1.javascript直接写入html输出流document.write();

document.write是直接写入到页面的内容流，如果在写之前没有调document.open, 浏览器会自动调用open。每次写完关闭之后重新调用该函数，会导致页面被重写。

2.onclick() 点击事件；

3.innerHTML() 改变html文本内容

4. JavaScript 是所有现代浏览器以及 HTML5 中的默认脚本语言。

5. JavaScript 可以通过不同的方式来输出数据：

* 使用 **window.alert()** 弹出警告框。
* 使用 **document.write()** 方法将内容写到 HTML 文档中。
* 使用 **innerHTML** 写入到 HTML 元素。
* 使用 **console.log()** 写入到浏览器的控制台。

6. JavaScript 对大小写是敏感的。JavaScript 中，常见的是驼峰法的命名规则

7. 通过 **typeOf()** 来获取JavaScript中变量的数据类型.

8. 在 JavaScript 中，用分号来结束语句是可选的。

9. 您可以在文本字符串中使用反斜杠对代码行进行换行

10. 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。如果重新声明 JavaScript 变量，该变量的值不会丢失

11. JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型

12. 对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。var person={firstname:"John", lastname:"Doe", id:5566};

对象属性有两种寻址方式：person.lastname / person[‘lastname’]

13. Undefined 这个值表示变量不含有值。可以通过将变量的值设置为 null 来清空变量。

14. JavaScript 变量均为对象。当您声明一个变量时，就创建了一个新的对象。

当您声明新变量时，可以使用关键词 "new" 来声明其类型

15. 对象的方法定义了一个函数，并作为对象的属性存储。对象方法通过添加 () 调用 (作为一个函数)。

16**调用带参数的函数:** 变量和参数必须以一致的顺序出现。

17.带有返回值的函数：使用 return 语句时，函数会停止执行，并返回指定的值给调用此函数的值。

在您仅仅希望退出函数时 ，也可使用 return 语句

18 在 JavaScript 函数内部声明的变量是*局部*变量，所以只能在函数内部访问它。

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。

在函数内部，有var和没var声明的变量是不一样的。有var声明的是局部变量，没var的，声明的全局变量

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

19 常见的HTML事件

onclick 用户点击 HTML 元素

onchange HTML 元素改变

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

20 = 赋值

== 相等

===为绝对相等，即数据类型与值都必须相等。

21.字符串属性

constructor 返回创建字符串属性的函数

length 返回字符串的长度

prototype 允许您向对象添加属性和方法

常见字符串方法：

charAt() 返回指定索引位置的字符

concat() 连接两个或多个字符串，返回连接后的字符串

indexOf() 返回字符串中检索指定字符第一次出现的位置

match() 找到一个或多个正则表达式的匹配

replace() 替换与正则表达式匹配的子串

slice() 提取字符串的片断，并在新的字符串中返回被提取的部分

split() 把字符串分割为子字符串数组

substring() 提取字符串中两个指定的索引号之间的字符

toString() 返回字符串对象值

trim() 移除字符串首尾空白

valueOf() 返回某个字符串对象的原始值

更多方法实例可以参见：[JavaScript String 对象](http://www.runoob.com/jsref/jsref-obj-string.html)。

22.Javascript的switch语句

使用 **break**来阻止代码自动地向下一个 case 运行。

使用 default 关键词来规定匹配不存在时做的事情

23. JavaScript 支持不同类型的循环：

* **for** - 循环代码块一定的次数
* **for/in** - 循环遍历对象的属性
* **while** - 当指定的条件为 true 时循环指定的代码块
* **do/while** - 同样当指定的条件为 true 时循环指定的代码块

24.JavaScript中：string与null 使用typeof返回都是Object

值为 undefined(空), 类型是undefined

25.JavaScript类型转换：

Number() 转换为数字， String() 转换为字符串， Boolean() 转化为布尔值。

26.常用的Date方法：

getDate() 从 Date 对象返回一个月中的某一天 (1 ~ 31)。

get+Day / Hours / Minutes / Month / Seconds / Time() 从 Date 对象返

getFullYear() 从 Date 对象以四位数字返回年份。

toLocaleTimeString() 根据本地时间把 Date 对象的时间部分转换为字符串

toGMTString() 根据格林威治时间 (GMT) 把 Date 对象转换为字符串，并返回结果。 toUTCString() 根据世界时间

更多关于日期转换为字符串的函:  [Date 方法](http://www.runoob.com/jsref/jsref-obj-date.html)

27.字符串转换成数字

parseFloat() 解析一个字符串，并返回一个浮点数

parseInt() 解析一个字符串，并返回一个整数。

28.JavaScript 的正则表达式

在 JavaScript 中，正则表达式通常用于两个字符串方法 : search() 和 replace()。

**search() 方法** 用于检索字符串中指定的子字符串，或检索与正则表达式相匹配的子字符串，并返回子串的起始位置。

**replace() 方法** 用于在字符串中用一些字符替换另一些字符，或替换一个与正则表达式匹配的子串。

完整的 RegExp 对象参考手册，请参考我们的 [JavaScript RegExp 参考手册](http://www.runoob.com/jsref/jsref-obj-regexp.html)。

正则表达式修饰符：

i 执行对大小写不敏感的匹配。

g 执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

m 执行多行匹配。

正则表达式模式：

\d 查找数字。

\s 查找空白字符。

\b 匹配单词边界

n+ 匹配任何包含至少一个 *n* 的字符串。

n\* 匹配任何包含零个或多个 *n* 的字符串。

n? 匹配任何包含零个或一个 *n* 的字符串。

test() 方法用于检测一个字符串是否匹配某个模式，如果字符串中含有匹配的文本，则返回 true，否则返回 false。

exec() 方法用于检索字符串中的正则表达式的匹配。该函数返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null。

29.JavaScript 错误 try catch throw

**try** 语句测试代码块的错误。

**catch** 语句处理错误。

**throw** 语句创建自定义错误。

e.message 获取错误详情；

**debugger** 关键字用于停止执行 JavaScript，并调用调试函数,与浏览器调试工具设置断点功能一样

30 JavaScript 严格模式“use strict”

严格模式作用域里不允许使用未声明的变量；

不允许删除变量或对象；

不允许删除函数；

不允许变量重名；

不允许使用八进制；

不允许使用转移字符；

不允许对只读属性赋值；

不允许对一个使用getter()方法读取的属性赋值；

不允许删除一个不允许删除的属性；

变量名不允许使用‘eval / arguments‘字符串；

由于一些安全原因，在作用域 eval() 创建的变量不能被调用

禁止this关键字指向全局对象。

为了向将来Javascript的新版本过渡，严格模式新增了一些保留关键字：

implements interface let package private protected public

static yield

31 JavaScript 常见使用误区：

switch 语句会使用恒等计算符(===)进行比较；

JavaScript 中的所有数据都是以 64 位**浮点型数据(float)** 来存储。

字符串断行需要使用反斜杠(\)

JavaScript 中，分号是可选的 。

JavaScript 不支持使用名字来索引数组，只允许使用数字索引。

在 JavaScript 中, **对象** 使用 **名字作为索引**。如果你使用名字作为索引，当访问数组时，JavaScript 会把数组重新定义为标准对象。

定义数组元素 ／ 对象 ，最后不能添加**，**

undefined 不是 NULL **null** 用于对象, **undefined** 用于变量，属性和方法。

32.JavaScript表单

约束验证HTML输入属性：

disabled 规定输入的元素不可用

max ／ min 规定输入元素的最大值

pattern 规定输入元素值的模式

required 规定输入元素字段是必需的

type 规则输入元素的类型

完整列表，请查看 [HTML 输入属性](http://www.runoob.com/html/html5-form-attributes.html)。

约束验证CSS伪类选择器

:disabled 选取属性为 "disabled" 属性的 input 元素

:invalid 选取无效的 input 元素

:optional 选择没有"required"属性的 input 元素

:required 选择有"required"属性的 input 元素

:valid 选取有效值的 input 元素

完整列表，请查看 [CSS 伪类](http://www.runoob.com/css/css-pseudo-classes.html)。

约束验证DOM方法

checkValidity() 如果 input 元素中的数据是合法的返回 true，否则返回 false。

setCustomValidity() 设置 input 元素的 validationMessage 属性，用于自定义错误提示信息的方法。使用 setCustomValidity 设置了自定义提示后，validity.customError 就会变成true，则 checkValidity 总是会返回false。

约束验证DOM属性

validity 布尔属性值，返回 input 输入值是否合法

validationMessage 浏览器错误提示信息

willValidate 指定 input 是否需要验证

input 元素的 **validity 属性**包含一系列关于 validity 数据属性:

customError 设置为 true, 如果设置了自定义的 validity 信息。

patternMismatch 设置为 true, 如果元素的值不匹配它的模式属性。

rangeOverlow / rangeUnderlow 设置为 true, 如果元素的值大于/小于设置的最大值。

stepMismatch 设置为 true, 如果元素的值不是按照规定的 step 属性设置。

tooLong 设置为 true, 如果元素的值超过了 maxLength 属性设置的长度。

typeMismatch 设置为 true, 如果元素的值不是预期相匹配的类型。

valueMissing 设置为 true，如果元素 (required 属性) 没有值。

valid 设置为 true，如果元素的值是合法的。

33. JavaScript – JSON  [JSON 教程](http://www.runoob.com/json/json-tutorial.html)

JSON 使用 JavaScript 语法，但是 JSON 格式仅仅是一个文本。文本可以被任何编程语言读取及作为数据格式传递。

JSON语法格式：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

相关函数：

JSON.parse() 用于将一个 JSON 字符串转换为 JavaScript 对象。

JSON.stringify() 用于将 JavaScript 值转换为 JSON 字符串。

34.JavaScript:viod(0)

void 是 JavaScript 中非常重要的关键字，该操作符指定要计算一个表达式但是不返回值。

35.href=’#’与href=’javascript:void(0)’的区别

**#** 包含了一个位置信息，默认的锚是**#top** 也就是网页的上端。

而javascript:void(0), 仅仅表示一个死链接。

36.JavaScript的函数

分号是用来分隔可执行JavaScript语句。 由于函数声明不是一个可执行语句，所以不以分号结束。

函数同样可以通过内置的 JavaScript 函数构造器（Function()）定义。

var myFunction = new Function("a", "b", "return a \* b");

函数表达式可以 "自调用"。自调用表达式会自动调用。如果表达式后面紧跟 () ，则会自动调用。

(function () {  
    var x = "Hello!!";      // 我将调用自己  
})();

函数可以做为一个值被使用。

toString()可以把一个函数当字符串返回.

如果函数在调用时未提供隐式参数，参数会默认设置为： **undefined**

37.JavaScript函数的方法

call() / apply() 是预定义的函数方法。 两个方法可用于调用函数，两个方法的第一个参数必须是对象本身。两者的区别在于第二个参数： apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入

在 JavaScript 严格模式(strict mode)下, 在调用函数时第一个参数会成为 **this** 的值， 即使该参数不是一个对象。

全局和局部变量即便名称相同，它们也是两个不同的变量。修改其中一个，不会影响另一个的值。

在 JavaScript 中，所有函数都能访问它们上一层的作用域。JavaScript 支持嵌套函数。嵌套函数可以访问上一层的函数变量。

38.attributes:获取一个属性作对象

getAttribute：获取某一个属性的值  
setAttribute：建立一个属性，并同时给属性捆绑一个值  
createAttribute：仅建立一个属性  
removeAttribute：删除一个属性

getAttributeNode：获取一个节点作为对象  
setAttributeNode：建立一个节点  
removeAttributeNode：删除一个节点

39.JavaScript 添加事件监听

addEventListener() 方法用于向指定元素添加事件句柄。

addEventListener() 方法添加的事件句柄不会覆盖已存在的事件句柄。

你可以向一个元素添加多个事件句柄。

你可以向同个元素添加多个同类型的事件句柄，如：两个 "click" 事件。

你可以向任何 DOM 对象添加事件监听，不仅仅是 HTML 元素。如： window 对象。

addEventListener() 方法可以更简单的控制事件（冒泡与捕获）。

使用 removeEventListener() 方法来移除事件的监听。

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还是捕获。该参数是可选的。

事件传递有两种方式：冒泡与捕获。

所有 HTML DOM 事件，可以查看我们完整的[HTML DOM Event 对象参考手册](http://www.runoob.com/jsref/dom-obj-event.html)。

40.创建JavaScript DOM元素结点

createElement(“a ”) 创建元素

createTextNode(“ b”) 创建文本内容

a.appendChild(b) 将b添加到a中

removeChild() 删除已有html元素 需要清楚您需要删除的元素，以及它的父元素

[HTML DOM 参考手册](http://www.runoob.com/jsref/jsref-tutorial.html)

41.JavaScript 高级教程

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...

无穷大（infinity） infinity正无穷大 -infinity 负无穷大

除以0可以产生无穷大 无穷大是一个数值

NaN 非数值

数字可以是数字或对象 ：var y = new Number(123);

42.JavaScript 算数(Math) 对象

Math.round(num) 取最接近的正整

Math.random() 产生0～1的随机数

Math.min / max () 返回给定的两个数的最大／最小值

[JavaScript Math 对象的参考手册](http://www.runoob.com/jsref/jsref-obj-math.html)

43.Window尺寸

有三种方法能够确定浏览器窗口的尺寸。

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

* window.innerHeight - 浏览器窗口的内部高度(包括滚动条)
* window.innerWidth - 浏览器窗口的内部宽度(包括滚动条)

对于 Internet Explorer 8、7、6、5：

* document.documentElement.clientHeight
* document.documentElement.clientWidth

或者

* document.body.clientHeight
* document.body.clientWidth

其他Window方法：

* window.open() - 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属性：

screen.availWidth 屏幕可用宽度

screen.availHeight 屏幕可用高度

screen.width／height 屏幕总宽度或总高度

screen.colorDepth 屏幕色彩深度

screen.pixelDepth 屏幕色彩分辨率

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。

**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.href 返回当前页面的URL

location.assign() 加载新的文档

Window Histroy

window.history 对象包含浏览器的历史。

**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。

* 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

Window Navigator

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

1.浏览器代号: " + navigator.appCodeName

2. 浏览器名称: " + navigator.appName

3. 浏览器版本: " + navigator.appVersion

4. 启用Cookies: " + navigator.cookieEnabled

5. 硬件平台: " + navigator.platform

6. 用户代理: " + navigator.userAgent

7. 用户代理语言: " + navigator.systemLanguage

注意：

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

44.JavaScript 弹窗

警告框：alert()

确认框：confirm()

确认框通常用于验证是否接受用户操作。

当确认卡弹出时，用户可以点击 "确认" 或者 "取消" 来确定用户操作。

当你点击 "确认", 确认框返回 true， 如果点击 "取消", 确认框返回 false。

提示框：prompt()

提示框经常用于提示用户在进入页面前输入某个值。

当提示框出现后，用户需要输入某个值，然后点击确认或取消按钮才能继续操纵。

如果用户点击确认，那么返回值为输入的值。如果用户点击取消，那么返回值为 null。

45.JavaScript 计时事件

在 JavaScritp 中使用计时事件是很容易的，两个关键方法是:

a. setInterval() - 间隔指定的毫秒数不停地执行指定的代码。

1.setInterval() 和 setTimeout() 是 HTML DOM Window对象的两个方法。

2.clearInterval() 用于停止 setInterval() 方法执行的函数代码。创建计时方法时你必须使用全局变量

b. setTimeout() - 暂停指定的毫秒数后执行指定的代码

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() 用于清理setTimeout （）必须在setTimeout事件设定触发前

46.JavaScript Cookie

Cookie用于储存web页面用户的信息，Cookie以名／值对的形式存储

例：document.cookie="username=<内容>; expires=<失效时间>; path=< path 参数告诉浏览器 cookie 的路径>”;

读区Cookie： var x = document.cookie;

document.cookie 将以字符串的方式返回所有的 cookie，类型格式： cookie1=value; cookie2=value; cookie3=value;

修改Cookie类似于创建Cookie

删除Cookie：删除 cookie 非常简单。您只需要设置 expires 参数为以前的时间即可

如果您设置了新的 cookie，旧的 cookie 不会被覆盖。 新 cookie 将添加document.cookie 中，所以如果您重新读取document.cookie，您将获得如下所示的数据：cookie1=value; cookie2=value;

47.JavaScript 表格的基本操作

//单个特殊单元格

function myFunction(){

var x = document.getElementById("myTable").rows[0].cells;

x[1].innerHTML = "修改后";

}

//删除操作

function rowDelete(r){

var i = r.parentNode.parentNode.rowIndex;

document.getElementById("myTable").deleteRow(i); //console.log(r.parentNode.parentNode.childNodes[1]); 同类型单元格获

}

//插入行

function insRow(){

var x = document.getElementById("myTable").insertRow(0);

x.insertCell(0).innerHTML = "新单元格1";

x.insertCell(1).innerHTML = "新单元格2";

}

//合并单元格

function setColspan(){

var x = document.getElementById("myTable").rows[0].cells;

x[0].colSpan ="2";

x[1].colSpan ="1";

}