# 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

|  |
| --- |
| var myObject = {  firstName:"John",  lastName: "Doe",  fullName: function () {  return this.firstName;  }  }  myObject.fullName(); // 返回 "John Doe" |

fullName 属于myObject对象，所以this即为myObject对象

### js闭包

函数内可以访问函数外的变量

|  |
| --- |
| var a = 4;  function myFunction() {  return a \* a;  }  a 是全局变量，在web页面中全局变量属于window对象，全局变量可应用于页面上的所有脚本。  如果变量声明不用var，则它就是一个全局变量，无论是在函数里还是函数外！ |

# DOM

### 概述

DOM树：包含文档（document）、元素（element）、属性（attribute）



查找HTML元素：

1. var x=document.getElementById("intro");
2. var x=document.getElementsByName(‘name’) 返回对象集合
3. var y=x.getElementsByTagName("p");返回对象集合
4. var x=document.getElementsByClassName("intro"); 返回对象集合

### 改变HTML内容

document.write(Date());

document.getElementById(id).innerHTML=new HTML

改变html元素属性

document.getElementById(id).attribute=new value attribute此处指具体的元素属性

### 改变CSS

document.getElementById(id).style.property=new style

style使用3.2中的attribute获取属性，后面的property继续获取具体的属性值

### DOM事件

1）事件属性：

|  |
| --- |
| <button onclick="displayDate()">Try it</button> |

这里，onclick类似于style、id等，也属于元素的属性，只不过是事件类型的属性；

2）通过DOM分配事件

等同于通过DOM给元素添加了事件属性；

|  |
| --- |
| <script>  document.getElementById("myBtn").onclick=function(){displayDate()};  </script> |

3）常用事件

onclick

onload页面加载完触发

onchange改变输入字段的内容时触发

onmouseover鼠标移至元素上触发

onmouseout鼠标移除元素时触发

onmousedown鼠标按下时触发

onmouseup鼠标松开时触发

onclick 其实在鼠标松开时先触发onclick，后触发onmouseup。

onfocus 当元素获取焦点时触发

### 事件监听

element.addEventListener(event, function, useCapture);

第一个参数是事件的类型 (如 "click" 或 "mousedown").

第二个参数是事件触发后调用的函数。

第三个参数是个布尔值用于描述事件是冒泡（默认）（false）还是捕获（true）。该参数是可选的。

注意:不要使用 "on" 前缀。 例如，使用 "click" ,而不是使用 "onclick"。

冒泡：内部元素的事件先被触发，后触发外部元素的事件

捕获：外部元素的事件先被触发，后触发内部元素的事件

该属性只是添加监听，不会覆盖已存在的监听；

可以添加多个同类的事件监听，比如两个click事件

可以向任何DOM元素添加事件监听，不仅仅是HTML元素，还可以是window对象等

### DOM元素

1）创建新的 HTML 元素：

先创建该元素，然后向一个已存在的元素追加该元素。

|  |
| --- |
| //创建p元素  var para=document.createElement("p");  //创建文本节点  var node=document.createTextNode("This is a new paragraph.");  //将文本节点追加进p元素内  para.appendChild(node);  //获取div1元素  var element=document.getElementById("div1");  //将填充了文本的p元素追加进已经存在的div1元素中  element.appendChild(para); |

2）删除已有的HTML元素

|  |
| --- |
| <div id="div1">  <p id="p1">This is a paragraph.</p>  <p id="p2">This is another paragraph.</p>  </div>  <script>  var parent=document.getElementById("div1");  var child=document.getElementById("p1");  //先将父子两个元素获取到  parent.removeChild(child);  </script> |

注意：js中删除元素时必须先找到该元素的父元素，使用removeChild删除子元素。不能 不引用父元素的情况下直接删除子元素。

但是 ，可以先将子元素获取到，使用parentNode获取他的父元素：

|  |
| --- |
| var child=document.getElementById("p1");  child.parentNode.removeChild(child); |

# 对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数... 对象只是带有属性和方法的特殊数据类型。

访问对象属性：objectName.propertyName

访问对象方法：objectName.methodName()

### 创建直接的对象实例

|  |
| --- |
| person=new Object();  person.firstname="John";  person.lastname="Doe";  person.age=50;  person.eyecolor="blue"; |

### 使用对象构造器

|  |
| --- |
| function person(firstname,lastname,age,eyecolor)  {  this.firstname=firstname;  this.lastname=lastname;  this.age=age;  //对象的方法  this.changeName=function (name)  {  this.lastname=name;  };  } |

这里的this指的是正在执行的函数本身；

运行时，this一般用在对象的方法中，指向调用该方法的对象；

|  |
| --- |
| var oCar = new object();  cCar.showColor = function(){alert(this.color);};//this就指的oCar对象 |

有了对象构造器，可以这样创建对象：

var myFather=new person("John","Doe",50,"blue");

### Number对象

1. Infinity 无穷大
2. - Infinity 无穷小
3. NaN 非数字值
4. isNaN() 一个值不是数字时返回true

### 字符串对象

1）属性：length 字符串长度

text.length

以下都是方法：

2）indexOf（char）字符串中某个指定的字符char首次出现的位置

|  |
| --- |
| var str = "Hello world, welcome to the universe."; |
| str.indexOf(’world’) |

返回：索引下标，未找到返回-1；

3）lastIndexOf(char) 在字符串的末尾开始查找某字符出现的位置

4）match（char）

str.match("world") 同‘正则’章节相同，返回匹配到的字符；

1. replace（char,replacement）替换字符
2. toUpperCase() / toLowerCase() 大小写转换

str.toUpperCase();

1. split(separator) 将字符串以separator分割为数组；

|  |
| --- |
| txt="a,b,c,d,e" // String  txt.split(","); // Split on commas |

### 日期对象

1. var d = new Date() 获得当日的日期
2. d. getFullYear() 获取年份
3. d. getTime() 获取Unix时间戳
4. d.setFullYear(2020,10,3); 设置具体的日期 //js里月份是0~11,0即是1月
5. d.getDay() 返回星期几 0-星期日 1-星期一…
6. d. getDate() 返回月份天数
7. 日期比较

|  |
| --- |
| var x=new Date();  x.setFullYear(2100,0,14);  var today = new Date();  if (x>today)  {  alert("Today is before 14th January 2100");  }  else  {  alert("Today is after 14th January 2100");  } |

1. 显示时钟

|  |
| --- |
| function time(){  var d = new Date();  var h = d.getHours();  var m = d.getMinutes();  var s = d.getSeconds();  m = checkTime(m);  s = checkTime(s);  document.getElementById('demo').innerHTML = h +':' +m+':'+s;  setTimeout("time()",500); //这里调用函数一定要加引号！！！  }  function checkTime(k) {  if(k<10){  return '0'+k;  }  return k;  } |

### 数组对象

1. 创建数组

|  |
| --- |
| var myCars=["Saab","Volvo","BMW"];  或var myCars=new Array("Saab","Volvo","BMW"); |

1. 访问数组

var name=myCars[0];

3）方法与属性

arr.length 数组元素的个数

arr.indexOf(‘a’) a在数组arr中的下标

### 布尔对象

|  |
| --- |
| var myBoolean=new Boolean(); |

如果布尔对象无初始值或者其值为:

0

-0

null

""

false

undefined

NaN

那么对象的值为 false。否则，其值为 true（即使当自变量为字符串 "false" 时）！

var myBoolean = new Boolean(‘false’);

### Math算数对象

1. Math.round（2.5） 四舍五入
2. Math.random(); 返回 0 到 1 之间的随机数。
3. Math.max(1,3,5,7) 返回几个数中最大数
4. Math.min(2,4,6,8) 返回几个数中的最小数

# Window

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"); |

获取浏览器窗口大小：（不同浏览器不同的方法）

|  |
| --- |
| var w=window.innerWidth  || document.documentElement.clientWidth  || document.body.clientWidth;  var h=window.innerHeight  || document.documentElement.clientHeight  || document.body.clientHeight; |

### 常用方法

window.open(url) - 打开新窗口

window.close() - 关闭当前窗口

window.moveTo() - 移动当前窗口

window.resizeTo() - 调整当前窗口的尺寸

### Window Screen

window.screen 对象包含有关用户屏幕的信息。编写时可以不使用 window 这个前缀。

screen.availWidth - 可用的屏幕宽度

screen.availHeight - 可用的屏幕高度

### window.location

window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

使用时也可以不加window前缀。

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.href 返回当前页面的URL

若给location.href赋值，即location.href = 'http://www.baidu.com' 则是重定向到新url；

等同于location.assign('http://www.baidu.com') //加载新文档

### Window History

window.history 对象包含浏览器的历史。编写时可以不使用 window 这个前缀。

history.back() - 与在浏览器点击后退按钮相同

history.forward() - 与在浏览器中点击按钮向前相同

### 弹窗

1. alert（）
2. confirm（提示信息） 当点击确认，返回true，点击取消，返回false
3. prompt（"sometext","defaultvalue"）

一般用于询问，提示框出现后，你需要输入一个值，然后点击确定或取消，当点击确定后，prompt返回你的输入值，点取消时返回null

### 计时事件

以下是Window对象的两个方法

setInterval（"javascript function",milliseconds）

间隔指定的毫秒数，不停地执行function的代码；

注意：这里的函数，必须加引号！

clearInterval（interval）停止setInterval()执行

|  |
| --- |
| myVar=setInterval("javascript function",milliseconds);  clearInterval(myVar); |

setTimeout（"javascript 函数",毫秒数）

一定时间后执行函数；

clearTimeout（）若上面的延时还未运行，使用此方法停止

|  |
| --- |
| myVar=setTimeout("javascript function",milliseconds);  clearTimeout(myVar); |

### js cookie