+308（±1.7976931348623157乘以10的308次方）

无穷大：Infinity

无穷小：-Infinity

4：包含十六进制数据，以 0x开头 0到9之间的数字，a(A)-f(F)之间字母构成。 a-f对应的数字是10-15  
5： 八进制直接以数字0开始，有0-7之间的数字组成。

6：NaN

a) NaN 非数值（Not a Number的简写）

console.log(“abc”/18); //结果是NaN

Undefined和任何数值计算为NaN;

NaN 与任何值都不相等，包括 NaN 本身

b) isNaN() :任何不能被转换为数值的值都会导致这个函数返回 true

（isNaN译为是否符合一个标准，什么标准呢？不是一个数字的标 准，如果符合了那么就不是一个数字，不符合就是一个数字）

isNaN(NaN);// true

isNaN(“blue”); // true

isNaN(123); // false

## 字符串类型：String

1.字符串由单引号或双引号括起  
2.例如单独一个字母也被称为字符串（例如：‘a’）  
3.转义字符 \" \'

4.字符串不可变

ECMAScript 中的字符串是不可变的，也就是说，字符串一旦创建，它们的值就不能改变。在内存中不会立刻消失，只能二次赋值，原有的字符在一定时间内被垃圾回收器回收。

要改变某个变量保存的字符串，首先要销毁原来的字符串，然后再用另一个包含新值的字符串填充该变量

例如：var str = "123"; str = str + "abc";

## 布尔类型：Boolean

只有2个值一个是true, 一个是false.   实际运算中true=1,false=0

虽然Boolean 类型的字面值只有两个，但 ECMAScript 中所有类型的值都有与这两个 Boolean 值等价的值

1. true

true、除0数字、“something”、Object(任何对象)为true

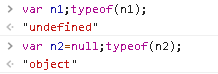
2、false

false、0 、“”、undefined 、null为false

3、if判断时会把（）内的值强行转换成boolean类型进行判断

## JavaScript 特殊类型（null、undefined）

区别：

（1）

（2）null == undefined的结果(true)

null ===undefined的结果(false)

（3）任何数据类型和undefined运算都是NaN;

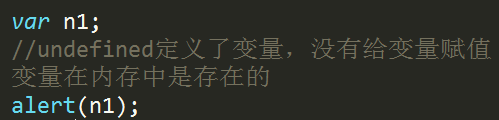
任何值和null运算，null可看做0运算。

例如：和数字运算时，10 + null结果为：10；10 + undefined结果为：NaN。

1、取值null（空）    含义：Javascript中的关键字，它表示一个特殊值。通常用来描述“空值”。

http://images2017.cnblogs.com/blog/1142681/201710/1142681-20171031165433590-662227581.png

2、取值undefined（未定义）   含义：定义了变量，但变量未初始化（变量没有赋值），表示值的空缺。在EcmaScript3中，undefined是可读写的变量，可以给它赋任何值。在EcmaScript5中将这个错误进行了修改。只有undefined这一个值。



## 复杂数据类型

Object、function、Array、Date、RegExp、Error.......

# 数据类型转换

## 数字类型转字符串

强制类型转换（String(),变量.tostring()）

注意： Null和undefined无toString方法，但可以使用String()。





数值类型的toString()，可以携带一个参数，输出对应进制的值

var num = 10;

console.log(num.toString()); //"10" 默认是10进制

console.log(num.toString(10));//"10"

console.log(num.toString(8)); //"12"

console.log(num.toString(16));//"a"

console.log(num.toString(2)); //"1010"

## 字符串转数字类型

强制类型转换(Number(),parseInt(),parseFloat())

空字符串parseInt()和parseFloat()返回NaN，Number("")返回0

◆Number



★数字类型的字符串，转换之后得到的数字。

★非数字字符串，转换之后得到是NaN。

★小数类型的字符串，转换之后得到的是原数字。

var num2 = Number(“18.99”); 结果为18.99数字型

◆parseInt



parseInt()可以传递两个参数，第一个参数是要转换的字符串，第二个参数是要转换的进制

var num4 = parseInt("0xA"); //返回10

var num6 = parseInt("A"); //返回什么？ NaN

var num1 = parseInt("A",16); //A按16进制解析是10

var num2 = parseInt("10",10); //10按10进制解析10

var num3 = parseInt("10",16); //10按16进制解析16

var num4 = parseInt("10",8); //10按8进制解析是8

var num5 = parseInt("10",2); //10按2进制解析是2

var num6 = parseInt("10",3); //10按3进制解析是3

★整数数字类型的字符串，转换之后得到的整数数字。

★数字开头的字符串，转换之后得到的是前边的数字。

★非数字开头的字符串，转换之后得到的是NaN。

★小数类型的字符串，转换之后取整。

123 = parseInt(“123.123aaaa”);

◆parseFloat

parseFloat()和parseInt非常相似，不同之处在于：

parseFloat会解析第一个. 遇到第二个.或者非数字结束

parseFloat不支持第二个参数，只能解析10进制数

如果解析的内容里只有整数，解析成整数

★整数数字类型的字符串，转换之后得到的整数数字。

★数字开头的字符串，转换之后得到的是前边的数字。

★非数字开头的字符串，转换之后得到的是NaN。

★小数类型的字符串，转换之后得到的是原数字。

123.123 = parseFloat(“123.123aaaa”);

## 将其他数据类型转换为布尔类型

强制类型转换 :Boolean()



★数字和字符串转完之后为true。

★undefined、null、0转完之后为false.

## 隐式数据类型转换

在运算过程中，程序自己进行的数据转换(程序员没有操作)

# JS变量

## 声明变量

Var 变量名

命名规则：1、必须使用字母、下划线”\_”、或美元符号”$”开始。2、然后可以使用任意多个英文字母、数字、下划线或美元符号组成，但变量名长度不能超过255个字符。3、在JS中区分大小写。4、变量可以不声明，直接使用，但不规范，需要先声明，后使用。5、不能使用脚本语言中保留的关键字及保留符号作为变量名。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

var pi=3.14;

var name="Bill Gates";

var answer='Yes I am!';

## 规范(建议遵守)

变量的名称要有意义

变量的命名遵守驼峰命名法，首字母小写,第二个单词的首字母大写，例如：userName

## 一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

var name="Gates", age=56, job="CEO";

声明也可横跨多行：

var name="Gates",

age=56,

job="CEO";

## Value = undefined

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

var carname;

## 重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

var carname="Volvo";

var carname;

## 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：

实例

var x // x 为 undefined

var x = 6; // x 为数字

var x = "Bill"; // x 为字符串

## 局部JS变量

在 JavaScript 函数内部声明的变量（使用 var）是**局部**变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。

您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。

只要函数运行完毕，本地变量就会被删除。

## 全局 JS变量

在函数外声明的变量是**全局**变量，网页上的所有脚本和函数都能访问它。

## JS变量的生存期

JavaScript 变量的生命期从它们被声明的时间开始。

局部变量会在函数运行以后被删除。

全局变量会在页面关闭后被删除。

## 向未声明的JS变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

这条语句：

carname="Volvo";

将声明一个**全局**变量 carname，即使它在函数内执行。

# JS对象

## 什么是对象

生活中的对象，一个车、一个手机

对象具有特征和行为

## 面向对象和基于对象

1、面向对象：可以创建自定义的类型、很好的支持继承和多态。面向对象的语言c++/java/c#...

面向对象的特征：封装、继承、多态

万物皆对象：世间的一切事物都可以用对象来描述

2、基于对象：无法创建自定义的类型、不能很好的支持继承和多态。基于对象的语言JavaScript

JavaScript 提供多个内建对象，比如 String、Date、Array 等等。

对象只是带有属性和方法的特殊数据类型。

## 创建 JavaScript 对象

JavaScript 中的几乎所有事务都是对象：字符串、数字、数组、日期、函数，等等。

你也可以创建自己的对象。

### 创建空白对象

var obj = new Object();

本例创建名为 "person" 的对象，并为其添加了四个属性：

### 创建自定义对象

#### This

1. this只出现在函数中。
2. 谁调用函数，this就指的是谁。
3. new People(); People中的this代指被创建的对象实例。

#### new

1.开辟内存空间，存储新创建的对象（ new Object() ）

2.把this设置为当前对象

3.执行内部代码，设置对象属性和方法

4.返回新创建的对象

#### 实例

var person=new Object();

person.firstname="Bill";

person.lastname="Gates";

person.attack = function () {

console.log("攻击了水晶！");

}

#### **对象字面量**

var o = {

name : "zs",

age : 18,

sex : true,

sayHi : function() {

console.log(this.name);

}

};

o.sayHi(); //对象o调用sayHi()方法，所以sayHi()中的this是对象o

#### 对象构造器

//创建一个构造函数

function Student(name){

//构造函数中的对象指的是this。

this.name = name;

this.sayHi = function () {

console.log(this.name+"说：大家好！");

}

}

一旦您有了对象构造器，就可以创建新的对象实例

var stu1 = new Student("王五");

stu1.sayHi();

**把方法添加到 JavaScript 对象**

方法只不过是附加在对象上的函数。

在**构造器函数**内部定义对象的方法：

function person(firstname,lastname,age,eyecolor)

{

this.firstname=firstname;

this.lastname=lastname;

this.changeName=changeName;

function changeName(name)

{

this.lastname=name;

}

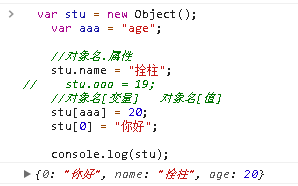
}

## 访问对象的属性

对象属性有两种寻址方式：

1、objectName.propertyName

2、objectName[“propertyName”]



本例使用 String 对象的 length 属性来查找字符串的长度：

var message="Hello World!";

var x=message.length;

// var x=message[“length”];

在以上代码执行后，x 的值是：

12

## 访问对象的方法

objectName.methodName()

这个例子使用 String 对象的 toUpperCase() 方法来把文本转换为大写：

var message="Hello world!";

var x=message.toUpperCase();

在以上代码执行后，x 的值是：

HELLO WORLD!

## JavaScript 类

JavaScript 是面向对象的语言，但 JavaScript 不使用类。

在 JavaScript 中，不会创建类，也不会通过类来创建对象（就像在其他面向对象的语言中那样）。

JavaScript 基于 prototype，而不是基于类的。

## 基本包装类型

为了方便操作基本数据类型，JavaScript还提供了三个特殊的引用类型：String/Number/Boolean

例如：

var s1 = "zhangsan";

var s2 = s1.substring(5);

s1是基本类型，基本类型是没有方法的

当调用s1.substring(5)的时候，先把s1包装成String类型的临时对象，再调用substring方法，最后销毁临时对象

相当于：

var s1 = new String("zhangsan");

var s2 = s1.substring(5);

s1 = null;

### Number 对象

Number 对象是原始数值的包装对象。

var myNum=new Number(value);

var myNum=Number(value);

当 Number() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 Number 对象。如果不用 new 运算符，把 Number() 作为一个函数来调用，它将把自己的参数转换成一个原始的数值，并且返回这个值（如果转换失败，则返回 NaN）。

#### Number 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_number.asp) | 返回对创建此对象的 Number 函数的引用。 |
| [MAX\_VALUE](http://www.w3school.com.cn/jsref/jsref_max_value.asp) | 可表示的最大的数。 |
| [MIN\_VALUE](http://www.w3school.com.cn/jsref/jsref_min_value.asp) | 可表示的最小的数。 |
| [NaN](http://www.w3school.com.cn/jsref/jsref_nan_number.asp) | 非数字值。 |
| [NEGATIVE\_INFINITY](http://www.w3school.com.cn/jsref/jsref_negative_infinity.asp) | 负无穷大，溢出时返回该值。 |
| [POSITIVE\_INFINITY](http://www.w3school.com.cn/jsref/jsref_positive_infinity.asp) | 正无穷大，溢出时返回该值。 |
| prototype | 使您有能力向对象添加属性和方法。 |

#### Number 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| [toString](http://www.w3school.com.cn/jsref/jsref_tostring_number.asp) | 把数字转换为字符串，使用指定的基数。 |
| [toLocaleString](http://www.w3school.com.cn/jsref/jsref_tolocalestring_number.asp) | 把数字转换为字符串，使用本地数字格式顺序。 |
| [toFixed](http://www.w3school.com.cn/jsref/jsref_tofixed.asp) | 把数字转换为字符串，结果的小数点后有指定位数的数字。 |
| [toExponential](http://www.w3school.com.cn/jsref/jsref_toexponential.asp) | 把对象的值转换为指数计数法。 |
| [toPrecision](http://www.w3school.com.cn/jsref/jsref_toprecision.asp) | 把数字格式化为指定的长度。 |
| [valueOf](http://www.w3school.com.cn/jsref/jsref_valueof_number.asp) | 返回一个 Number 对象的基本数字值。 |

### String 对象

new String(s);

String(s);

当 String() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 *s* 或 *s* 的字符串表示。

当不用 new 运算符调用 String() 时，它只把 *s* 转换成原始的字符串，并返回转换后的值。

#### String 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| constructor | 对创建该对象的函数的引用 |
| [length](http://www.w3school.com.cn/jsref/jsref_length_string.asp) | 字符串的长度 |
| prototype | 允许您向对象添加属性和方法 |

#### String 对象方法

字符串所有的方法，都不会修改字符串本身(字符串是不可变的)，操作完成会返回一个新的字符串

|  |  |
| --- | --- |
| 方法 | 描述 |
| [anchor()](http://www.w3school.com.cn/jsref/jsref_anchor.asp) | 创建 HTML 锚。 |
| [big()](http://www.w3school.com.cn/jsref/jsref_big.asp) | 用大号字体显示字符串。 |
| [blink()](http://www.w3school.com.cn/jsref/jsref_blink.asp) | 显示闪动字符串。 |
| [bold()](http://www.w3school.com.cn/jsref/jsref_bold.asp) | 使用粗体显示字符串。 |
| [charAt()](http://www.w3school.com.cn/jsref/jsref_charAt.asp) | 返回在指定位置的字符。 |
| [charCodeAt()](http://www.w3school.com.cn/jsref/jsref_charCodeAt.asp) | 返回在指定的位置的字符的 Unicode 编码。 |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_string.asp) | 连接字符串。 |
| [fixed()](http://www.w3school.com.cn/jsref/jsref_fixed.asp) | 以打字机文本显示字符串。 |
| [fontcolor()](http://www.w3school.com.cn/jsref/jsref_fontcolor.asp) | 使用指定的颜色来显示字符串。 |
| [fontsize()](http://www.w3school.com.cn/jsref/jsref_fontsize.asp) | 使用指定的尺寸来显示字符串。 |
| [fromCharCode()](http://www.w3school.com.cn/jsref/jsref_fromCharCode.asp) | 从字符编码创建一个字符串。 |
| [indexOf()](http://www.w3school.com.cn/jsref/jsref_indexOf.asp) | 检索字符串。 |
| [italics()](http://www.w3school.com.cn/jsref/jsref_italics.asp) | 使用斜体显示字符串。 |
| [lastIndexOf()](http://www.w3school.com.cn/jsref/jsref_lastIndexOf.asp) | 从后向前搜索字符串。 |
| [link()](http://www.w3school.com.cn/jsref/jsref_link.asp) | 将字符串显示为链接。 |
| [localeCompare()](http://www.w3school.com.cn/jsref/jsref_localeCompare.asp) | 用本地特定的顺序来比较两个字符串。 |
| [match()](http://www.w3school.com.cn/jsref/jsref_match.asp) | 找到一个或多个正则表达式的匹配。 |
| [replace()](http://www.w3school.com.cn/jsref/jsref_replace.asp) | 替换与正则表达式匹配的子串。 |
| [search()](http://www.w3school.com.cn/jsref/jsref_search.asp) | 检索与正则表达式相匹配的值。 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_string.asp) | 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()](http://www.w3school.com.cn/jsref/jsref_small.asp) | 使用小字号来显示字符串。 |
| [split()](http://www.w3school.com.cn/jsref/jsref_split.asp) | 把字符串分割为字符串数组。 |
| [strike()](http://www.w3school.com.cn/jsref/jsref_strike.asp) | 使用删除线来显示字符串。 |
| [sub()](http://www.w3school.com.cn/jsref/jsref_sub.asp) | 把字符串显示为下标。 |
| [substr()](http://www.w3school.com.cn/jsref/jsref_substr.asp) | 从起始索引号提取字符串中指定数目的字符。 |
| [substring()](http://www.w3school.com.cn/jsref/jsref_substring.asp) | 提取字符串中两个指定的索引号之间的字符。 |
| [sup()](http://www.w3school.com.cn/jsref/jsref_sup.asp) | 把字符串显示为上标。 |
| [toLocaleLowerCase()](http://www.w3school.com.cn/jsref/jsref_toLocaleLowerCase.asp) | 把字符串转换为小写。 |
| [toLocaleUpperCase()](http://www.w3school.com.cn/jsref/jsref_toLocaleUpperCase.asp) | 把字符串转换为大写。 |
| [toLowerCase()](http://www.w3school.com.cn/jsref/jsref_toLowerCase.asp) | 把字符串转换为小写。 |
| [toUpperCase()](http://www.w3school.com.cn/jsref/jsref_toUpperCase.asp) | 把字符串转换为大写。 |
| toSource() | 代表对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_string.asp) | 返回字符串。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueOf_string.asp) | 返回某个字符串对象的原始值。 |

##### 字符方法

charAt() //获取指定位置处字符

charCodeAt() //获取指定位置处字符的ASCII码

str[0] //HTML5，IE8+支持 和charAt()等效

##### 字符串操作方法

concat() //拼接字符串，等效于+，+更常用

slice() //从start位置开始，截取到end位置，end取不到

substring() //从start位置开始，截取到end位置，end取不到

substr() //从start位置开始，截取length个字符

##### 位置方法

indexOf() //返回指定内容在元字符串中的位置

lastIndexOf() //从后往前找，只找第一个匹配的

##### 去除空白

trim() //只能去除字符串前后的空白

##### 大小写转换方法

to(Locale)UpperCase() //转换大写

to(Locale)LowerCase() //转换小写

## Array（数组）对象

### 对象定义（数组的构造函数）

var arr = new Array(参数);

参数位置一个数值时为数组长度，多个数值时为数组中的元素。

1、下面的代码创建名为 cars 的数组：

var cars=new Array();

cars[0]="Audi";

cars[1]="BMW";

cars[2]="Volvo";

2、或者 (condensed array):在创建数组对象的时候赋值

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

3、或者 (literal array):不创建变量，直接赋值（字面量数组）

实例

var cars=["Audi","BMW","Volvo"];

注：在创建数组时可为数组指定长度，长度可任意指定，但实际上数组是变长的。

var cars=new Array(8);

### 获取数组中的元素

数组中的指定元素 = 数组名[索引值];

数组的索引代表的是数组中的元素在数组中的位置，从0开始。

如果获取数组中元素是，数组名[索引值]，没有指定索引（元素没那么多），系统不报错，而是给定值为undefined；

### 遍历数组（获取并操作数组中的每一个元素）（！！！重点！！！）

var arr = [1,2,3];  
for(var i = 0;i<arr.**length**;i++){  
 arr[i] //如此操作数组中的每一个元素  
}

### Array 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_array.asp) | 返回对创建此对象的数组函数的引用。 |
| [length](http://www.w3school.com.cn/jsref/jsref_length_array.asp) | 设置或返回数组中元素的数目。 |
| [prototype](http://www.w3school.com.cn/jsref/jsref_prototype_array.asp) | 使您有能力向对象添加属性和方法。 |

### Array 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_array.asp) | 连接两个或更多的数组，并返回结果。 |
| [join()](http://www.w3school.com.cn/jsref/jsref_join.asp) | 把数组的所有元素放入一个字符串。元素通过指定的分隔符分隔。 |
| [pop()](http://www.w3school.com.cn/jsref/jsref_pop.asp) | 删除并返回数组的最后一个元素 |
| [push()](http://www.w3school.com.cn/jsref/jsref_push.asp) | 向数组的末尾添加一个或更多元素，并返回新的长度。 |
| [reverse()](http://www.w3school.com.cn/jsref/jsref_reverse.asp) | 颠倒数组中元素的顺序。 |
| [shift()](http://www.w3school.com.cn/jsref/jsref_shift.asp) | 删除并返回数组的第一个元素 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_array.asp) | 从某个已有的数组返回选定的元素 |
| [sort()](http://www.w3school.com.cn/jsref/jsref_sort.asp) | 对数组的元素进行排序 |
| [splice()](http://www.w3school.com.cn/jsref/jsref_splice.asp) | 删除元素，并向数组添加新元素。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_array.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_array.asp) | 把数组转换为字符串，并返回结果。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString_array.asp) | 把数组转换为本地数组，并返回结果。 |
| [unshift()](http://www.w3school.com.cn/jsref/jsref_unshift.asp) | 向数组的开头添加一个或更多元素，并返回新的长度。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_array.asp) | 返回数组对象的原始值 |

#### 检测数组

Instanceof: 是一个关键字。 判断A是否是B类型。

布尔类型值 = A Instanceof B ;

Array.isArray() //HTML5中新增 判断是不是数组

布尔类型值 = Array.isArray(变量) ;

调用者：Array 参数：变量(被检测值) 返回值：布尔类型

#### 转换数组

toString() //把数组转换成字符串，每一项用,分割

字符串 = 数组.toString();

valueOf() //返回数组对象本身，用于返回指定对象的原始值

// Array：返回数组对象本身  
var array = ["CodePlayer", true, 12, -5];  
document.writeln( array.valueOf() === array ); // true

Join //根据每个字符把数组元素连起来变成字符串

字符串 = 数组.join(变量);

变量可以有可以没有。不写默认用逗号分隔，无缝连接用空字符串。

#### 栈操作(先进后出)

push() //在数组最后面插入项，返回数组的长度

数组1改后的长度 = 数组1.push(元素1);

pop() //取出数组中的最后一项，修改length属性，返回最后一项

被删除的元素 = 数组1.pop();

#### 队列操作(先进先出)

push()

shift()//取出数组中的第一个元素，修改length属性，返回最后一项

被删除的元素 = 数组1.shift();

unshift() //在数组最前面插入项，返回数组的长度

数组1改后的长度 = 数组1.unshift(元素1);

#### 排序方法

reverse() //翻转数组（原数组将被反转，返回值也是被反转后的数组）

反转后的数组 = 数组1.reverse();

sort();

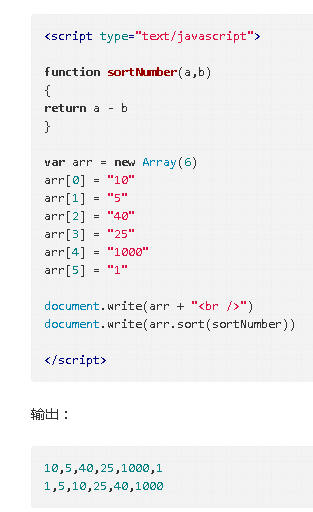
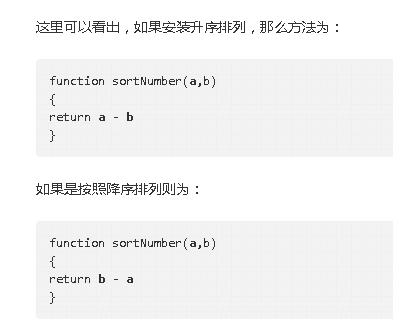
从小到大排序后的数组 = 数组1.sort(function(a,b){

return a-b;

});

**无参：**按照数组元素的首字符对应的Unicode编码值从小到大排列数组元素。

**带参：**必须为函数（回调函数--callback）。函数中带有两个参数，代表数组中的前后元素。如果计算后（a-b），返回值为负数，a排b前面。等于0不动。返回值为正数，a排b后面。

#### 操作方法

concat() //把参数拼接到当前数组

新数组 = 数组1.concat(数组2);

slice() //从当前数组中截取一个新的数组，不影响原来的数组，参数start从0开始,end从1开始

新数组 = 数组1.slice(索引1，索引2);

splice()//删除或替换当前数组的某些项目，参数start,deleteCount,options(要替换的项目)

新数组 = 数组1.splice(起始索引，结束索引，替换内容);

#### 位置方法

indexOf()、lastIndexOf() //如果没找到返回-1

索引值 = 数组.indexOf/lastIndexOf(数组中的元素);

#### 迭代方法

不会修改原数组

every()、filter()、forEach()、map()、some()

数组/boolean/无 = 数组.every/filter/forEach/map/some(

function(element,index,arr){

程序和返回值；

}

);

//对数组中每一项运行以下函数，如果都返回true，every返回true，如果有一项返回false，则停止遍历 every返回false；不写默认返回false

array.every(function(item,index,arr) {

})

//对数组中每一项运行以下函数，该函数返回结果是true的项组成的新数组  
var arr = array.filter(function(item,index,arr) {  
});  
console.log(arr);

//遍历数组  
array.forEach(function(item,index,arr){  
});

//对数组中每一项运行以下函数，返回该函数的结果组成的新数组  
var arr = array.map(function(item,index,arr) {  
 return "\"" + item + "\"";  
})

//对数组中每一项运行以下函数，如果该函数对某一项返回true，则some返回true  
var b = array.some(function(item,index,arr) {  
 if (item == "ww") {  
 return true;  
 }  
 return false;  
});

#### 清空数组

var array = [1,2,3,4,5,6];

array.splice(0,array.length); //删除数组中所有项目

array.length = 0; //length属性可以赋值，其它语言中length是只读

array = []; //推荐

### 了解成员数量（数组属性length）

JS数组的length属性是可变的，这一点需要特别注意。

var arr = [55,22,33,55,44];

document.write(arr.length);//数组长度为5

arr.length = 10;//增大数组的长度

document.write(arr.length);//数组长度变为10

### 二维数组

定义方法一：

var myarr = new Array(); //先声明一维

for(var i=0;i<2;i++){ //一维长度为2

myarr[i]=new Array(); //再声明二维

for(var j=0; j<2;j++){ //二维长度为3

myarr[i][j] = i+j; //赋值

}

}

定义方法二：

var myarr = [[0,1,2],[2,3,1]]

赋值：

myarr[0][1] = 5;

## Date（日期）对象

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(参数) ;

**注：1、**参数也可以没有，此时Date 对象自动使用当前的日期和时间作为其初始值。

2、有参数时。参数格式为“MM DD,YYYY,hh:mm:ss“（即月 日，年，时：分：秒）。举例如下。

var dt =new Date(“May 15,2012,09:45:23”) ;

### Date 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| [constructor](http://www.w3school.com.cn/jsref/jsref_constructor_date.asp) | 返回对创建此对象的 Date 函数的引用。 |
| [prototype](http://www.w3school.com.cn/jsref/jsref_prototype_date.asp) | 使您有能力向对象添加属性和方法。 |

### Date 对象方法

|  |  |
| --- | --- |
| 方法 | 描述 |
| [Date()](http://www.w3school.com.cn/jsref/jsref_Date.asp) | 返回当日的日期和时间。 |
| [getDate()](http://www.w3school.com.cn/jsref/jsref_getDate.asp) | 从 Date 对象返回一个月中的某一天 (1 ~ 31)。 |
| [getDay()](http://www.w3school.com.cn/jsref/jsref_getDay.asp) | 从 Date 对象返回一周中的某一天（返回值是 0（周日） 到 6（周六） 之间的一个整数）。 |
| [getMonth()](http://www.w3school.com.cn/jsref/jsref_getMonth.asp) | 从 Date 对象返回月份（以数组形式存储 返回0到11）。 |
| [getFullYear()](http://www.w3school.com.cn/jsref/jsref_getFullYear.asp) | 从 Date 对象以四位数字返回年份。 |
| [getYear()](http://www.w3school.com.cn/jsref/jsref_getYear.asp) | 请使用 getFullYear() 方法代替。 |
| [getHours()](http://www.w3school.com.cn/jsref/jsref_getHours.asp) | 返回 Date 对象的小时 (0 ~ 23)。 |
| [getMinutes()](http://www.w3school.com.cn/jsref/jsref_getMinutes.asp) | 返回 Date 对象的分钟 (0 ~ 59)。 |
| [getSeconds()](http://www.w3school.com.cn/jsref/jsref_getSeconds.asp) | 返回 Date 对象的秒数 (0 ~ 59)。 |
| [getMilliseconds()](http://www.w3school.com.cn/jsref/jsref_getMilliseconds.asp) | 返回 Date 对象的毫秒(0 ~ 999)。 |
| [getTime()](http://www.w3school.com.cn/jsref/jsref_getTime.asp) | 返回 1970 年 1 月 1 日至今的毫秒数。 |
| [getTimezoneOffset()](http://www.w3school.com.cn/jsref/jsref_getTimezoneOffset.asp) | 返回本地时间与格林威治标准时间 (GMT) 的分钟差。 |
| [getUTCDate()](http://www.w3school.com.cn/jsref/jsref_getUTCDate.asp) | 根据世界时从 Date 对象返回月中的一天 (1 ~ 31)。 |
| [getUTCDay()](http://www.w3school.com.cn/jsref/jsref_getUTCDay.asp) | 根据世界时从 Date 对象返回周中的一天 (0 ~ 6)。 |
| [getUTCMonth()](http://www.w3school.com.cn/jsref/jsref_getUTCMonth.asp) | 根据世界时从 Date 对象返回月份 (0 ~ 11)。 |
| [getUTCFullYear()](http://www.w3school.com.cn/jsref/jsref_getUTCFullYear.asp) | 根据世界时从 Date 对象返回四位数的年份。 |
| [getUTCHours()](http://www.w3school.com.cn/jsref/jsref_getUTCHours.asp) | 根据世界时返回 Date 对象的小时 (0 ~ 23)。 |
| [getUTCMinutes()](http://www.w3school.com.cn/jsref/jsref_getUTCMinutes.asp) | 根据世界时返回 Date 对象的分钟 (0 ~ 59)。 |
| [getUTCSeconds()](http://www.w3school.com.cn/jsref/jsref_getUTCSeconds.asp) | 根据世界时返回 Date 对象的秒钟 (0 ~ 59)。 |
| [getUTCMilliseconds()](http://www.w3school.com.cn/jsref/jsref_getUTCMilliseconds.asp) | 根据世界时返回 Date 对象的毫秒(0 ~ 999)。 |
| [parse()](http://www.w3school.com.cn/jsref/jsref_parse.asp) | 返回1970年1月1日午夜到指定日期（字符串）的毫秒数。 |
| [setDate()](http://www.w3school.com.cn/jsref/jsref_setDate.asp) | 设置 Date 对象中月的某一天 (1 ~ 31)。 |
| [setMonth()](http://www.w3school.com.cn/jsref/jsref_setMonth.asp) | 设置 Date 对象中月份 (0 ~ 11)。 |
| [setFullYear()](http://www.w3school.com.cn/jsref/jsref_setFullYear.asp) | 设置 Date 对象中的年份（四位数字）。 |
| [setYear()](http://www.w3school.com.cn/jsref/jsref_setYear.asp) | 请使用 setFullYear() 方法代替。 |
| [setHours()](http://www.w3school.com.cn/jsref/jsref_setHours.asp) | 设置 Date 对象中的小时 (0 ~ 23)。 |
| [setMinutes()](http://www.w3school.com.cn/jsref/jsref_setMinutes.asp) | 设置 Date 对象中的分钟 (0 ~ 59)。 |
| [setSeconds()](http://www.w3school.com.cn/jsref/jsref_setSeconds.asp) | 设置 Date 对象中的秒钟 (0 ~ 59)。 |
| [setMilliseconds()](http://www.w3school.com.cn/jsref/jsref_setMilliseconds.asp) | 设置 Date 对象中的毫秒 (0 ~ 999)。 |
| [setTime()](http://www.w3school.com.cn/jsref/jsref_setTime.asp) | 以毫秒设置 Date 对象。 |
| [setUTCDate()](http://www.w3school.com.cn/jsref/jsref_setUTCDate.asp) | 根据世界时设置 Date 对象中月份的一天 (1 ~ 31)。 |
| [setUTCMonth()](http://www.w3school.com.cn/jsref/jsref_setUTCMonth.asp) | 根据世界时设置 Date 对象中的月份 (0 ~ 11)。 |
| [setUTCFullYear()](http://www.w3school.com.cn/jsref/jsref_setUTCFullYear.asp) | 根据世界时设置 Date 对象中的年份（四位数字）。 |
| [setUTCHours()](http://www.w3school.com.cn/jsref/jsref_setutchours.asp) | 根据世界时设置 Date 对象中的小时 (0 ~ 23)。 |
| [setUTCMinutes()](http://www.w3school.com.cn/jsref/jsref_setUTCMinutes.asp) | 根据世界时设置 Date 对象中的分钟 (0 ~ 59)。 |
| [setUTCSeconds()](http://www.w3school.com.cn/jsref/jsref_setUTCSeconds.asp) | 根据世界时设置 Date 对象中的秒钟 (0 ~ 59)。 |
| [setUTCMilliseconds()](http://www.w3school.com.cn/jsref/jsref_setUTCMilliseconds.asp) | 根据世界时设置 Date 对象中的毫秒 (0 ~ 999)。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_boolean.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_date.asp) | 把 Date 对象转换为字符串。 |
| [toTimeString()](http://www.w3school.com.cn/jsref/jsref_toTimeString.asp) | 把 Date 对象的时间部分转换为字符串。 |
| [toDateString()](http://www.w3school.com.cn/jsref/jsref_toDateString.asp) | 把 Date 对象的日期部分转换为字符串。 |
| [toGMTString()](http://www.w3school.com.cn/jsref/jsref_toGMTString.asp) | 请使用 toUTCString() 方法代替。 |
| [toUTCString()](http://www.w3school.com.cn/jsref/jsref_toUTCString.asp) | 根据世界时，把 Date 对象转换为字符串。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString.asp) | 根据本地时间格式，把 Date 对象转换为字符串。 |
| [toLocaleTimeString()](http://www.w3school.com.cn/jsref/jsref_toLocaleTimeString.asp) | 根据本地时间格式，把 Date 对象的时间部分转换为字符串。 |
| [toLocaleDateString()](http://www.w3school.com.cn/jsref/jsref_toLocaleDateString.asp) | 根据本地时间格式，把 Date 对象的日期部分转换为字符串。 |
| [UTC()](http://www.w3school.com.cn/jsref/jsref_utc.asp) | 根据世界时返回 1970 年 1 月 1 日 到指定日期的毫秒数。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueOf_date.asp) | 返回 Date 对象的原始值。 |

### 比较Date（日期）

日期对象也可用于比较两个日期。

function tab(date1,date2){

    var oDate1 = new Date(date1);

    var oDate2 = new Date(date2);

    if(oDate1.getTime() > oDate2.getTime()){

        console.log('第一个大');

    } else {

        console.log('第二个大');

    }

}

tab('2015-10-10','2015-10-11');

## Math 对象

Math 对象用于执行数学任务。

var pi\_value=Math.PI;

var sqrt\_value=Math.sqrt(15);

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

### Math 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [E](http://www.w3school.com.cn/jsref/jsref_e.asp) | 返回算术常量 e，即自然对数的底数（约等于2.718）。 |
| [LN2](http://www.w3school.com.cn/jsref/jsref_ln2.asp) | 返回 2 的自然对数（约等于0.693）。 |
| [LN10](http://www.w3school.com.cn/jsref/jsref_ln10.asp) | 返回 10 的自然对数（约等于2.302）。 |
| [LOG2E](http://www.w3school.com.cn/jsref/jsref_log2e.asp) | 返回以 2 为底的 e 的对数（约等于 1.414）。 |
| [LOG10E](http://www.w3school.com.cn/jsref/jsref_log10e.asp) | 返回以 10 为底的 e 的对数（约等于0.434）。 |
| [PI](http://www.w3school.com.cn/jsref/jsref_pi.asp) | 返回圆周率（约等于3.14159）。 |
| [SQRT1\_2](http://www.w3school.com.cn/jsref/jsref_sqrt1_2.asp) | 返回返回 2 的平方根的倒数（约等于 0.707）。 |
| [SQRT2](http://www.w3school.com.cn/jsref/jsref_sqrt2.asp) | 返回 2 的平方根（约等于 1.414）。 |

### Math 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [abs(x)](http://www.w3school.com.cn/jsref/jsref_abs.asp) | 返回数的绝对值。 |
| [acos(x)](http://www.w3school.com.cn/jsref/jsref_acos.asp) | 返回数的反余弦值。 |
| [asin(x)](http://www.w3school.com.cn/jsref/jsref_asin.asp) | 返回数的反正弦值。 |
| [atan(x)](http://www.w3school.com.cn/jsref/jsref_atan.asp) | 以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。 |
| [atan2(y,x)](http://www.w3school.com.cn/jsref/jsref_atan2.asp) | 返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。 |
| [ceil(x)](http://www.w3school.com.cn/jsref/jsref_ceil.asp) | 对数进行上舍入（天花板函数）大于等于 x，并且与它最接近的整数。 |
| [cos(x)](http://www.w3school.com.cn/jsref/jsref_cos.asp) | 返回数的余弦。 |
| [exp(x)](http://www.w3school.com.cn/jsref/jsref_exp.asp) | 返回 e 的指数。 |
| [floor(x)](http://www.w3school.com.cn/jsref/jsref_floor.asp) | 对数进行下舍入（地板函数）。 |
| [log(x)](http://www.w3school.com.cn/jsref/jsref_log.asp) | 返回数的自然对数（底为e）。 |
| [max(x,y)](http://www.w3school.com.cn/jsref/jsref_max.asp) | 返回 x 和 y 中的最高值。 |
| [min(x,y)](http://www.w3school.com.cn/jsref/jsref_min.asp) | 返回 x 和 y 中的最低值。 |
| [pow(x,y)](http://www.w3school.com.cn/jsref/jsref_pow.asp) | 返回 x 的 y 次幂。 |
| [random()](http://www.w3school.com.cn/jsref/jsref_random.asp) | 返回 0 ~ 1 之间的随机数（包括0 不包括1）。 |
| [round(x)](http://www.w3school.com.cn/jsref/jsref_round.asp) | 把数四舍五入为最接近的整数。 |
| [sin(x)](http://www.w3school.com.cn/jsref/jsref_sin.asp) | 返回数的正弦。 |
| [sqrt(x)](http://www.w3school.com.cn/jsref/jsref_sqrt.asp) | 返回数的平方根。 |
| [tan(x)](http://www.w3school.com.cn/jsref/jsref_tan.asp) | 返回角的正切。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_math.asp) | 返回该对象的源代码。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_math.asp) | 返回 Math 对象的原始值。 |

## RegExp 对象

new RegExp(pattern, attributes);

**参数**

参数 *pattern* 是一个字符串，指定了正则表达式的模式或其他正则表达式。

参数 *attributes* 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 *pattern* 是正则表达式，而不是字符串，则必须省略该参数。

**返回值**

一个新的 RegExp 对象，具有指定的模式和标志。如果参数 *pattern* 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。

如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 *pattern* 是正则表达式时，它只返回 *pattern*，而不再创建一个新的 RegExp 对象。

**抛出**

SyntaxError - 如果 *pattern* 不是合法的正则表达式，或 *attributes* 含有 "g"、"i" 和 "m" 之外的字符，抛出该异常。

TypeError - 如果 *pattern* 是 RegExp 对象，但没有省略 *attributes* 参数，抛出该异常。

### RegExp 对象的属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| [global](http://www.w3school.com.cn/jsref/jsref_regexp_global.asp) | RegExp 对象是否具有标志 g。 |
| [ignoreCase](http://www.w3school.com.cn/jsref/jsref_regexp_ignorecase.asp) | RegExp 对象是否具有标志 i。 |
| [lastIndex](http://www.w3school.com.cn/jsref/jsref_lastindex_regexp.asp) | 一个整数，标示开始下一次匹配的字符位置。 |
| [multiline](http://www.w3school.com.cn/jsref/jsref_multiline_regexp.asp) | RegExp 对象是否具有标志 m。 |
| [source](http://www.w3school.com.cn/jsref/jsref_source_regexp.asp) | 正则表达式的源文本。 |

### RegExp 对象的方法

**1、test() 方法检索字符串中的指定值。返回值是 true 或 false。**

var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

由于该字符串中存在字母 "e"，以上代码的输出将是：true

**2、exec() 方法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。**

例子1：

var patt1=new RegExp("e");

document.write(patt1.exec("The best things in life are free"));

由于该字符串中存在字母 "e"，以上代码的输出将是：e

例子2：

var patt1=new RegExp("e","g");

do

{

result=patt1.exec("The best things in life are free");

document.write(result);

}

while (result!=null)

由于这个字符串中 6 个 "e" 字母，代码的输出将是：

eeeeeenull

**3、compile() 方法用于改变 RegExp。**

compile() 既可以改变检索模式，也可以添加或删除第二个参数。

var patt1=new RegExp("e");

document.write(patt1.test("The best things in life are free"));

patt1.compile("d");

document.write(patt1.test("The best things in life are free"));

由于字符串中存在 "e"，而没有 "d"，以上代码的输出是：

truefalse

## JS全局对象

全局属性和函数可用于所有内建的 JavaScript 对象。

**顶层函数（全局函数）**

|  |  |
| --- | --- |
| 函数 | 描述 |
| [decodeURI()](http://www.w3school.com.cn/jsref/jsref_decodeURI.asp) | 解码某个编码的 URI。 |
| [decodeURIComponent()](http://www.w3school.com.cn/jsref/jsref_decodeURIComponent.asp) | 解码一个编码的 URI 组件。 |
| [encodeURI()](http://www.w3school.com.cn/jsref/jsref_encodeuri.asp) | 把字符串编码为 URI。 |
| [encodeURIComponent()](http://www.w3school.com.cn/jsref/jsref_encodeURIComponent.asp) | 把字符串编码为 URI 组件。 |
| [escape()](http://www.w3school.com.cn/jsref/jsref_escape.asp) | 对字符串进行编码。 |
| [eval()](http://www.w3school.com.cn/jsref/jsref_eval.asp) | 计算 JavaScript 字符串，并把它作为脚本代码来执行。 |
| [getClass()](http://www.w3school.com.cn/jsref/jsref_getClass.asp) | 返回一个 JavaObject 的 JavaClass。 |
| [isFinite()](http://www.w3school.com.cn/jsref/jsref_isFinite.asp) | 检查某个值是否为有穷大的数。 |
| [isNaN()](http://www.w3school.com.cn/jsref/jsref_isNaN.asp) | 检查某个值是否是数字。 |
| [Number()](http://www.w3school.com.cn/jsref/jsref_number.asp) | 把对象的值转换为数字。 |
| [parseFloat()](http://www.w3school.com.cn/jsref/jsref_parseFloat.asp) | 解析一个字符串并返回一个浮点数。 |
| [parseInt()](http://www.w3school.com.cn/jsref/jsref_parseInt.asp) | 解析一个字符串并返回一个整数。 |
| [String()](http://www.w3school.com.cn/jsref/jsref_string.asp) | 把对象的值转换为字符串。 |
| [unescape()](http://www.w3school.com.cn/jsref/jsref_unescape.asp) | 对由 escape() 编码的字符串进行解码。 |

**顶层属性（全局属性）**

|  |  |
| --- | --- |
| 方法 | 描述 |
| [Infinity](http://www.w3school.com.cn/jsref/jsref_infinity.asp) | 代表正的无穷大的数值。 |
| [java](http://www.w3school.com.cn/jsref/jsref_java.asp) | 代表 java.\* 包层级的一个 JavaPackage。 |
| [NaN](http://www.w3school.com.cn/jsref/jsref_nan.asp) | 指示某个值是不是数字值。 |
| [Packages](http://www.w3school.com.cn/jsref/jsref_Packages.asp) | 根 JavaPackage 对象。 |
| [undefined](http://www.w3school.com.cn/jsref/jsref_undefined.asp) | 指示未定义的值。 |

## JS事件对象

事件通常与函数配合使用，这样就可以通过发生的事件来驱动函数执行。

HTML 4.0 的新特性之一是有能力使 HTML 事件触发浏览器中的动作（action），比如当用户点击某个 HTML 元素时启动一段 JavaScript。下面是一个属性列表，这些属性可插入 HTML 标签来定义事件动作。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 当以下情况发生时，出现此事件 | FF | N | IE |
| onabort | 图像加载被中断 | 1 | 3 | 4 |
| onblur | 元素失去焦点 | 1 | 2 | 3 |
| onchange | 用户改变域的内容，文本内容或下拉菜单中的选项发生改变 | 1 | 2 | 3 |
| onclick | 鼠标点击某个对象 | 1 | 2 | 3 |
| ondblclick | 鼠标双击某个对象 | 1 | 4 | 4 |
| onerror | 当加载文档或图像时发生某个错误 | 1 | 3 | 4 |
| onfocus | 元素获得焦点 | 1 | 2 | 3 |
| onkeydown | 某个键盘的键被按下 | 1 | 4 | 3 |
| onkeypress | 某个键盘的键被按下或按住 | 1 | 4 | 3 |
| onkeyup | 某个键盘的键被松开 | 1 | 4 | 3 |
| onload | 网页文档加载事件，某个页面或图像被完成加载 | 1 | 2 | 3 |
| onmousedown | 某个鼠标按键被按下 | 1 | 4 | 4 |
| onmousemove | 鼠标被移动 | 1 | 6 | 3 |
| onmouseout | 鼠标从某元素移开 | 1 | 4 | 4 |
| onmouseover | 鼠标被移到某元素之上 | 1 | 2 | 3 |
| onmouseup | 某个鼠标按键被松开 | 1 | 4 | 4 |
| onreset | 重置表单时，重置按钮被点击 | 1 | 3 | 4 |
| onresize | 窗口或框架被调整尺寸 | 1 | 4 | 4 |
| onselect | 文本被选定 | 1 | 2 | 3 |
| onsubmit | 表单提交事件，提交按钮被点击 | 1 | 2 | 3 |
| onunload | 用户退出页面，即关闭网页时 | 1 | 2 | 3 |

### onload 和 onunload 事件

onload 和 onunload 事件会在用户进入或离开页面时被触发。

onload 事件可用于检测访问者的浏览器类型和浏览器版本，并基于这些信息来加载网页的正确版本。

onload 和 onunload 事件可用于处理 cookie。

实例

<body onload="checkCookies()">

<script>

function checkCookies()

{

if (navigator.cookieEnabled==true)

{

alert("已启用 cookie")

}

else

{

alert("未启用 cookie")

}

}

</script>

</body>

### onchange 事件

onchange 事件常结合对输入字段的验证来使用。

下面是一个如何使用 onchange 的例子。当用户改变输入字段的内容时，会调用 upperCase() 函数。

<input type="text" id="fname" onchange="upperCase()">

### onmouseover 和 onmouseout 事件

onmouseover 和 onmouseout 事件可用于在用户的鼠标移至 HTML 元素上方或移出元素时触发函数。

### onmousedown、onmouseup 以及 onclick 事件

onmousedown, onmouseup 以及 onclick 构成了鼠标点击事件的所有部分。首先当点击鼠标按钮时，会触发 onmousedown 事件，当释放鼠标按钮时，会触发 onmouseup 事件，最后，当完成鼠标点击时，会触发 onclick 事件。

### onfocus和onblur事件

当输入字段获得焦点时，会触发onfocus事件。

当光标离开当前获得聚焦对象的时候，触发onblur事件。

# JS函数

## 函数定义

**1、函数声明（自定义声明）**

console.log(f(5,6));

function f(a,b) {

return a + b;

}

//此处的代码执行没有问题，JavaScript解析器首先会把当前作用域的函数声明提前到整个作用域的最前面。

**2、函数表达式**

var myFun = function (a,b){

return a + b;

}

console.log(myFun(6,7));

若：

console.log(myFun(6,7));

var myFun = function (a,b){

return a + b;

}

//则报错：myFun is not a function

## 函数调用

函数定义好后，是不能自动执行的，需要调用它，直接在需要的位置写函数名。

第一种情况：在<script>标签内调用

<script>

function add(){

Sum = 5+6;

Alert(sum);

}

add();

</script>

第二种情况：在HTML文件中调用，如通过点击按钮后调用定义好的函数。

## 返回值

var temp = 函数名() = 该函数的返回值;

1. 如果函数没有显示的使用 return语句 ，那么函数有默认的返回值：undefined

2. 如果函数使用 return语句，那么跟再return后面的值，就成了函数的返回值

3. 如果函数使用 return语句，但是return后面没有任何值，那么函数的返回值 也是：undefined

4. 函数使用return语句后，这个函数会在执行完 return 语句之后停止并立即退出，也就是说return后面的所有其他代码都不会再执行。

## 函数名、函数体和函数加载问题（重点记忆）

1、函数名是什么

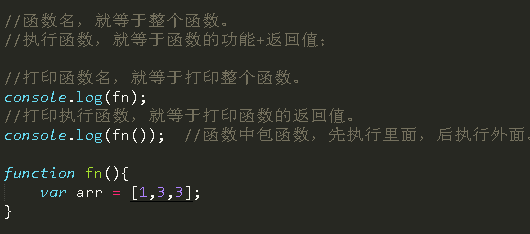
函数名 == 整个函数。

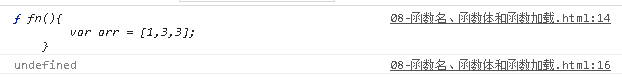
function fn(){alert(1)};

console.log(fn) == console.log(function fn(){alert(1)});

2、函数加载问题

JS加载的时候，只加载函数名，不加载函数体。所以如果想使用内部的成员变量，需要调用函数。





## 变量和作用域（隐式全局变量和变量声明提升）

### 变量和作用域（函数中的变量需要函数执行后才能使用）

1. 块级作用域

在其它语言中，任何一对花括号中的语句都属于一个块，在这之中定义的所有变量在代码块外都是不可见的

JavaScript中没有块级作用域

1. 全局变量（成员变量）

哪里都可以访问到的变量。

（函数外部或者进入javascript之后立即定义的变量和函数内部不带有var的变量）

1. 局部变量

函数内部的变量，只有函数内部可以访问到。

（函数内部用var定义的变量和形参）

四、其它

1、函数内部可以访问到该函数所属的外部作用域的变量(作用域链)

2、不使用var声明的变量是全局变量，不推荐使用。

3、变量退出作用域之后会销毁，全局变量关闭网页或浏览器才会销毁





### 隐式全局变量

隐式全局变量就是隐藏的全局变量不好被发现。

function fn（）{

var a = b = c = 1; // b和c就是隐式全局变量

}

注意:

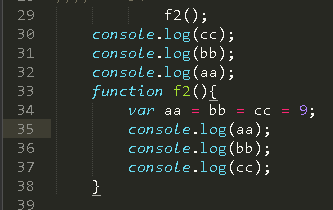
function fn（）{

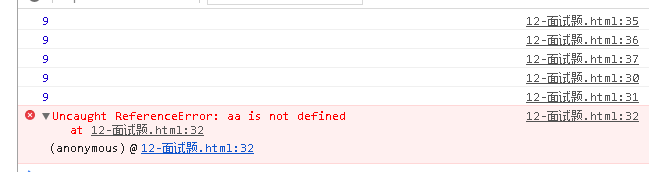
var a = b = c = 1; // b和c就是隐式全局变量（等号）

var a = 1; b = 2; c = 3; // b和c就是隐式全局变量（分号）

var a = 1 , b = 2 , c = 3; // b和c就不是隐式全局变量（逗号）

}

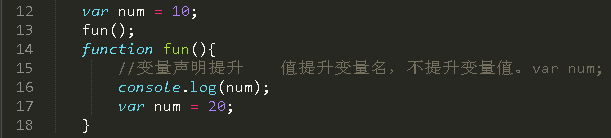




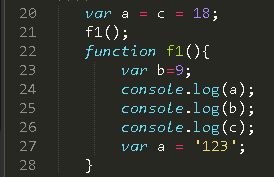
### 变量声明提升（出现原因：预解析）

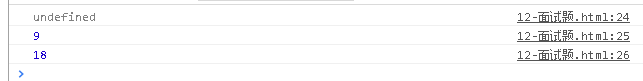
函数中，定义变量在使用变量之后。

定义变量的时候，变量的声明会被提升到作用域的最上面，变量的赋值不会提升，容易出现undefined。计算后形成NaN。









//提前看一眼这个习惯叫什么呢？ 预解析！  
//变量声明提升：在预解析的时候，成员变量和函数，被提升到最高位置，方便其他程序访问。  
//变量声明提升特点：成员变量只提升变量名，不提升变量值。但是，函数是所有内容全部提升。（function直接定义的）  
//函数范围内照样会出现变量声明提升  
//什么情况容易出现变量声明提升：使用变量在定义变量之前。

## 小知识

### 函数不调用不执行

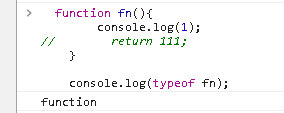
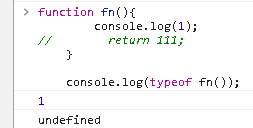
### 函数名就等于（整个函数）

### 参数相当于局部变量

### 就近原则使用变量

### 两个平级的函数中的变量不会相互影响（可以使用同样的形参名）

### 加载函数的时候，只加载函数名，不加载函数体

## 函数高级

### 没有重载（了解）

什么是重载：方法的签名相同(函数返回值、函数名称、函数参数)，其它语言(c++、Java、c#)中有方法的重载。

JavaScript中没有方法的重载

function f1(a,b) {

return a + b;

}

function f1(a,b,c) {

return a + b + c;

}

var result = f1(5,6); //NaN

三个参数的f1把两个参数的f1覆盖，调用的是三个参数的f1

证明JavaScript中没有重载

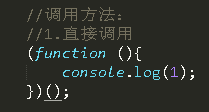
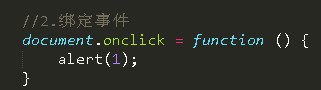
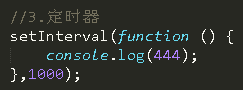
### 匿名函数（了解）

定义：匿名函数就是没有名字的函数。

作用：

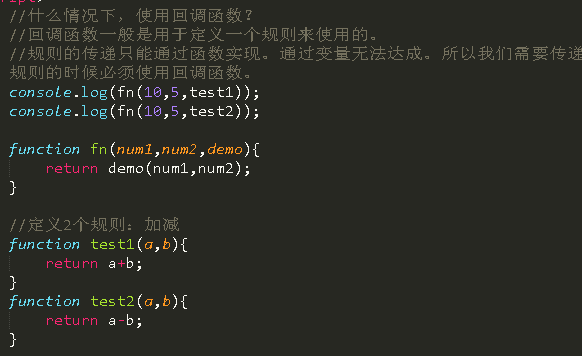
1. 不需要定义函数名的时候。（群众演员没必要起名，百万雄师下江南）
2. 书写起来更简便。

匿名函数的调用有三种方法：

1. 直接调用或自调用。
2. 事件绑定。
3. 定时器。

### 函数是一种类型（了解）

### 函数作为参数，回调函数（了解）



### 递归（理解）

递归：就是函数自己调用自己。（懂得）

必须有跳出条件。

# JS错误

## JS测试和捕捉

**try** 语句允许我们定义在执行时进行错误测试的代码块。

**catch** 语句允许我们定义当 try 代码块发生错误时，所执行的代码块。

JavaScript 语句 **try** 和 **catch** 是成对出现的。

语法

try

{

//在这里运行代码

}

catch(err)

{

//在这里处理错误

}

## Throw 语句

throw 语句允许我们创建自定义错误。

正确的技术术语是：创建或**抛出异常**（exception）。

如果把 throw 与 try 和 catch 一起使用，那么您能够控制程序流，并生成自定义的错误消息。

语法

throw exception

异常可以是 JavaScript 字符串、数字、逻辑值或对象。

# 认识DOM

文档对象模型DOM（Document Object Model）定义访问和处理HTML文档的标准方法，DOM将HTML文档呈现为带有元素、属性和文本的树结构（节点树）。

三种常见的DOM节点：1、元素节点2、文本节点3、属性节点

## Document 对象

Document 对象代表整个 HTML 文档，可用来访问页面中的所有元素。目的其实就是为了能让js操作html元素而制定的一个规范。

Document 对象是 Window 对象的一个部分，可通过 window.document 属性来访问。

**IE:***Internet Explorer,***F:***Firefox,***O:***Opera,***W3C:***World Wide Web Consortium (Internet 标准).*

## Document 对象的集合

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 集合 | 描述 | IE | F | O | W3C |
| [all[]](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/coll_doc_all.asp.html) | 提供对文档中所有 HTML 元素的访问。 | 4 | 1 | 9 | Yes |
| [anchors[]](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/coll_doc_anchors.asp.html) | 返回对文档中所有 Anchor 对象的引用。 | 4 | 1 | 9 | Yes |
| applets | 返回对文档中所有 Applet 对象的引用。 | - | - | - | - |
| [forms[]](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/coll_doc_forms.asp.html) | 返回对文档中所有 Form 对象引用。 | 4 | 1 | 9 | Yes |
| [images[]](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/coll_doc_images.asp.html) | 返回对文档中所有 Image 对象引用。 | 4 | 1 | 9 | Yes |
| [links[]](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/coll_doc_links.asp.html) | 返回对文档中所有 Area 和 Link 对象引用。 | 4 | 1 | 9 | Yes |

## Document 对象的属性

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 属性 | 描述 | IE | F | O | W3C |
| body | 提供对 <body> 元素的直接访问。对于定义了框架集的文档，该属性引用最外层的 <frameset>。 |  |  |  |  |
| [cookie](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_cookie.asp.html) | 设置或返回与当前文档有关的所有 cookie。 | 4 | 1 | 9 | Yes |
| [domain](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_domain.asp.html) | 返回当前文档的域名。 | 4 | 1 | 9 | Yes |
| [lastModified](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_lastmodified.asp.html) | 返回文档被最后修改的日期和时间。 | 4 | 1 | No | No |
| [referrer](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_referrer.asp.html) | 返回载入当前文档的文档的 URL。 | 4 | 1 | 9 | Yes |
| [title](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_title.asp.html) | 返回当前文档的标题。 | 4 | 1 | 9 | Yes |
| [URL](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_doc_url.asp.html) | 返回当前文档的 URL。 | 4 | 1 | 9 | Yes |

## Document 对象的方法

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 方法 | 描述 | IE | F | O | W3C |
| [close()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_close.asp.html) | 关闭用 document.open() 方法打开的输出流，并显示选定的数据。 | 4 | 1 | 9 | Yes |
| [getElementById()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_getelementbyid.asp.html) | 返回对拥有指定 id 的第一个对象的引用。 | 5 | 1 | 9 | Yes |
| [getElementsByName()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_getelementsbyname.asp.html) | 返回带有指定名称的对象集合。 | 5 | 1 | 9 | Yes |
| [getElementsByTagName()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_getelementsbytagname.asp.html) | 返回带有指定标签名的对象集合。 | 5 | 1 | 9 | Yes |
| [open()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_open.asp.html) | 打开一个流，以收集来自任何 document.write() 或 document.writeln() 方法的输出。 | 4 | 1 | 9 | Yes |
| [write()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_write.asp.html) | 向文档写 HTML 表达式 或 JavaScript 代码。 | 4 | 1 | 9 | Yes |
| [writeln()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_doc_writeln.asp.html) | 等同于 write() 方法，不同的是在每个表达式之后写一个换行符。 | 4 | 1 | 9 | Yes |

## 获取节点

### 通过ID获取元素

document.getElementById(“id”)

注：获取的元素是一个对象，如想对元素进行操作，我们要通过它的属性或方法。

### 通过TagName获取元素

本例查找 id="main" 的元素，然后查找 "main" 中的所有 <p> 元素：

var x=document.getElementById("main");

var y=x.getElementsByTagName("p");

### 通过name属性获取元素

**document.getElementByName(name)**

注：1、因为文档中的name属性可能不唯一，所有getElementByName()方法返回的是元素的数组，而不是一个元素。

1. 和数组类似也有length属性，可以和访问数组一样的方法来访问，从0开始。

### 通过ClassName获取元素

**document.getElementBy*ClassName* (*"a"*)**

通过类名查找 HTML 元素在 IE 5,6,7,8 中无效

## DOM 访问关系

### Js中的父子兄访问关系

|  |  |  |  |
| --- | --- | --- | --- |
| 父节点 | 兄弟节点 | 子节点 | 所有子节点 |
| parentNode | nextSibling  nextElementSibling  previousSibling  previousElementSibling | firstChild  firstElementChild  lastChild  lastElementChild | childNodes  children |

### childNodes和children区别

1、childNodes：它是标准属性，它返回指定元素的子元素集合，包括HTML节点，所有属性，文本节点

火狐 谷歌等高版本会把换行也看做是子节点

nodeType==1时才是元素节点

nodeType == 1 表示 的是 元素节点 记住 元素就是标签

nodeType == 2 表示是属性节点 了解

nodeType == 3 是文本节点 了解

2、children：非标准属性，它返回指定元素的子元素集合。

但它只返回HTML节点，甚至不返回文本节点，虽然不是标准的DOM属性，但它和innerHTML方法一样，得到了几乎所有浏览器的支持。

children在IE6/7/8中包含注释节点

## DOM节点操作

### 创**建节点**

createElement()

var createDiv = document.createElement("div");

### 插入子节点

1.appendChild（）

appendChild() 方法向节点添加最后一个子节点

2.insertBefore(插入的节点，参照节点)

insertBefore() 方法 ，如果不是在末尾插入节点，而是想放在特定的位置上，用这个方法.

该方法接受2个参数，第一个是要插入的节点，第二个是参照节点。如果 第二个参数 为null（不是不写），则默认插入到后面

可以给某个id 的前面

**实例**

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

<div id="div1">

<p id="p1">这是一个段落</p>

<p id="p2">这是另一个段落</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>

### 删除节点

removeChild（）

父节点.removeChild（子节点）；

不知道父级的情况下，可以这么写：node.parentNode.removeChild(node)

**示例：删除已有的 HTML 元素**

如需删除 HTML 元素，您必须首先获得该元素的父元素：

实例

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另一个段落。</p>

</div>

<script>

var parent=document.getElementById("div1");

var child=document.getElementById("p1");

parent.removeChild(child);

</script>

常用的解决方案：找到您希望删除的子元素，然后使用其 parentNode 属性来找到父元素：

var child=document.getElementById("p1");

child.parentNode.removeChild(child);

### 复制节点

cloneNode()

var newNode = oldNode.cloneNode() ;

用于复制节点， 接受一个布尔值参数， true 表示深复制（复制节点及其所有子节点）， false 表示浅复制（复制节点本身，不复制子节点）

### 设置节点属性

获取：getAttribute(名称)

设置：setAttribute(名称, 值)

删除：removeAttribute(名称)

## document.write()输出内容

document.write()可用于直接向HTML输出流写内容。

1、输出内容用”“括起，直接输出”“号内的内容。

document.write(“I love Javascript”);

2、通过变量输出内容

var mystr = “I love Javascript”;

document.write(mystr);

3、输出多项内容，内容之间用+号连接

var mystr = “I love Javascript”;

document.write(mystr+”hello”);

4、输出HTML标签，并起作用，标签使用”“括起来。

var mystr = “I love Javascript”;

document.write(mystr+”<br>”);//输出I love Javascript后，输出一个换行符

document.write(”hello”);

注：请使用 document.write() 仅仅向文档输出写内容。

如果在文档已完成加载后执行 document.write，整个 HTML 页面将被覆盖：

## innerHTML属性

innerHTML属性用于获取或替换HTML元素的内容

**Object.innerHTML**

注：1、Object是获取的元素对象，如通过document.getElementById(“id”)获取的元素。

2、注意书写，innerHTML区分大小写。

## 改变HTML样式

Object.style.property=new style;

注：1、Object是获取的元素对象，如通过document.getElementById(“id”)获取的元素。

2、property包括如height、color、font等CSS样式属性

## 改变HTML属性

如需改变 HTML 元素的属性，请使用这个语法：

document.getElementById(id).attribute=new value

本例改变了 <img> 元素的 src 属性：

<script>

document.getElementById("image").src="landscape.jpg";

</script>

## 显示和隐藏（display属性）

Object.style.display =value

注：1、Object是获取的元素对象，如通过document.getElementById(“id”)获取的元素。

2、value取值：none、block

## 控制类名（className属性）

className属性设置或返回元素的class属性

Object.className = className

作用：1、获取元素的class属性

1. 为网页的某个元素指定一个css样式来更改该元素的外观

## 对事件做出反应

我们可以在事件发生时执行 JavaScript，比如当用户在 HTML 元素上点击时。

如需在用户点击某个元素时执行代码，请向一个 HTML 事件属性添加 JavaScript 代码：

onclick=JavaScript

在本例中，当用户在 <h1> 元素上点击时，会改变其内容：

<h1 onclick="this.innerHTML='谢谢!'">请点击该文本</h1>

本例从事件处理器调用一个函数：

<!DOCTYPE html>

<html>

<head>

<script>

function changetext(id)

{id.innerHTML="谢谢!";}

</script>

</head>

<body>

<h1 onclick="changetext(this)">请点击该文本</h1>

</body>

</html>

## HTML事件属性

如需向 HTML 元素分配事件，您可以使用事件属性。

实例

向 button 元素分配 onclick 事件：

<button onclick="displayDate()">点击这里</button>

## 使用HTML DOM来分配事件

HTML DOM 允许您通过使用 JavaScript 来向 HTML 元素分配事件：

实例

向 button 元素分配 onclick 事件：

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

</script>

## 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。

ookie 有两种清除方式：

1、通过浏览器工具清除 cookie (有第三方的工具，浏览器自身也有这种功能

2、通过设置 cookie 的有效期来清除 cookie

## cookie的存取操作

document.cookie = 'username=Darren'

以上代码中'username'表示 cookie 名称，'Darren'表示这个名称对应的值。假设 cookie 名称并不存在，那么就是创建一个新的 cookie；如果存在就是修改了这个 cookie 名称对应的值。如果要多次创建 cookie ，重复使用这个方法即可。

## cookie的读取操作

要精确的对 cookie 进行读取其实很简单，就是对字符串进行操作。

## 设置cookie的有效期

文章中常常出现的 cookie 的生命周期也就是有效期和失效期，即 cookie 的存在时间。在默认的情况下，cookie 会在浏览器关闭的时候自动清除，但是我们可以通过expires来设置 cookie 的有效期。语法如下：

document.cookie = "name=value;expires=date"

上面代码中的date值为GMT(格林威治时间)格式的日期型字符串，生成方式如下：

var \_date = new Date();  
\_date.setDate(\_date.getDate()+30); //保存30天  
\_date.toGMTString(); //把date对象转换为字符串

## cookie 路径概念

　　在基础知识中有提到 cookie 有域和路径的概念，现在来介绍路径在 cookie 中的作用。

　　cookie 一般都是由于用户访问页面而被创建的，可是并不是只有在创建 cookie 的页面才可以访问这个 cookie。

　　默认情况下，只有与创建 cookie 的页面在同一个目录或子目录下的网页才可以访问，这个是因为安全方面的考虑，造成不是所有页面都可以随意访问其他页面创建的 cookie。举个例子：

　　在 "http://www.cnblogs.com/Darren\_code/" 这个页面创建一个cookie，那么在"/Darren\_code/"这个路径下的页面如： "http://www.cnblogs.com/Darren\_code/archive/2011/11/07/Cookie.html"这个页面默认就能取到cookie信息。

　　可在默认情况下， "http://www.cnblogs.com"或者 "http://www.cnblogs.com/xxxx/" 就不可以访问这个 cookie。

　　那么如何让这个 cookie 能被其他目录或者父级的目录访问类，通过设置 cookie 的路径就可以实现。例子如下：

　　document.cookie = "name=value;path=path"  
　　document.cookie = "name=value;expires=date;path=path"

 　　红色字体path就是 cookie 的路径，最常用的例子就是让 cookie 在根目录下,这样不管是哪个子页面创建的 cookie，所有的页面都可以访问到了:

　　document.cookie = "name=Darren;path=/"

## cookie 域概念

　　路径能解决在同一个域下访问 cookie 的问题，咱们接着说 cookie 实现**同域**之间访问的问题。语法如下：

　　document.cookie = "name=value;path=path;domain=domain"

　　红色的domain就是设置的 cookie 域的值。

　　例如 "www.qq.com" 与 "sports.qq.com" 公用一个关联的域名"qq.com"，我们如果想让 "sports.qq.com" 下的cookie被 "www.qq.com" 访问，我们就需要用到 cookie 的domain属性，并且需要把path属性设置为 "/"。例：

　　document.cookie = "username=Darren;path=/;domain=qq.com"

　　注：一定的是同域之间的访问，不能把domain的值设置成非主域的域名。

## cookie 安全性

　　通常 cookie 信息都是使用HTTP连接传递数据，这种传递方式很容易被查看，所以 cookie 存储的信息容易被窃取。假如 cookie 中所传递的内容比较重要，那么就要求使用加密的数据传输。

　　所以 cookie 的这个属性的名称是“secure”，默认的值为空。如果一个 cookie 的属性为secure，那么它与服务器之间就通过HTTPS或者其它安全协议传递数据。语法如下：

　　document.cookie = "username=Darren;secure"

　　把cookie设置为secure，只保证 cookie 与服务器之间的数据传输过程加密，而保存在本地的 cookie文件并不加密。如果想让本地cookie也加密，得自己加密数据。

　　注：就算设置了secure 属性也并不代表他人不能看到你机器本地保存的 cookie 信息，所以说到底，别把重要信息放cookie就对了。

## cookie 编码细节

　　在输入cookie信息时不能包含空格，分号，逗号等特殊符号，而在一般情况下，cookie 信息的存储都是采用未编码的方式。所以，在设置 cookie 信息以前要先使用escape()函数将 cookie 值信息进行编码，在获取到 cookie 值时再使用unescape()函数把值进行转换回来。如设置cookie时：

　　document.cookie = name + "="+ escape (value)

　　再看看基础用法时提到过的getCookie()内的一句：

　　return unescape(document.cookie.substring(c\_start,c\_end))

这样就不用担心因为在cookie值中出现了特殊符号而导致 cookie 信息出错了。

# 浏览器对象模型 (BOM)

浏览器对象模型（Browser Object Model）尚无正式标准。

## Window对象

所有浏览器都支持 **window** 对象。它表示浏览器窗口。

所有 JavaScript 全局对象、函数以及变量均自动成为 window 对象的成员。全局变量是 window 对象的属性。全局函数是 window 对象的方法。甚至 HTML DOM 的 document 也是 window 对象的属性之一：

window.document.getElementById("header");

与此相同：

document.getElementById("header");

**IE:** Internet Explorer, **F:** Firefox, **O:** Opera.

## Window 对象的集合

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Collection | Description | IE | F | O |
| frames[] | 返回窗口中所有命名的框架。该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架或 <iframe>。属性 frames.length 存放数组 frames[] 中含有的元素个数。注意，frames[] 数组中引用的框架可能还包括框架，它们自己也具有 frames[] 数组。 | 4 | 1 | 9 |

## Window 对象的属性

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 属性 | 描述 | IE | F | O |
| [closed](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_closed.asp.html) | 返回窗口是否已被关闭。 | 4 | 1 | 9 |
| [defaultStatus](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_defaultstatus.asp.html) | 设置或返回窗口状态栏中的默认文本。 | 4 | No | 9 |
| [document](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_document.asp.html) | 对 Document 对象的只读引用。请参阅 [Document 对象](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_document.asp.html)。 | 4 | 1 | 9 |
| [history](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_history.asp.html) | 对 History 对象的只读引用。请参数 [History 对象](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_history.asp.html)。 | 4 | 1 | 9 |
| [innerheight](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_innerheight_innerwidth.asp.html) | 返回窗口的文档显示区的高度。 | No | No | No |
| [innerwidth](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_innerheight_innerwidth.asp.html) | 返回窗口的文档显示区的宽度。 | No | No | No |
| length | 设置或返回窗口中的框架数量。 | 4 | 1 | 9 |
| [location](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_location.asp.html) | 用于窗口或框架的 Location 对象。请参阅 [Location 对象](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_location.asp.html)。 | 4 | 1 | 9 |
| [name](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_name.asp.html) | 设置或返回窗口的名称。 | 4 | 1 | 9 |
| [Navigator](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_navigator.asp.html) | 对 Navigator 对象的只读引用。请参数 [Navigator 对象](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_navigator.asp.html)。 | 4 | 1 | 9 |
| [opener](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_opener.asp.html) | 返回对创建此窗口的窗口的引用。 | 4 | 1 | 9 |
| [outerheight](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_outerheight.asp.html) | 返回窗口的外部高度。 | No | No | No |
| [outerwidth](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_outerwidth.asp.html) | 返回窗口的外部宽度。 | No | No | No |
| pageXOffset | 设置或返回当前页面相对于窗口显示区左上角的 X 位置。 | No | No | No |
| pageYOffset | 设置或返回当前页面相对于窗口显示区左上角的 Y 位置。 | No | No | No |
| parent | 返回父窗口。 | 4 | 1 | 9 |
| [Screen](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_screen.asp.html) | 对 Screen 对象的只读引用。请参数 [Screen 对象](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/dom_obj_screen.asp.html)。 | 4 | 1 | 9 |
| [self](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_self.asp.html) | 返回对当前窗口的引用。等价于 Window 属性。 | 4 | 1 | 9 |
| [status](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_status.asp.html) | 设置窗口状态栏的文本。 | 4 | No | 9 |
| [top](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/prop_win_top.asp.html) | 返回最顶层的先辈窗口。 | 4 | 1 | 9 |
| window | window 属性等价于 self 属性，它包含了对窗口自身的引用。 | 4 | 1 | 9 |
| screenLeft  screenTop  screenX  screenY | 只读整数。声明了窗口的左上角在屏幕上的的 x 坐标和 y 坐标。IE、Safari 和 Opera 支持 screenLeft 和 screenTop，而 Firefox 和 Safari 支持 screenX 和 screenY。 | 4 | 1 | 9 |

## Window 对象的方法

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 方法 | 描述 | IE | F | O |
| [alert()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_alert.asp.html) | 显示带有一段消息和一个确认按钮的警告框。 | 4 | 1 | 9 |
| [blur()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_blur.asp.html) | 把键盘焦点从顶层窗口移开。 | 4 | 1 | 9 |
| [clearInterval()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_clearinterval.asp.html) | 取消由 setInterval() 设置的 timeout。 | 4 | 1 | 9 |
| [clearTimeout()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_cleartimeout.asp.html) | 取消由 setTimeout() 方法设置的 timeout。 | 4 | 1 | 9 |
| [close()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_close.asp.html) | 关闭浏览器窗口。 | 4 | 1 | 9 |
| [confirm()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_confirm.asp.html) | 显示带有一段消息以及确认按钮和取消按钮的对话框。 | 4 | 1 | 9 |
| [createPopup()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_createpopup.asp.html) | 创建一个 pop-up 窗口。 | 4 | No | No |
| [focus()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_focus.asp.html) | 把键盘焦点给予一个窗口。 | 4 | 1 | 9 |
| [moveBy()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_moveby.asp.html) | 可相对窗口的当前坐标把它移动指定的像素。 | 4 | 1 | 9 |
| [moveTo()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_moveto.asp.html) | 把窗口的左上角移动到一个指定的坐标。 | 4 | 1 | 9 |
| [open()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_open.asp.html) | 打开一个新的浏览器窗口或查找一个已命名的窗口。 | 4 | 1 | 9 |
| [print()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_print.asp.html) | 打印当前窗口的内容。 | 5 | 1 | 9 |
| [prompt()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_prompt.asp.html) | 显示可提示用户输入的对话框。 | 4 | 1 | 9 |
| [resizeBy()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_resizeby.asp.html) | 按照指定的像素调整窗口的大小。 | 4 | 1 | 9 |
| [resizeTo()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_resizeto.asp.html) | 把窗口的大小调整到指定的宽度和高度。 | 4 | 1.5 | 9 |
| [scrollBy()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_scrollby.asp.html) | 按照指定的像素值来滚动内容。 | 4 | 1 | 9 |
| [scrollTo()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_scrollto.asp.html) | 把内容滚动到指定的坐标。 | 4 | 1 | 9 |
| [setInterval()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_setinterval.asp.html) | 按照指定的周期（以毫秒计）来调用函数或计算表达式。 | 4 | 1 | 9 |
| [setTimeout()](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_settimeout.asp.html) | 在指定的毫秒数后调用函数或计算表达式。 | 4 | 1 | 9 |

## Window对象常用事件

|  |  |
| --- | --- |
| 事件名称 | 说明 |
| onload | 一个页面或一幅图像完成加载 |
| [onmouseover](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_blur.asp.html) | 鼠标移到某个元素之上 |
| [onclick](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_clearinterval.asp.html) | 鼠标单击某个对象 |
| [onkeydown](http://www.jb51.net/shouce/htmldom/jb51.net.htmldom/htmldom/met_win_cleartimeout.asp.html) | 某个键盘按键被按下 |
| onchange | 域的内容被改变 |

## Window尺寸

有三种方法能够确定浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）。

对于Internet Explorer、Chrome、Firefox、Opera 以及 Safari：

window.innerHeight - 浏览器窗口的内部高度

window.innerWidth - 浏览器窗口的内部宽度

对于 Internet Explorer 8、7、6、5：

document.documentElement.clientHeight

document.documentElement.clientWidth

或者

document.body.clientHeight

document.body.clientWidth

实用的 JavaScript 方案（涵盖所有浏览器）：

var w=window.innerWidth

|| document.documentElement.clientWidth

|| document.body.clientWidth;

var h=window.innerHeight

|| document.documentElement.clientHeight

|| document.body.clientHeight;

该例显示浏览器窗口的高度和宽度：（不包括工具栏/滚动条）

## 打开新窗口（window.open）

open()方法可以查找一个已经存在或新建的浏览器窗口。

window.open([URL],[窗口名称],[参数字符串])

参数说明：

URL：可选参数，在窗口中要显示网页的网址或路径。如果省略这个参数，或者它的值是空字符串，那么窗口就不显示任何文档。

窗口名称：可选参数，被打开窗口的名称。

1、该名称由字母、数字、下划线字符组成。

2、”\_top“、”\_blank “、”\_self”具有特殊意义的名称。

\_top：框架网页中在上部窗口中显示目标网页

\_blank：在新创扩种显示目标网页

\_self：在当前窗口中显示目标网页

3、相同name的窗口只能创建一个，要想创建多个窗口则name不能相同。

4、name不能包含空格。

参数字符串：可选参数，设置窗口参数，各参数用逗号隔开。有top窗口顶部离屏幕顶部的像素数、left窗口左端离屏幕左端的像素数、width宽度、height高度、menubar有没有菜单、 toolbar有没有工具条、scrollbar有没有滚动条、status有没有状态栏（yes,no）

注：运行结果考虑浏览器兼容问题

## 关闭窗口（window.close）

window.close();//关闭本窗口

<窗口对象>.close();//关闭指定的窗口

//将新打开的窗口对象，存储在变量mywin中

var mywin=window.open('http://www.imooc.com');

muwin.close();

注：上面代码在打开新窗口的同时，关闭该窗口，看不到被打开的窗口。

## 警告（alert消息对话框）

alert(字符串或变量);

alert弹出消息对话框（包含一个确定按钮）

1、在点击对话框”确定“按钮前，不能进行任何其他操作。

2、消息对话框通常可以用于调试程序。

3、alert输出内容，可以是字符串或变量，与document.write相似。

## 确认（confirm消息对话框）

confirm(str);

参数说明：在消息对话框中要显示的文本

返回值：Boolean值

当用户点击”确定”按钮时，返回true

当用户点击”取消”按钮时，返回false

var mymessage=confirm("你喜欢JS吗？");

if (mymessage==true) {

document.write("good!");

}

else{

document.write("just learn it!");

}

注：消息对话框是排他的，即用户在点击对话框按钮前，不能进行任何其他操作。

## 提问（prompt消息对话框）

prompt(str1,str2);

参数说明：

str1：要显示在消息对话框中的文本，不可修改

str2：文本框中的内容，可以修改

返回值：

1、点击确定按钮，文本框中的内容将作为函数返回值

2、点击取消按钮，将返回null

var myname=prompt("请输入你的姓名：");

if (myname!=null){

alert("welcome"+myname);

}

else{

alert("hello my friend");

}

注：在用户点击对话框的按钮前，不能进行任何其他操作。

## setTimeout()

var t=setTimeout("javascript语句",毫秒)

setTimeout() 方法会返回某个值。在上面的语句中，值被储存在名为 t 的变量中。假如你希望取消这个 setTimeout()，你可以使用这个变量名来指定它。

setTimeout() 的第一个参数是含有 JavaScript 语句的字符串。这个语句可能诸如 "alert('5 seconds!')"，或者对函数的调用，诸如 alertMsg()"。

第二个参数指示从当前起多少毫秒后执行第一个参数。

**提示：**1000 毫秒等于一秒。

## clearTimeout()

clearTimeout(setTimeout\_variable)

## 其他 Window 方法

window.moveTo() - 移动当前窗口

window.resizeTo() - 调整当前窗口的尺寸

## Screen对象

window.screen 对象在编写时可以不使用 window 这个前缀。

### Window Screen 可用宽度

screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如窗口任务栏。

<script>

document.write("可用宽度："+screen.availWidth);

</script>

以上代码输出为：可用宽度：1600

### Window Screen 可用高度

screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如窗口任务栏。

<script>

document.write("可用高度：" + screen.availHeight);

</script>

以上代码输出为：可用高度：860

## Location 对象

Location 对象包含有关当前 URL 的信息。

Location 对象是 Window 对象的一个部分，可通过 window.location 属性来访问。

window.location 对象在编写时可不使用 window 这个前缀。

一些例子：

location.hostname 返回 web 主机的域名

location.pathname 返回当前页面的路径和文件名

location.port 返回 web 主机的端口 （80 或 443）

location.protocol 返回所使用的 web 协议（http:// 或 https://）

### Location 对象属性

|  |  |
| --- | --- |
| 属性 | 描述 |
| [hash](http://www.w3school.com.cn/jsref/prop_loc_hash.asp) | 设置或返回从井号 (#) 开始的 URL（锚）。 |
| [host](http://www.w3school.com.cn/jsref/prop_loc_host.asp) | 设置或返回主机名和当前 URL 的端口号。 |
| [hostname](http://www.w3school.com.cn/jsref/prop_loc_hostname.asp) | 设置或返回当前 URL 的主机名。 |
| [href](http://www.w3school.com.cn/jsref/prop_loc_href.asp) | 设置或返回完整的 URL。 |
| [pathname](http://www.w3school.com.cn/jsref/prop_loc_pathname.asp) | 设置或返回当前 URL 的路径部分。 |
| [port](http://www.w3school.com.cn/jsref/prop_loc_port.asp) | 设置或返回当前 URL 的端口号。 |
| [protocol](http://www.w3school.com.cn/jsref/prop_loc_protocol.asp) | 设置或返回当前 URL 的协议。 |
| [search](http://www.w3school.com.cn/jsref/prop_loc_search.asp) | 设置或返回从问号 (?) 开始的 URL（查询部分）。 |

### Location 对象方法

|  |  |
| --- | --- |
| 属性 | 描述 |
| [assign()](http://www.w3school.com.cn/jsref/met_loc_assign.asp) | 加载新的文档。 |
| [reload()](http://www.w3school.com.cn/jsref/met_loc_reload.asp) | 重新加载当前文档。 |
| [replace()](http://www.w3school.com.cn/jsref/met_loc_replace.asp) | 用新的文档替换当前文档。 |

### Window Location Href

location.href 属性返回当前页面的 URL。

<script>

document.write(location.href);

</script>

以上代码输出为：

http://www.w3school.com.cn/js/js\_window\_location.asp

### Window Location Pathname

location.pathname 属性返回 URL 的路径名。

<script>

document.write(location.pathname);

</script>

以上代码输出为：

/js/js\_window\_location.asp

### Window Location Assign

location.assign() 方法加载新的文档。

window.location.assign("http://www.w3school.com.cn")

## History对象

window.history 对象在编写时可不使用 window 这个前缀。

为了保护用户隐私，对 JavaScript 访问该对象的方法做出了限制。

### Window History Back

history.back() 方法加载历史列表中的前一个 URL。

这与在浏览器中点击后退按钮是相同的：

window.history.back()

### Window History Forward

history forward() 方法加载历史列表中的下一个 URL。

这与在浏览器中点击前进按钮是相同的：

window.history.forward()

## Navigator对象

window.navigator 对象包含有关访问者浏览器的信息，在编写时可不使用 window 这个前缀。

<div id="example"></div>

<script>

txt = "<p>Browser CodeName: " + navigator.appCodeName + "</p>";

txt+= "<p>Browser Name: " + navigator.appName + "</p>";

txt+= "<p>Browser Version: " + navigator.appVersion + "</p>";

txt+= "<p>Cookies Enabled: " + navigator.cookieEnabled + "</p>";

txt+= "<p>Platform: " + navigator.platform + "</p>";

txt+= "<p>User-agent header: " + navigator.userAgent + "</p>";

txt+= "<p>User-agent language: " + navigator.systemLanguage + "</p>";

document.getElementById("example").innerHTML=txt;

</script>

**警告：**来自 navigator 对象的信息具有误导性，不应该被用于检测浏览器版本，这是因为：

navigator 数据可被浏览器使用者更改。

浏览器无法报告晚于浏览器发布的新操作系统。

## 浏览器检测

由于 navigator 可误导浏览器检测，使用对象检测可用来嗅探不同的浏览器。

由于不同的浏览器支持不同的对象，您可以使用对象来检测浏览器。例如，由于只有 Opera 支持属性 "window.opera"，您可以据此识别出 Opera。

例子：if (window.opera) {...some action...}

# AJAX

AJAX = 异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。

AJAX 是一种用于创建快速动态网页的技术，通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

通过 jQuery AJAX 方法，您能够使用 HTTP Get 和 HTTP Post 从远程服务器上请求文本、HTML、XML 或 JSON - 同时您能够把这些外部数据直接载入网页的被选元素中。

**提示：**如果没有 jQuery，AJAX 编程还是有些难度的。

## 创建 XMLHttpRequest 对象

XMLHttpRequest 是 AJAX 的基础。所有现代浏览器均支持 XMLHttpRequest 对象（IE5 和 IE6 使用 ActiveXObject）。

**XMLHttpRequest 用于在后台与服务器交换数据**。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

创建 XMLHttpRequest 对象的语法：

variable=new XMLHttpRequest();

老版本的 Internet Explorer （IE5 和 IE6）使用 ActiveX 对象：

variable=new ActiveXObject("Microsoft.XMLHTTP");

为了应对所有的现代浏览器，包括 IE5 和 IE6，请检查浏览器是否支持 XMLHttpRequest 对象。如果支持，则创建 XMLHttpRequest 对象。如果不支持，则创建 ActiveXObject ：

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

## 一个简单的get请求实例

<html>

<head>

<script type="text/javascript">

function loadXMLDoc()

{

var xmlhttp;

if (window.XMLHttpRequest)

{// code for IE7+, Firefox, Chrome, Opera, Safari

xmlhttp=new XMLHttpRequest();

}

else

{// code for IE6, IE5

xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");

}

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","/ajax/demo\_get.asp",true);

xmlhttp.send();

}

</script>

</head>

<body>

<h2>AJAX</h2>

<button type="button" onclick="loadXMLDoc()">请求数据</button>

<div id="myDiv"></div>

</body>

</html>

## 向服务器发送请求

如需将请求发送到服务器，我们使用 XMLHttpRequest 对象的 open() 和 send() 方法：

xmlhttp.open("GET","test1.txt",true);

xmlhttp.send();

|  |  |
| --- | --- |
| 方法 | 描述 |
| open(*method*,*url*,*async*) | 规定请求的类型、URL 以及是否异步处理请求。  *method*：请求的类型；GET 或 POST  *url*：文件在服务器上的位置  *async*：true（异步）或 false（同步） |
| send(*string*) | 将请求发送到服务器。  *string*：仅用于 POST 请求 |

### GET 还是 POST？

与 POST 相比，GET 更简单也更快，并且在大部分情况下都能用。

然而，在以下情况中，请使用 POST 请求：

1、无法使用缓存文件（更新服务器上的文件或数据库）

2、向服务器发送大量数据（POST 没有数据量限制）

3、发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

### GET 请求

一个简单的 GET 请求：

xmlhttp.open("GET","demo\_get.asp",true);

xmlhttp.send();

在上面的例子中，您可能得到的是缓存的结果。

为了避免这种情况，请向 URL 添加一个唯一的 ID：

xmlhttp.open("GET","demo\_get.asp?t=" + Math.random(),true);

xmlhttp.send();

如果您希望通过 GET 方法发送信息，请向 URL 添加信息：

xmlhttp.open("GET","demo\_get2.asp?fname=Bill&lname=Gates",true);

xmlhttp.send();

### POST 请求

一个简单 POST 请求：

xmlhttp.open("POST","demo\_post.asp",true);

xmlhttp.send();

如果需要像 HTML 表单那样 POST 数据，请使用 setRequestHeader() 来添加 HTTP 头。然后在 send() 方法中规定您希望发送的数据：

xmlhttp.open("POST","ajax\_test.asp",true);

xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");

xmlhttp.send("fname=Bill&lname=Gates");

|  |  |
| --- | --- |
| 方法 | 描述 |
| setRequestHeader(*header*,*value*) | 向请求添加 HTTP 头。  *header*: 规定头的名称  *value*: 规定头的值 |

### url - 服务器上的文件

open() 方法的 *url* 参数是服务器上文件的地址：

xmlhttp.open("GET","ajax\_test.asp",true);

该文件可以是任何类型的文件，比如 .txt 和 .xml，或者服务器脚本文件，比如 .asp 和 .php （在传回响应之前，能够在服务器上执行任务）。

### 异步 - True 或 False？

AJAX 指的是异步 JavaScript 和 XML，XMLHttpRequest 对象如果要用于 AJAX 的话，其 open() 方法的 async 参数必须设置为 true：

xmlhttp.open("GET","ajax\_test.asp",true);

对于 web 开发人员来说，发送异步请求是一个巨大的进步。很多在服务器执行的任务都相当费时。AJAX 出现之前，这可能会引起应用程序挂起或停止。

通过 AJAX，JavaScript 无需等待服务器的响应，而是：

1、在等待服务器响应时执行其他脚本

2、当响应就绪后对响应进行处理

### Async = true

当使用 async=true 时，请规定在响应处于 onreadystatechange 事件中的就绪状态时执行的函数：

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

xmlhttp.open("GET","test1.txt",true);

xmlhttp.send();

### Async = false

如需使用 async=false，请将 open() 方法中的第三个参数改为 false：

xmlhttp.open("GET","test1.txt",false);

我们不推荐使用 async=false，但是对于一些小型的请求，也是可以的。

请记住，JavaScript 会等到服务器响应就绪才继续执行。如果服务器繁忙或缓慢，应用程序会挂起或停止。

**注释：**当您使用 async=false 时，请不要编写 onreadystatechange 函数 - 把代码放到 send() 语句后面即可：

xmlhttp.open("GET","test1.txt",false);

xmlhttp.send();

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

## 服务器响应

如需获得来自服务器的响应，请使用 XMLHttpRequest 对象的 responseText 或 responseXML 属性。

|  |  |
| --- | --- |
| **属性** | **描述** |
| responseText | 获得字符串形式的响应数据。 |
| responseXML | 获得 XML 形式的响应数据。 |

### responseText 属性

如果来自服务器的响应并非 XML，请使用 responseText 属性。

responseText 属性返回字符串形式的响应，因此您可以这样使用：

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

### responseXML 属性

如果来自服务器的响应是 XML，而且需要作为 XML 对象进行解析，请使用 responseXML 属性：

请求 [books.xml](http://www.w3school.com.cn/example/xmle/books.xml) 文件，并解析响应：

xmlDoc=xmlhttp.responseXML;

txt="";

x=xmlDoc.getElementsByTagName("ARTIST");

for (i=0;i<x.length;i++)

{

txt=txt + x[i].childNodes[0].nodeValue + "<br />";

}

document.getElementById("myDiv").innerHTML=txt;

## onreadystatechange 事件

当请求被发送到服务器时，我们需要执行一些基于响应的任务。

每当 readyState 改变时，就会触发 onreadystatechange 事件。

readyState 属性存有 XMLHttpRequest 的状态信息。

下面是 XMLHttpRequest 对象的三个重要的属性：

|  |  |
| --- | --- |
| 属性 | 描述 |
| onreadystatechange | 存储函数（或函数名），每当 readyState 属性改变时，就会调用该函数。 |
| readyState | 存有 XMLHttpRequest 的状态。从 0 到 4 发生变化。  0: 请求未初始化  1: 服务器连接已建立  2: 请求已接收  3: 请求处理中  4: 请求已完成，且响应已就绪 |
| status | 200: "OK"  404: 未找到页面 |

在 onreadystatechange 事件中，我们规定当服务器响应已做好被处理的准备时所执行的任务。

当 readyState 等于 4 且状态为 200 时，表示响应已就绪：

xmlhttp.onreadystatechange=function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

}

注释：onreadystatechange 事件被触发 5 次（0 - 4），对应着 readyState 的每个变化。

## 使用 Callback 函数

callback 函数是一种以参数形式传递给另一个函数的函数。

如果您的网站上存在多个 AJAX 任务，那么您应该为创建 XMLHttpRequest 对象编写一个**标准**的函数，并为每个 AJAX 任务调用该函数。

该函数调用应该包含 URL 以及发生 onreadystatechange 事件时执行的任务（每次调用可能不尽相同）：

function myFunction()

{

loadXMLDoc("ajax\_info.txt",function()

{

if (xmlhttp.readyState==4 && xmlhttp.status==200)

{

document.getElementById("myDiv").innerHTML=xmlhttp.responseText;

}

});

}