

-

整理自互联网，仅供学习交流。



2018-10-30

[公司名称]

[公司地址]

目录

123

# JS简介

JavaScript简介:

JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为JavaScript引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在HTML（标准通用标记语言下的一个应用）网页上使用，用来给HTML网页增加动态功能。Node.js。

在1995年时，由Netscape公司的Brendan Eich，在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管理层希望它外观看起来像Java，因此取名为JavaScript。但实际上它的语法风格与Self及Scheme较为接近。

为了取得技术优势，微软推出了JScript，CEnvi推出ScriptEase，与JavaScript同样可在浏览器上运行。为了统一规格，因为JavaScript兼容于ECMA标准，因此也称为ECMAScript。

组成部分:

JavaScript 的内容，包含以下三部分：

ECMAScript（核心）：JavaScript 语言基础(规定了 JavaScript 脚本的核心语法，如 数据类型、关键字、保留字、运算符、对象和语句等，它不属于任何浏览器。)；

DOM（文档对象模型）：规定了访问HTML和XML的接口(提供了访问 HTML 文档（如body、form、div、textarea等）的途径以及操作方法)；

BOM（浏览器对象模型）：提供了独立于内容在浏览器窗口之间进行交互的对象和方法(提供了访问某些功能（如浏览器窗口大小、版本信息、浏览历史记录等）的途径以及操作方法)。

## 基本特点:

是一种解释性脚本语言（代码不进行预编译）。

主要用来向HTML（标准通用标记语言下的一个应用）页面添加交互行为。

可以直接嵌入HTML页面，但写成单独的js文件有利于结构和行为的分离。

## 日常用途:

嵌入动态文本于HTML页面。

对浏览器事件做出响应。

读写HTML元素。

在数据被提交到服务器之前验证数据。

检测访客的浏览器信息。

控制cookies，包括创建和修改等。

## 作用：

交互式操作

表单验证

网页特效

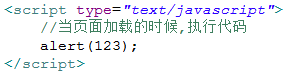
web游戏

服务器脚本开发

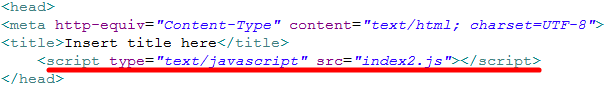
## js代码存在的位置

**JavaScript存在的位置**:

**方式1:** 在<html>标签中,任何地方添加<script></script>标签.标签中内容就是js代码. 虽然可以放在页面的任何地方,但是规范放在<head>标签中.

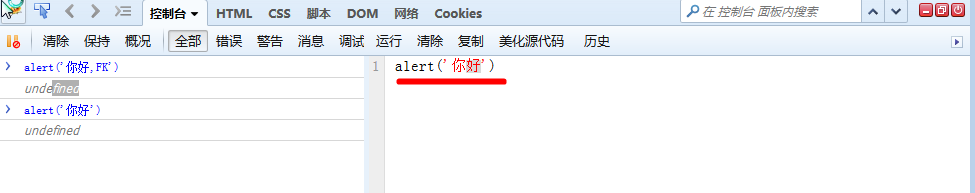


**方式2**:单独使用一个文件来编写javascript代码(**js文件**),在需要使用的页面中引入该文件.



**方式3**:把代码编写的**a标签**的href属性中,点击a标签的时候,就执行里面代码.





## 基础语法

基本语法(大多数和Java一样):

JavaScript中的标识符:

变量，常量，函数，语句块也有名字，我们统统称之为标识符。标识符可以由任意顺序的大小写字母、数字、下划线(\_)和美元符号($)组成，标识符不能以数字开头，不能是JavaScript中的保留字或关键字(和Java一样)。

合法的标识符举例：indentifier、username、user\_name、\_userName、$username

非法的标识符举例：int、98.3、Hello World

JavaScript严格区分大小写.

username和userName是两个完全不同的符号

JavaScript程序代码的格式

每条功能执行语句的最后必须用分号(;)结束，每个词之间用空格、制表符、换行符或大括号、小括号这样的分隔符隔开 。

语句块使用{}来表示

JavaScript程序的注释

/\*…\*/中可以嵌套“//”注释，但不能嵌套“ /\*…\*/”。 、/\*\*..文档注释.\*/

### 定义变量:

什么是变量，系统为之设置一个标识，程序可以用变量名来指向具体的对象内存，并通过变量名来获得对应的对象值。

**声明变量要使用var关键字**

例如：**var name;**//一般不使用name作为变量名

声明变量的同时为其赋值

例如：var name = "will";

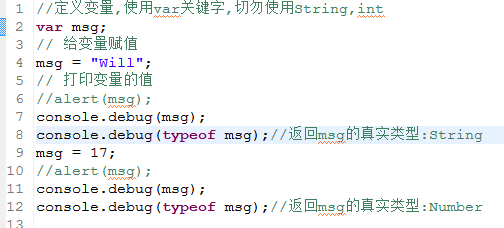
对已赋值的变量赋予一个其他类型的数据

例如：name = 17;

不事先声明变量而直接使用,报错 not defined.(未定义)

例如：x = 1234;

　提示：javascript定义变量无需指定类型，任何类型都使用var声明,感觉var就有点类型于java中的Object类型了

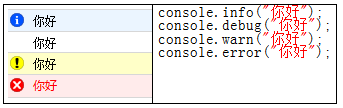


### 变量的打印:

**方式1:使用alert(内容)**;

**方式2:使用console.debug(内容)**; 注意:只有W3C的浏览器才支持(IE不行),Firebug中,Google的控制台.

好比System.out.println(内容);



## JS的基本类型和常量:

Number(数字类型)

1、整型常量(10进制\8进制\16进制)

十六进制以0x或0X开头，例如：0x8a。

八进制必须以0开头，例如：0123。

十进制的第一位不能是0（数字0除外），例如：123。

２、实型常量

12.32、193.98、 5E7、4e5等。

特殊数值：NaN(Not a Number)、Infinity(除数为零)，所对应的判断函数isNaN()、isFinite()

**Boolean(布尔值) true和false**。

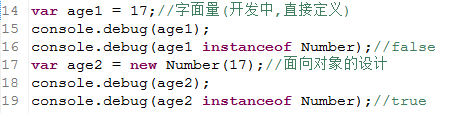
String字符串(注意js中没有char类型，所有'a'也是一个字符串)

“a book of JavaScript”、‘a’、 “a”、“”。

字符串中的特殊字符，需要以反斜杠（\）后跟一个普通字符来表示。

**null常量:表示引用任何的内存空间.**

**undefined常量 (未定义,定义未赋值)**



## 解决表单重复提交

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <title>Form表单</title>  <script type="text/javascript">  var isFlag = false; //表单是否已经提交标识，默认为false  function submitFlag() {  if (isFlag == false) {  isFlag = true;  return true;  } else {  return false;  }  }  </script>  </head>  <body>  <form action="${pageContext.request.contextPath}/DoFormServlet"  method="post" onsubmit="return submitFlag()">  用户名：<input type="text" name="userName"> <input type="submit"  value="提交" id="submit">  </form>  </body>  </html> |

除了用这种方式之外，经常见的另一种方式就是表单提交之后，将提交按钮设置为不可用，让用户没有机会点击第二次提交按钮，代码如下：

|  |
| --- |
| function dosubmit(){  //获取表单提交按钮  var btnSubmit = document.getElementById("submit");  //将表单提交按钮设置为不可用，这样就可以避免用户再次点击提交按钮  btnSubmit.disabled= "disabled";  //返回true让表单可以正常提交  return true;  } |

## JS中的运算符和Java中大多数相同:

算术运算符

赋值运算符

比较运算符

逻辑运算符

三元运算符

位运算符

------------------------------------------------

**JS中特殊存在的案例**:

**第一个不同**:

在JS中,比较相等的运算符: **==,===**.

问题:=和==以及===的区别:

= : 和Java相同,用于给变量赋值.

== : 用来比较变量或者常量的内容,不管数据类型.

===: 比较数据类型和内容,若数据类型相同,且内容相同返回true.

console.debug(17 == "17");//true

console.debug(17 === "17");//false

**第二个不同**:

在逻辑运算中，**0、""、false、null、undefined、NaN均表示false**。

----------------------------------------------------------------------------------

a && b : 将a, b转换为Boolean类型, 再执行逻辑与, 若表达式结果为true返回b, false返回a

a || b : 将a, b转换为Boolean类型, 再执行逻辑或, 若表达式结果为true返回a, false返回b

---------------------------------------------------------------------------------

&& 和|| 运算符的区别(开关)：

**&&操作:返回最后一个为true的值,或者第一个为false的值.**

**||操作:返回第一个为true的值,或则最后一个为false的值.**

console.debug(true && true)

console.debug(1 && true)

console.debug(1 && 2)

console.debug("A" && 2)

console.debug("" && 2)

console.debug(null && "B")

console.debug("A" && "B")

console.debug(1 && 2 && 3)

console.debug(1 && null && 3)

console.debug("" && null && 0)

if(123){

alert('OK');

}

if(0){

alert('OK');

}

## 函数(function)

### 函数的作用

1):将脚本编写为函数，就可以避免页面载入时执行该脚本。

2):函数包含着一些代码，这些代码可以多次被调用。

### 函数的定义和调用:

定义函数分类:普通函数/匿名函数.

函数必须调用才会执行.

定义语法:

function *函数名*(参数列表)

{

函数体

[return 值/变量]

}

### 函数的返回:

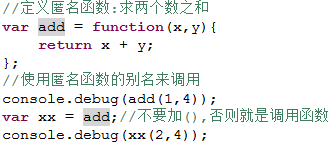
如果在函数中使用了return,那么函数的返回结果就是 return 后的**值**.

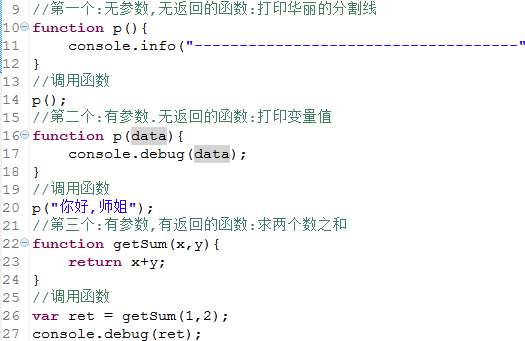
如果在函数中没有使用return,说明该函数没有结果,不用接受,若接受,结果为:undefined.

### 匿名函数(经常使用):

在定义函数的时候,没有定义函数名称.

此时,可以把函数当做是一个**值/变量,可以赋给另一个变量**.





## 全局变量和局部变量

**变量根据在function的内外定义位置不同,分成**:

**全局变量:function外部定义的变量称为全局变量**

**局部变量:function内部定义的变量称为局部变量**

访问变量原则：就近原则，谁离我最近我就使用谁。

-----------------------------------------------------

<script language="javascript">

var msg = "全局变量";

function show()

{

msg = "局部变量";

}

show();

alert(msg);//局部变量

</script>

-----------------------------------------------------

<script language="javascript">

var msg = "全局变量";

function show()

{

**var msg**;

msg = "局部变量";

}

show();

alert(msg);//全局变量

</script>

## 系统函数

**参见与:W3C javascript.chm ---> JavaScript 全局对象参考手册**.

在JS中已经预先定义好的,可以直接拿来使用的函数.

-------------------------------------------------------------

encodeURI及encodeURIComponent方法。

返回对一个URI字符串编码后的结果。

decodeURI及decodeURIComponent()方法。

将一个已编码的URI字符串解码成最初始的字符串并返回。

parseInt方法

将一个字符串按指定的进制转换成一个整数，语法格式为：parseInt(numString, [radix])。如果没有指定第二个参数，则前缀为 ‘0x’ 的字符串被视为十六进制，前缀为 ‘0’ 的字符串被视为八进制，所有其他字符串都被视为是十进制。

parseFloat方法

将一个字符串转换成对应的小数。

isNaN方法

检查一个值是否为NaN。

escape方法(不推荐使用,改为encodeURIComponent)

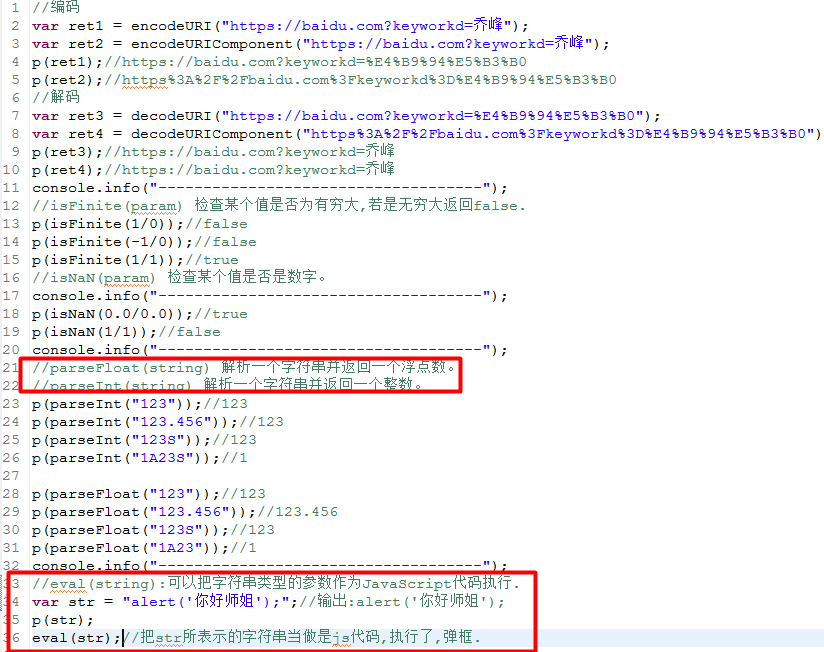
返回对一个字符串进行编码后的结果字符串。所有空格、标点、重音符号以及任何其他非 ASCII 字符都用 %xx 编码替换，其中xx等于表示该字符的Unicode编码的十六进制数，字符值大于255的字符以%uxxxx格式存储。

unescape 方法(不推荐使用，改为decodeURIComponent)

将一个用escape方法编码的结果字符串解码成原始字符串并返回。

eval 方法 \*

将参数字符串作为一个JavaScript表达式执行。



## 面向对象

对象中所包含的变量就是对象的属性，对象中所包含的对属性进行操作的函数就是对象的方法，对象的属性和方法都叫对象的成员。

类是对某一类事物的描述，是抽象上的概念；而对象实例是一类事物中的具体个例。

能够被用来创建对象实例的函数就叫对象的构造函数。使用new关键字和对象的构造函数就可以创建对象实例，

语法格式如下：var objInstance = new ObjName([传递给该对象的实际参数列表]);

----------------------------------------------

函数:面向过程的,可以独立存在和运行.

方法:面向对象的,函数在面向对象中的称谓,此时不能独立存在,因为要依附于对象,也使用对象来调用.

构造函数:专门用来构建对象的.

类:就是一个函数,是一个特殊的函数,构造函数.

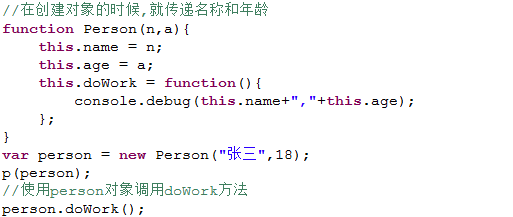
**js创建一个类,只需要定义类的构造函数(方法)**:



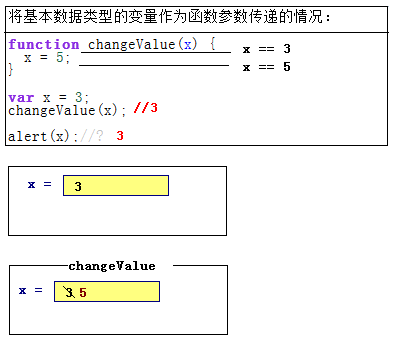
**在JS中this表示什么**? this就表示当前对象

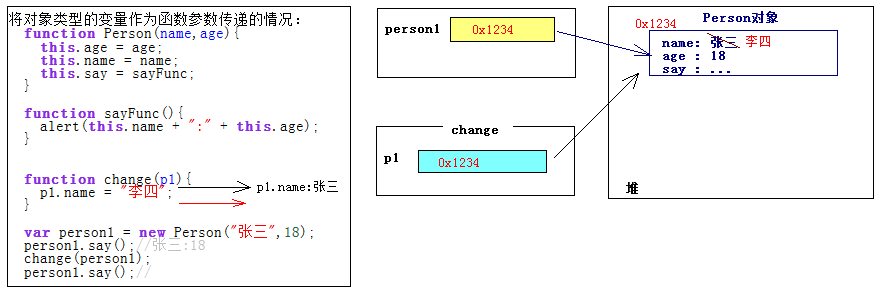
1):this在构造器(类)中,此时this就表示当前被创建的对象.

2):this在方法中,此时哪一个对象调用this所在的方法,this就是哪一个对象.



## 函数参数的值传递方式





## 内置对象

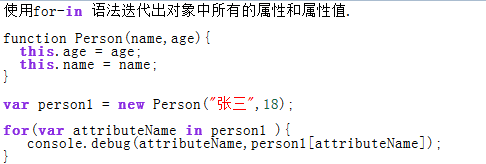
**for in循环好比是Java中的foreach循环**:

for in循环作用:

1.遍历数组里所有的数组元素的索引.

2.遍历js对象的所有属性.

--------------------------------------------------



**参见与:W3C javascript.chm ---> ECMAScript 引用类型**.

Object:

创建对象,并设置属性和方法

var obj = new Object();

obj.name = "will";

obj.age = 17;

obj.sayHello = function() {

};

// 对象的构造函数

alert(obj.constructor);

// 是否有指定的属性

debug(obj.hasOwnProperty("name1"));

// 迭代对象所有的属性+方法:for...in

for (attr in obj) {

alert(attr)//属性名

}

---------------------------------------------------------

Array(单独讲)

Boolean

----------------------------------------------------------

Date:

打印当前系统时间:

var d = new Date();

var day = d.getDate();

day = day<10?"0"+day:day;

var time = d.getFullYear() + "-" + (d.getMonth()+1) + "-" + day + " "

+ d.getHours() + ":" + d.getMinutes() + ":" + d.getSeconds();

----------------------------------------------------------

Math:

var num = Math.random();

----------------------------------------------------------

Number

----------------------------------------------------------

String:

// 随机生成A到Z之间的字母:65+26

// 随机生成0~25

var num = parseInt(Math.random() \* 26);//

num = num + 65;

alert(String.fromCharCode(num));

-----------------------------------------------------------

RegExp

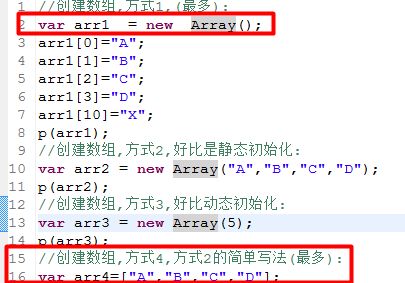


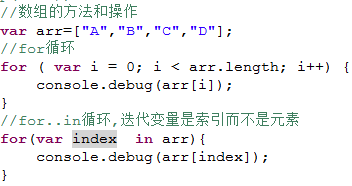
## 数组

### 为什么使用数组:存储多个数据.

js中的数组和Java中的ArrayList有异曲同工之妙,爽.

### 如何创建数组对象:





**操作数组的属性和方法**:

length－获得数组的长度；

concat－连接数组；

join－把数组转换成字符串；

pop－删除最后一个元素；

push－放入一个元素；

reverse－颠倒数据中的元素顺序；

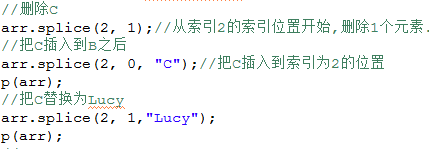
shift－移出第一个元素；

splice－截取数组;

sort－排序数组;

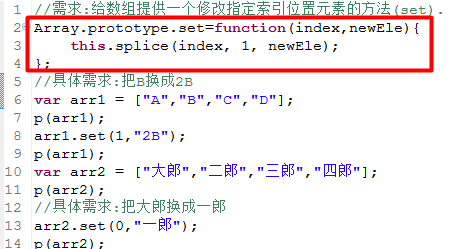
unshift－在前面追加元素；

**splice(start, deletecount, items) 方法用于插入、删除或替换数组的元素。**



## prototype

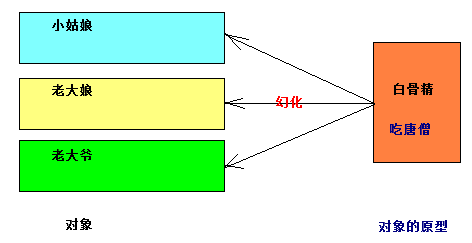
**需求:给数组提供一个修改指定索引位置元素的方法(set)**.



**使用prototype**:

**可以理解为在类上增加静态方法.**

**静态方法属于类,属于该类所有的对象,如此该方法就属于所有的对象.**



小结

