# 1.1简介

**01. JavaScript是脚本语言**

JavaScript 是一种轻量级的编程语言。

JavaScript 是可插入 HTML 页面的编程代码。

JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。



**02.通过JavaScript你可以**

直接写入HTML输出流。

对事件做出反应。

改变HTML内容。

改变HTML样式。

改变HTML图片。

验证输入。

# 1.2 JavaScript用法

（1）脚本可以放在HTML的<head><body>中。如需在HTML页面中插入脚本，请使用<script>标签，浏览器会解释并执行位于 <script> </script>之间的 JavaScript 代码。

**注：**那些老旧的实例可能会在 <script> 标签中使用 type="text/javascript"。现在已经不必这样做了。JavaScript 是所有现代浏览器以及 HTML5 中的默认脚本语言。

（2） 脚本可以放在外部JavaScript文件中，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，如：<script src="myScript.js"></script>

**注：**

①在标签中填写 onclick 事件调用函数时，不是 onclick=函数名， 而是 onclick=函数名+()，代码如下：<button onclick="myfunction()" type="button">点击这里</button>

②外部 javascript 文件不使用 <script> 标签，直接写 javascript 代码。

③HTML 输出流中使用 document.write，相当于添加在原有html代码中添加一串html代码。而如果在文档加载后使用（如使用函数），会覆盖整个文档。

# 1.3 JavaScript输出

JavaScript 没有任何打印或者输出的函数。但可以通过不同的方式来输出数据：

①使用 window.alert() 弹出警告框。

②使用 document.write() 方法将内容写到 HTML 文档中。

③使用 innerHTML 写入到 HTML 元素。

④使用 console.log() 写入到浏览器的控制台。

你需要知道的细节：

**（1）window.alert 的补充:**

window.alert(5+6) 与 window.alert("5+6") 输出的值是不一样的。 window.alert(5+6) 会输出 11，而window.alert("5+6") 会输出 5+6。

这是因为当用引号时会认为引号中是字符串，从而直接将引号中的内容打印出来。

**（2）document.getElementByld() 和 document.write() 的区别**

document.write() 方法可以用在两个方面：

①页面载入过程中用实时脚本创建页面内容，用来追加一些标签。

②清除当前页面内容(包括源文档的任何变量或值)。重新生成内容。

它的第②点和 document.getElementByld() 相区别。前者在清除全部页面内容，然后生成新的内容，即“覆盖文档"。而后者仅仅是替换了元素里的内容。

**（3）关于document.write()你需要注意：**

<button onclick="winTest()">按钮</button>

function winTest()

{

    var txt1 = "This is a new window.";

    var txt2 = "This is a test.";

    document.open("text/html","replace");//加上

    document.writeln(txt1);

    document.write(txt2);

    document.close();//加上

}

该方法将关闭 open() 方法打开的文档流，并强制地显示出所有缓存的输出内容。如果您使用 write() 方法动态地输出一个文档，必须记住当你这么做的时候要调用 close() 方法，以确保所有文档内容都能显示。document.write() 不会隐式调用 document.close() 方法的，否则将不会有 This is a new window. 内容了

一旦调用了close()，就不应该再次调用 write()，因为这会隐式地调用 open() 来擦除当前文档并开始一个新的文档。

在载入页面后，浏览器输出流自动关闭。在此之后，比如延迟脚本 [setTimeout()] 或是 onload 执行的方法，任何一个对当前页面进行操作的 document.write()方法将打开—个新的输出流，它将清除当前页面内容(包括源文档的任何变量或值)。

**（4）console.log的用处**

主要是方便你调式javascript用的, 你可以看到你在页面中输出的内容。

相比alert他的优点是：

他能看到结构化的东西，如果是alert，弹出一个对象就是[object object],但是console能看到对象的内容。

console不会打断你页面的操作，如果用alert弹出来内容，那么页面就死了，但是console输出内容后你页面还可以正常操作。

console里面的内容非常丰富，你可以在控制台输入 console，然后就可看到：

此外，它还可以添加显示样式：

console.log('%cfuck', 'font-size:20px')

1.4 JavaScript语法

（1）常量：const定义，且定义之后不可再更改；

（2）字面量：字面量就是没有用标识符封装起来的量，是“值”的原始状态。数字、表达式、字符串、数组、对象、函数。

（3）变量：javaScript是弱类型编程语言，都用var来声明变量。

变量可以以字母、$和\_符号开头（后两者不推荐）；

变量名称对大小写敏感（y和Y是不同的变量）

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当向变量分配文本值时，应该用双引号或单引号包围这个值。

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

（4）语句：JavaScript 语句可以写在函数内，函数可以重复引用。

引用一个函数 = 调用函数(执行函数内的语句)。

① ; 分号用于分隔 JavaScript 语句；使用分号的另一用处是在一行中编写多条语句。

② {} 代码块的作用是一并地执行语句序列。

③ 语句标示符 JavaScript 语句通常以一个语句标识符为开始，并执行该语句。

语句标识符是保留关键字不能作为变量名使用。

如：if else; for...in; for; do...while; while; break; continue;

switch case; try...catch; return; throw; function; var;

④ 空格JavaScript 会忽略多余的空格。您可以向脚本添加空格，来提高其可读性。

（5）JavaScript对字母大小写敏感

函数getVar() 与getvar()是不同的；变量name和Name也是不同的。

（6）命名规则

①匈牙利命名方式：变量名+对象描述

整数i 浮点fl 布尔b 字符串s 数组a 对象o 函数fn 正则re

例：iAge = 18;

②驼峰命名：

全部小写：单词与单词间全部用下划线分割

大小写混合：

每个单词的第一个字母大写,这种叫“大驼峰”；

第一个单词字母小写，其他单词第一个字母大写，叫做“小驼峰”。

（7）JavaScript字符集

使用Unicode字符集，Unicode覆盖了所有的字符，包含标点等字符。

（8）注释

页面注释<!--注释内容-->

JavaScript注释：

单行注释：//

多行注释：/\*注释内容\*/

注释的作用：

①方便二次阅读，有助于代码维护；

②方便后续开发者理解维护你的代码；

③使用注释符号验证浏览器是否支持JavaScript脚本功能，如下：

<script>

<!--

document.write("您的浏览器支持JavaScript脚本!");

//这是JavaScript注释符号

-->

</script>

# 1.5 JavaScript变量

ES6之前版本JS只有函数作用域和全局作用域，没有块级作用域，所以{}限定不了var声明变量的访问范围。

函数作用域中声明的变量称为局部变量，会在函数运行以后被删除；

全局作用域中的变量称为全局变量，会在页面关闭后被删除。

⬩ 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。

⬩ JavaScript允许重复声明变量。有异议：若该变量已声明，就不会再被声明；

⬩ 如果把值赋给尚未声明的变量，该变量将被自动作为 window 的一个属性。非严格模式下给未声明变量赋值创建的全局变量，是全局对象的可配置属性，可以删除。

1. **var** carname="Volvo";
2. **var** carname;    //该变量值依然是Volvo

⬩ JavaScript没有重载的概念，允许重复声明函数，后声明的覆盖之前的。

⬩ 变量与函数重名的时候，变量生效

这涉及到了变量和函数的预解析：

⬩ 变量声明会被顶置，函数声明也会被顶置且比变量更先声明。

⬩ 变量的声明和赋值语句一起写时，JS引擎在解析时，会将其拆成声明和赋值2部分，声明置顶，赋值保留在原来位置。

⬩ 声明过的变量不会再重复声明。

在ES6版本新增let，用于定义块级作用域变量。

⬩ let没有变量提升，即存在暂时性死区；

注：ES6 明确规定，如果区块中存在let和const命令，这个区块对这些命令声明的变量，从一开始就形成了封闭作用域。凡是在声明之前就使用这些变量，就会报错。在代码块内，使用let命令声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ）。

⬩ let变量不能重复声明

# 1.6 JavaScript数据类型

**值类型(基本类型)：**字符串（String）、数字(Number)、布尔(Boolean)、对空（Null）、未定义（Undefined）、Symbol。

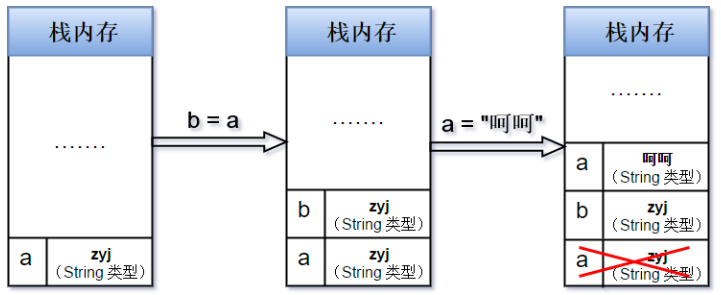
**引用数据类型：**对象(Object)、数组(Array)、函数(Function)。

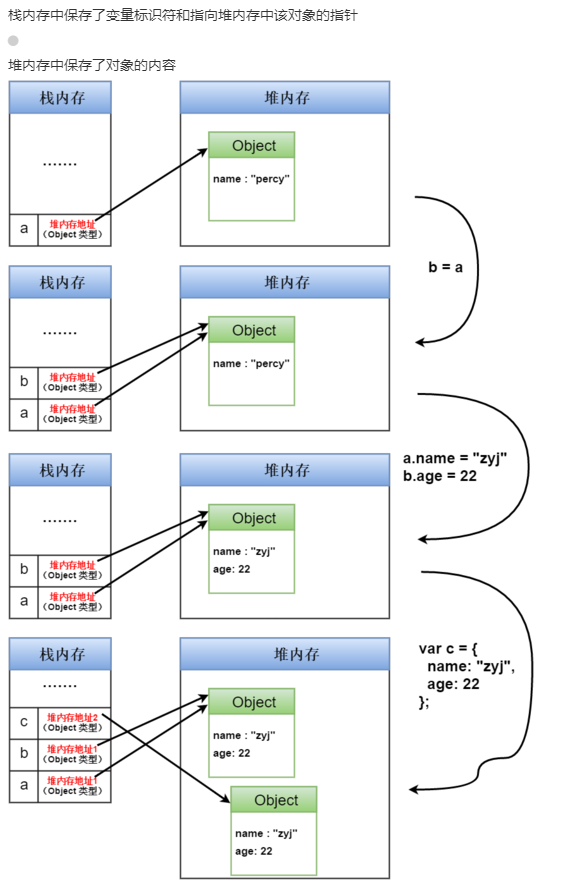
注：Symbol 是 ES6 引入了一种新的原始数据类型，表示独一无二的值。

⬩ 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。

⬩ 基本类型的变量是存放在栈内存（Stack）里的，栈内存中包括了变量的标识符和变量的值。

⬩ 栈内存中保存了变量标识符和指向堆内存中该对象的指针，堆内存中保存了对象的内容





# 1.7 JavaScript对象

JavaScript 对象是拥有属性和方法的数据。

1. <script>
2. **var** person = {
3. firstName: "John",
4. lastName : "Doe",
5. id : 5566,
6. fullName : **function**()
7. {
8. **return** **this**.firstName + " " + **this**.lastName;
9. }
10. };
11. document.getElementById("demo").innerHTML = person.fullName();
12. </script>

# 1.8 JavaScript函数

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

关键词 function 必须是小写的，并且必须以与函数名称相同的大小写来调用函数。

变量和参数必须以一致的顺序出现。第一个变量就是第一个被传递的参数的给定的值，以此类推。

如果实参个数比形参少，那么剩下的默认赋值为 undefined；

如果实参传的比形参数量多，那么是全部都会被传进去的，只不过没有对应的形参可以引用（但可以用 arguments 来获取剩下的参数）。

1. <script>
2. **function** test(arg1,arg2,arg3,arg4) {
3. console.log("agr4:"+arg4);      //undefined
4. **for**(**var** i=0; i<arguments.length; i++) {
5. console.log(arguments[i]);
6. }
7. }
8. test(1,2,3); //输出 1 2
9. </script>

# 1.9 JavaScript事件

HTML事件是发生在HTML元素上的事情。在HTML页面中使用JavaScript时，可以触发这些事件。HTML事件可以是浏览器行为，也可以是用户行为。HTML的实例：

HTML页面完成加载；

HTML input字段改变时；

HTML按钮被点击；

通常，当事件发生时，你可以做些事情。



事件可以用于处理表单验证，用户输入，用户行为及浏览器操作：

页面加载时触发事件；

页面关闭时触发事件；

用户点击按钮执行动作；

验证用户输入内容的合法性...

HTML事件属性可以直接执行JavaScript代码，也可以调用JavaScript函数。你可以为HTML元素指定自己的事件处理程序。

# 1.10 JavaScript字符串

字符可以是插入到引号（单引号或双引号）中的任何字符。

①可以使用索引来访问字符串中的每个字符，索引从0开始。

②字符串中的引号不能与字符串的引号相同，可以通过添加转义 (\)字符来使用引号。



注：" "中用' ' 和 ' '中用" "可以不加反斜杠。

③字符串可以是对象。

通常，JavaScript字符串是原始值，可以使用字符创建var x = "John"，typeof x 返回String；但也可以使用new关键字将字符串定义为一个对象var y = new String("John")，typeof y返回Object。注：尽量不要创建String对象，它会拖慢执行速度，并可能产生其他副作用。

**字符串属性和方法：**

原始值字符串，如 "John", 没有属性和方法(因为他们不是对象)。

原始值可以使用 JavaScript 的属性和方法，因为 JavaScript 在执行方法和属性时可以把原始值当作对象。

**String对象属性**

constructor 对创建该对象的函数的引用

length 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**

charAt(index) 返回在指定位置的字符。

charCodeAt(index) 返回在指定的位置的字符的 Unicode 编码。

concat(str1,str2,...,strX) 连接两个或更多字符串，并返回新的字符串。

fromCharCode(n1,n2,...,nX) 将 Unicode 编码转为字符。

indexOf(searchValue[,start]) 返回某个指定的字符串值在字符串中首次出现的位置。

includes(searchValue[,start]) 查找字符串中是否包含指定的子字符串,start默认为 0。

lastIndexOf(searchValue[,start]) 从后向前搜索字符串，并从起始位置(0)开始计算返回字符串最后出现的位置。

match(regexp) 查找找到一个或多个正则表达式的匹配。

repeat(count) 复制字符串指定次数，并将它们连接在一起返回。

replace(searchValue[,newValue]) 在字符串中查找匹配的子串， 并替换与正则表达式匹配的子串。

search(searchvalue) 查找与正则表达式相匹配的值。

slice(start,end) 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(separator[,limit]) 把字符串分割为字符串数组,limit指定返回的数组的最大长度。

startsWith(searchvalue[, start]) 查看字符串是否以指定的子字符串开头。

substr(start[,length]) 从起始索引号提取字符串中指定数目的字符。

substring(from[,to]) 提取字符串中两个指定的索引号之间的字符。

toLowerCase() 把字符串转换为小写。

toUpperCase() 把字符串转换为大写。

trim() 去除字符串两边的空白

toLocaleLowerCase() 根据本地主机的语言环境把字符串转换为小写。

toLocaleUpperCase() 根据本地主机的语言环境把字符串转换为大写。

valueOf() 返回某个字符串对象的原始值。

toString() 返回一个字符串。

**String HTML包装方法**

anchor(name) 创建 HTML 锚。返回加了 <a> 标签的字符串

big() 用大号字体显示字符串。返回加了 <big> 标签的字符串

blink() 显示闪动字符串。。。。

bold() 使用粗体显示字符串。

fixed() 以打字机文本显示字符串。

fontcolor() 使用指定的颜色来显示字符串。

fontsize() 使用指定的尺寸来显示字符串。

italics() 使用斜体显示字符串。

link() 将字符串显示为链接。

small() 使用小字号来显示字符串。

strike() 用于显示加删除线的字符串。

sub() 把字符串显示为下标。

sup() 把字符串显示为上标。

# 1.11 JavaScript运算符

1. 算术运算符：+ - \* / % ++ --

2. 赋值运算符：= += -= \*= /= %=

3. 用于字符串的 + 运算符

4. 对于字符串和数字进行加法运算

注意：

① 字符串一个很能强大的数据类型；在执行加 + 时,将被加的对象统一处理为字符串。空格会计入长度。

② bool 类型在与数字类型进行相加时，视为 0 或者 1 处理。

③ null 类型与数字类型进行累加时，视为 0 处理。

④ bool 类型与 null 类型进行累加时，视为其与整数类型累加处理。

⑤ undefined 除了与字符串进行累加时有效（undefined 视为字符串"undefined"处理），其他情况皆返回 NaN。

6、求模 % 运算，运算结果只取决于第一个数字的正负。

# 1.12 JavaScript比较和逻辑运算符

1.比较运算符

== 等于 != 不等于

=== 绝对等于（比较的是值和类型） !== 不绝对等于

> >= < <=

2.逻辑运算符

&& || !

3.条件运算符

var out=condition?value1:value2;

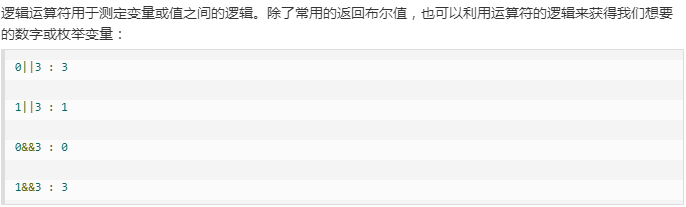
注：==与===比较

对于String、number等基础类型，==和==是有区别的：

（1）如果值类型不同，===结果就是false；==只比较转化为同一类型后的值是否相等。

（2）如果类型相同，===与==的比较结果一致。

对于Array、Object高级类型，==和===是没有区别的，进行指针地址比较。



理解js继承的6中方式：

<https://www.cnblogs.com/Grace-zyy/p/8206002.html>

JS高级进阶

事件的绑定、捕获及冒泡、执行顺序

<https://www.jb51.net/article/94394.htm>

<https://blog.csdn.net/lianzhang861/article/details/80776379>

JavaScript的事件执行机制---进阶必备 <https://www.jianshu.com/p/3f75e7de05ea>