JavaScript阶段总结

1.变量声明使用var 关键字；函数和自定义对象声明使用function关键字；

2.javascript---表示行为；CSS---表示显示；HTML表示结构

3.JavaScript与Java不同点：

基于对象和面向对象

解释和编译

强变量和弱变量

代码格式不一样

嵌入方式不一样

静态联编和动态联编（对象引用在运行时进行检查，对象引用必须在编译时的进行）

4.在HTML中定义脚本使用<script>标签：例如：

A .<script type="text/javascript">....</script>

B .<script language="javascript">....</script> ------h5不建议使用

5.在HTML中引入外部的js文件，同样使用<script>标签，例如：

<script type="text/javascript" src="../../js/JsDemo.js">**</script> ------必须要有结束标签**

6.javascript中的注释有：

// ------表示行注释；

/\*....\*/ ------表示块注释。

7.document.write（"...."）---表示向当前HTML页面写入内容

8.javascript中三种对话框分别是：

alert() ---一个按钮

confirm() ---两个按钮，得到的结果是true或false

prompt() ---两个按钮，可以接受用户输入

9.关闭一个浏览器窗口，两种方式：

A.<a href="javascript:self.close()">关闭窗口</a>

B.通过事件，调用window.close（）方法：

<script >

    function NVGClose() {

**window.close();**

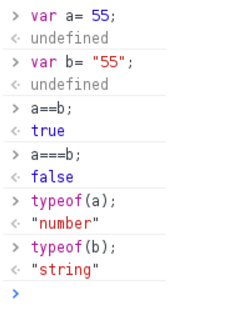
   }

</script>

<input type="button" value="关闭" onclick=“NVGClose()">

10.声明变量可以用：

   var <变量名> [= <值>]



说明：= 是赋值；**==是判断值的内容是否相等；===是判断值的内容与类型是否相等！**

11.javascript中的foreach循环----for in 循环：

// 创建具有某些属性的对象

  var myObject = new Object();

  myObject.name = "James";

  myObject.age = "22";

  myObject.phone = "555 1234";

  // 枚举（循环）对象的所有属性

**for (prop in myObject){**   // 显示 "The property 'name' is James"，等等。

      window.alert("The property '" + prop + "' is " + myObject[prop]);

**}**

12.声明函数：

function 函数名 （参数,变元）{

函数体;

return 表达式;

}

13.获取当前函数的参数个数：

var num = function\_Name.arguments.length;

14.错误处理：-----和java中类似

function Age()

  {

     try {

   var m="age";

   var n=20;

    document.write(parseInt(m)+n);

  //抛出语句

   throw new Error("not a valid number");

     }  catch (errMsg)     {

          alert(errMsg.message);

      }

 }

15.javascript中的对象相比java对象，多了**事件属性**

**调用属性和方法也和java中类似。**

16.Microsoft Jscript 提供了 11 个内部（或“内置”）对象。

它们是Array、Boolean、Date、Function、Global、Math、Number、Object、RegExp、Error 以及 String 对象。.

查看这些对象的方法或属性：可以通过在浏览器的控制台（开发者模式下），**直接敲击类名或者对象名. 可以列出存在的属性和方法。**

17.JavaScript数组定义几种方式：

方式一：

var <数组名> = new Array();

这样就定义了一个空数组。以后要添加数组元素，就用：

<数组名>[<下标>] = ...;

注意这里的方括号不是“可以省略”的意思，数组的下标表示方法就是用方括号括起来。

方式二：

  var <数组名> = new Array(n);  // n 是数组长度

方式三：

如果想在定义数组的时候直接初始化数据，请用：

var <数组名> = new Array(<元素1>, <元素2>, <元素3>...);

例如，var myArray = new Array(1, 4.5, ‘Hi’); 定义了一个数组 myArray，里边的元素是：myArray[0] == 1; myArray[1] == 4.5; myArray[2] == ‘Hi’。

但是，如果元素列表中只有一个元素，而这个元素又是一个正整数的话，这将定义一个包含<正整数>个空元素的数组。

方式四：

  var myArray = [“first”,”second”,”third”];

说明：javascript中的数组和java中不同的是：

A.数组长度可以变化----这一点和List类似

B.数组元素并不要求是同一数据类型----这一点和python中类似。

18.javascript中创建对象也是使用new关键字，如：var d = new Date();

这个方法使 d 成为日期对象，并且已有初始值：当前时间。

var d = new Date(99, 10, 1);       //99 年 10 月 1 日

  var d = new Date('Oct 1, 1999');   //99 年 10 月 1 日

19.全局对象从不现形，它可以说是虚拟出来的，目的在于把全局函数“对象化”。在 Microsoft JScript 语言参考中，它叫做“Global 对象”，但是引用它的方法和属性从来不用“Global.xxx”（况且这样做会出错），而直接用“xxx”。

**eval()** ----把括号内的字符串当作标准语句或表达式来运行。

**isFinite()** ----如果括号内的数字是“有限”的（介于 Number.MIN\_VALUE 和 Number.MAX\_VALUE 之间）就返回 true；否则返回 false。

**isNaN()** ----如果括号内的值是“NaN”则返回 true 否则返回 false。NaN --- Not a Number

**parseInt()** ----返回把括号内的内容转换成整数之后的值。如果括号内是字符串，则字符串开头的数字部分被转换成整数，如果以字母开头，则返回“NaN”。

**parseFloat()** ----返回把括号内的字符串转换成浮点数之后的值，字符串开头的数字部分被转换成浮点数，如果以字母开头，则返回“NaN”。

**toString()** ----用法：<对象>.toString()；把对象转换成字符串。如果在括号中指定一个数值，则转换过程中所有数值转换成特定进制。

20.JavaScript对文字进行编码：

涉及3个函数：escape,encodeURI,encodeURIComponent，

相应3个解码函数：unescape,decodeURI,decodeURIComponent。

关于URL编码/javascript/js url 编码/url的三个js编码函数 - vfvfb\_csdn\_我的地盘 - 博客频道 - CSDN.NET http://blog.csdn.net/vfvfb/article/details/7770409

21.javascript中自定义构造函数：例如：

下面的示例为 pasta 对象定义了构造函数。

注意 **this 关键字的使用，它指向当前对象。**

// pasta 是有四个参数的构造器。

function pasta(grain, width){

    this.grain = grain;    // 是用什么粮食做的？

    this.width = width;  // 多宽？（数值）

    this.toString = pastaToString;

    // 这里添加 toString 方法（如下定义）。

    // **注意在函数的名称后没有加圆括号；**

    // 这不是一个函数调用，而是对函数自身的引用。

}

function pastaToString()

{

    // 返回对象的属性。

    return "Grain: " + this.grain + "\n" +

        "Width: " + this.width + "\n" ;

}

22.对于自定义对象的构造函数，在创建对象时，多指定参数时不会起作用。

23.expando属性，可以只为每个对象单例添加属性，其他个体不会有，有点类似java中的继承。

spaghetti.color = "pale straw";

 myObject["not a valid identifier"]= "This is the property value";

myObject[100] ="100";

myArray.expando = "JScript!";// 添加某些 expando 属性

myArray["another Expando"] = "Windows"; //因为两个 expando 属性，并不影响长度

24.原型对象：

如果要将对象所有实例的附加属性显示出来，必须将它们添加到构造函数或构造器原型对象中。例如：

pasta.**prototype**.foodgroup = "carbohydrates"

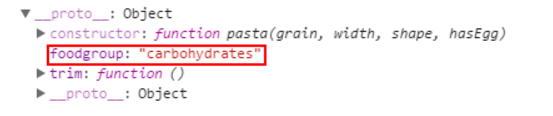
String.prototype.trim = function(){

    // 用正则表达式将前后空格用空字符串替代。

    return this.replace(/(^\s\*)|(\s\*$)/g, "");

}

说明：添加后的属性或方法，就像原有方法一样使用，但是，属性位置并不是在当前对象下，而是在当前对象下的



25.with语句使用：通常用来缩短特定情形下必须写的代码量。例如：

请注意 Math 的重复使用：

  x = Math.cos(3 \* Math.PI) + Math.sin(Math.LN10);

  y = Math.tan(14 \* Math.E);

当使用 with 语句时，代码变得更短且更易读：

**with (Math) {**

**x = cos(3 \* PI) + sin(LN10);**

**y = tan(14 \* E);**

**}**

26.浏览器的内部对象系统（宿主对象）：----可以在浏览器控制台查看属性和方法

**浏览器对象(Navigator)** ----提供有关浏览器的信息

**屏幕对象(screen)** ----反映了当前用户的屏幕设置

**窗口对象(Window)** ----Window对象处于对象层次的最顶端，它提供了处理Navigator窗口的方法和属性。

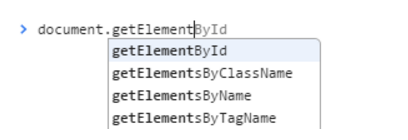
**位置对象(Location)** -----Location对象提供了与当前打开的URL一起工作的方法和属性，它是一个静态的对象。

**历史对象(History)** -----History对象提供了与历史清单有关的信息。

**文档对象(Document)** -----document对象包含了与文档元素(elements)一起工作的对象，它将这些元素封装起来供编程人员使用。



27.javascript中获取页面中的标签元素的方法有：



28.DOM----Document Object Model

29.HTML DOM 节点

在 HTML DOM (Document Object Model) 中 , 每一个元素都是 节点:

* 文档是一个文档。
* 所有的HTML元素都是元素节点。
* 所有 HTML 属性都是属性节点。
* 文本插入到 HTML 元素是文本节点。are text nodes。
* 注释是注释节点。

30.Document 对象

当浏览器载入 HTML 文档, 它就会成为 document 对象。

document 对象是HTML文档的根节点与所有其他节点（元素节点，文本节点，属性节点, 注释节点）。

Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。

提示：Document 对象是 Window 对象的一部分，可通过 window.document 属性对其进行访问。

31.节点父、子和同胞

节点树中的节点彼此拥有层级关系。

父（parent）、子（child）和同胞（sibling）等术语用于描述这些关系。父节点拥有子节点。同级的子节点被称为同胞（兄弟或姐妹）。

* 在节点树中，顶端节点被称为根（root）
* 每个节点都有父节点、除了根（它没有父节点）
* 一个节点可拥有任意数量的子节点
* 同胞是拥有相同父节点的节点

32.HTML DOM 对象 - 方法和属性

一些常用的 HTML DOM 方法：

* getElementById(id) - 获取带有指定 id 的节点（元素）
* appendChild(node) - 插入新的子节点（元素）
* removeChild(node) - 删除子节点（元素）

一些常用的 HTML DOM 属性：

* innerHTML - 节点（元素）的文本值
* parentNode - 节点（元素）的父节点
* childNodes - 节点（元素）的子节点
* attributes - 节点（元素）的属性节点

一些 DOM 对象方法

这里提供一些您将在本教程中学到的常用方法：

|  |  |
| --- | --- |
| **方法** | **描述** |
| getElementById() | 返回带有指定 ID 的元素。 |
| getElementsByTagName() | 返回包含带有指定标签名称的所有元素的节点列表（集合/节点数组）。 |
| getElementsByClassName() | 返回包含带有指定类名的所有元素的节点列表。 |
| appendChild() | 把新的子节点添加到指定节点。 |
| removeChild() | 删除子节点。 |
| replaceChild() | 替换子节点。 |
| insertBefore() | 在指定的子节点前面插入新的子节点。 |
| createAttribute() | 创建属性节点。 |
| createElement() | 创建元素节点。 |
| createTextNode() | 创建文本节点。 |
| getAttribute() | 返回指定的属性值。 |
| setAttribute() | 把指定属性设置或修改为指定的值。 |

33.innerHtml和InnerText的区别

return false；//抑制默认行为

命名空间；//防止命名冲突