# 基础

### js输出

window.alert() 弹出警告框

document.write() 方法将内容写到 HTML 文档中（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 写入到 HTML 元素

console.log() 写入到浏览器的控制台

### js大小写敏感

### js中undefined,null,NaN的区别

类型分析：

|  |
| --- |
| var a1;  var a2 = true;  var a3 = 1;  var a4 = "Hello";  var a5 = new Object();  var a6 = null;  var a7 = NaN;  var a8 = undefined;  alert(typeof a); //显示"undefined"  alert(typeof a1); //显示"undefined"  alert(typeof a2); //显示"boolean"  alert(typeof a3); //显示"number"  alert(typeof a4); //显示"string"  alert(typeof a5); //显示"object"  alert(typeof a6); //显示"object"  alert(typeof a7); //显示"number"  alert(typeof a8); //显示"undefined" |

结论：未定义和定义未赋值的类型为undefined；null是一种特殊的object，NaN是一种特殊的number。

比较运算：

|  |
| --- |
| var a1; //a1的值为undefined  var a2 = null;  var a3 = NaN;  alert(a1 == a2); //显示"true"  alert(a1 != a2); //显示"false"  alert(a1 == a3); //显示"false"  alert(a1 != a3); //显示"true"  alert(a2 == a3); //显示"false"  alert(a2 != a3); //显示"true"  alert(a3 == a3); //显示"false"  alert(a3 != a3); //显示"true" |

结论：定义未赋值的变量值为undefined，undefined和null是相等的，NaN与任何值都不等，与自己也不等

重点：undefined表示：1、未声明的变量，2、声明但没赋值的变量，3、不存在的对象属性

null表示：空值。

所以判断一个变量是否未定义，必须用===undefined，因为==undefined只判断值，不判断类型，null和undefined的值是相等的！

### js对象

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

调用属性： name=person.lastname;

name=person["lastname"];

### 局部变量和全局变量

局部变量：在函数内部使用var声明的变量，只在函数内使用,函数执行完销毁

全局变量：在函数外声明的，网页上的所有脚本和函数都能访问，页面关闭后销毁

这里跟php不同，函数可以不用传参，直接使用全局变量

若向未定义的变量赋值，该变量将自动作为全局变量

在 HTML 中, 全局变量是 window 对象: 所有数据变量都属于 window 对象。

|  |
| --- |
| person="John Doe";  function change(){  document.getElementById('demo').innerHTML=person;  document.getElementById('demo').innerHTML=window.person;  } |

### HTML事件

|  |  |
| --- | --- |
| **事件** | **描述** |
| onchange | 表单元素的内容改变时触发 |
| onclick | 用户点击 HTML 元素 |
| onmouseover | 用户在一个HTML元素上移动鼠标 |
| onmouseout | 用户从一个HTML元素上移开鼠标 |
| onkeydown | 用户按下键盘按键 |
| onload | 浏览器已完成页面的加载（用于body，页面图片及css等加载完触发） |

### 字符串

|  |
| --- |
| var carname = 'Volvo XC60';  //使用索引来访问字符串中的字符  var character = carname[7];  //使用length计算字符串长度  var sln = carname.length |

### 类型转换

typeof 操作符查看变量类型，他不是函数！

typeof ‘john’ string



请注意：

NaN 的数据类型是 number

数组(Array)的数据类型是 object

日期(Date)的数据类型为 object

null 的数据类型是 object

未定义变量的数据类型为 undefined

使用 constructor 属性来查看对象是否为数组

test.constructor

|  |
| --- |
| function isArray(myArray) {  return myArray.constructor.toString().indexOf("Array") > -1;  } |

全局方法：String（）转成字符串

|  |
| --- |
| String（123）; |

等同于Number方法toString()

|  |
| --- |
| var num =213;  var str = num.toString();  document.write(typeof str); |

类似的布尔、日期都有toString()方法

全局方法：Number() 将字符串转换为数字

|  |
| --- |
| Number(" ") // 返回 0  Number("") // 返回 0  Number("99 88") // 返回 NaN |

一元运算符 +

|  |
| --- |
| var y = "5"; // y 是一个字符串  var x = + y; // x 是一个数字  //+不能转换字符串,但x还是数字类型的  var y = "John"; // y 是一个字符串  var x = + y; // x 是一个数字 (NaN) |

布尔类型转数字

|  |
| --- |
| Number(false) // 返回 0  Number(true) // 返回 1 |

时间类型转数字

|  |
| --- |
| d = new Date();  Number(d) // 返回 1404568027739 |

自动转换：

|  |
| --- |
| 5 + null // 返回 5 because null is converted to 0  "5" + null // 返回"5null" because null is converted to "null"  "5" + 1 // 返回 "51" because 1 is converted to "1"  "5" - 1 // 返回 4 because "5" is converted to 5 |

自动转成字符串

当尝试输出一个对象或一个变量时，会自动调用变量的toString()方法；

|  |
| --- |
| document.getElementById("demo").innerHTML = myVar;  // if myVar = {name:"Fjohn"} // toString 转换为 "[object Object]"  // if myVar = [1,2,3,4] // toString 转换为 "1,2,3,4"  // if myVar = new Date() // toString 转换为 "Fri Jul 18 2014 09:08:55 GMT+0200" |

### 正则

stringObject.search(regex|string)

返回第一个与正则匹配到的子串的起始位置，无匹配返回-1；

|  |
| --- |
| var str = "Visit w3cschool";  var n = str.search(/w3cschool/i); js的正则表达式不加引号 |

参数也可以直接是字符串，字符串会转换成正则表达式。

此时，类似于字符串的indexOf()方法,返回首次出现的位置

|  |
| --- |
| var str = "Visit w3cschool!";  var n = str.search("w3cschool");  var n = str.indexOf("w3cschool"); |

stringObject.match(regex|string)

返回匹配到的结果的数组。

stringObjet.replace(regex,replacement)

返回替换后的新字符串

### 错误处理

* + 1. try catch

|  |
| --- |
| try  {  adddlert("Welcome guest!");  }  catch(err)  {  txt="本页有一个错误。\n\n";  txt+="错误描述：" + err.message + "\n\n";  txt+="点击确定继续。\n\n";  alert(txt);  } |

* + 1. throw

throw会抛出错误，在catch里会捕捉到错误err

|  |
| --- |
| try {  var x=*document*.getElementById("demo").value;  if(x=="") throw "empty";  if(isNaN(x)) throw "not a number";  if(x>10) throw "too high";  if(x<5) throw "too low"; } catch(*err*) {  var *y*=*document*.getElementById("mess");  *y*.innerHTML="Error: x is " + *err* + "."; } |

### 调试

* + 1. 断点

调试模式的sources里，在页面的左边栏行号处点击，即为设置断点。

* + 1. debugger

在js里添加debugger，效果跟在调试模式的断点一样。

|  |
| --- |
| var x = 15 \* 5;  debugger;  document.getElementbyId("demo").innerHTML = x; |

### 表单验证



### Json

json语法规则：

数据为 键/值 对。

数据由逗号分隔。

大括号保存对象

方括号保存数组

example:

|  |
| --- |
| {"employees":[  {"firstName":"John", "lastName":"Doe"},  {"firstName":"Anna", "lastName":"Smith"},  {"firstName":"Peter", "lastName":"Jones"}  ]}  在以上实例中，对象 "employees" 是一个数组。包含了三个对象。  每个为个对象为员工的记录（姓和名）。 |

json字符串转换为javascript对象

|  |
| --- |
| var obj = JSON.parse(text); |



js中用 . 来访问对象属性，用 [] 访问数组元素。

### javascript:void(0)

void：计算一个表达式但是不返回值

|  |
| --- |
| function getValue(){   var a,b,c;   a = void ( b = 5, c = 7 );   document.write('a = ' + a + ' b = ' + b +' c = ' + c ); }  void不返回值，所以a没有赋值  a = undefined b = 5 c = 7 |

定义一个死链接：

<a href="javascript:void(0);">

# 函数

### 提升

提升（Hoisting）是 JavaScript 默认将当前作用域提升到前面去的的行为。

所以，函数可以在声明之前就调用，没有顺序的问题。

|  |
| --- |
| myFunction(5);  function myFunction(y) {  return y \* y;  } |

### 自调用函数

|  |
| --- |
| (function () {  var x = "Hello!!"; // 我将调用自己  })();  前面括号内是个匿名函数，它就等于一个函数名，后面加个括号等于调用，也可以传递参数。 |

### arguments对象

JavaScript 函数有个内置的对象 arguments 对象.

arguments 对象包含了函数调用的参数数组。

arguments.length 参数个数

arguments[i] 第几个参数

### 函数调用

每种方式的不同在于 this 的初始化。

在js中，this指向函数执行时的当前对象。

* + 1. 作为一个函数调用

|  |
| --- |
| function myFunction() {     alert(this); } myFunction();           // [object Window]  以上函数不属于任何对象，但在js中它始终属于默认的全局对象；  在 HTML 中默认的全局对象是 HTML 页面本身，所以函数是属于 HTML 页面。  在浏览器中的页面对象是浏览器窗口(window 对象)。以上函数会自动变为 window 对象的函数。 所以这里this指的就是window  myFunction() 和 window.myFunction() 是一样的： |

* + 1. 作为对象方法调用