# day09-JavaScript基础

* 今日内容
* JavaScript介绍
* JavaScript与HTML的结合方式
* JavaScript的语法
* EMCAScript的常见对象

## 1. JavaScript的介绍

### 1.1 JavaScript\_简介

* 概念：一门客户端脚本语言

运行在客户端浏览器中的。每一个浏览器都有JavaScript的解析引擎

脚本语言：不需要编译，直接就可以被浏览器解析执行了

* 功能：  
  可以来增强用户和html页面的交互过程，可以来控制html元素，让页面有一些动态的效果，增强用户的体验。

### 1.2 JavaScript\_发展史

1. 1992年，Nombase公司，开发出第一门客户端脚本语言，专门用于表单的校验。

命名为 ： C-- ，后来更名为：ScriptEase

2. 1995年，Netscape(网景)公司，开发了一门客户端脚本语言：LiveScript。

后来请来SUN公司的专家，修改LiveScript，命名为JavaScript

3. 1996年，微软抄袭JavaScript开发出JScript语言

4. 1997年，ECMA(欧洲计算机制造商协会)，制定出客户端脚本语言的标准：

ECMAScript，就是统一了所有客户端脚本语言的编码方式。  
  
     JavaScript = ECMAScript + JavaScript自己特有的东西(BOM+DOM)

BOM : Browser Object Model

DOM : Document Object Model

## 2. JavaScript与html结合方式

### 2.1 JavaScript\_语法\_与html结合方式

* 内部JS：  
    定义<script>，标签体内容就是js代码



* 外部JS：  
    定义<script>，通过src属性引入外部的js文件



* 注意事项 :

1. <script>可以定义在html页面的任何地方。但是定义的位置会影响执行顺序。

2. <script>可以定义多个

## 3. JavaScript的语法

### 3.1 JavaScript\_语法\_注释&数据类型

#### 3.1.1 注释

A. 单行注释 : //

B. 多行注释 : /\* \*/

#### 3.1.2 数据类型

* 原始数据类型:

number : 整数/小数/NaN -> Nan : not a number 一个不是数字类型的数字(就不是个数)

string : 字符/字符串

boolean : true/false

null : 一个对象为空的占位符

undefined : 未定义 -> 如果一个变量没有给初始化值,则会被默认赋值为undefined

* 引用数据类型:

对象.

### 3.2 JavaScript\_语法\_变量

* 变量：一小块存储数据的内存空间
* Java与JS的语言区别

Java语言是强类型语言，而JavaScript是弱类型语言.

强类型：在开辟变量存储空间时，定义了空间将来存储的数据的数据类型。

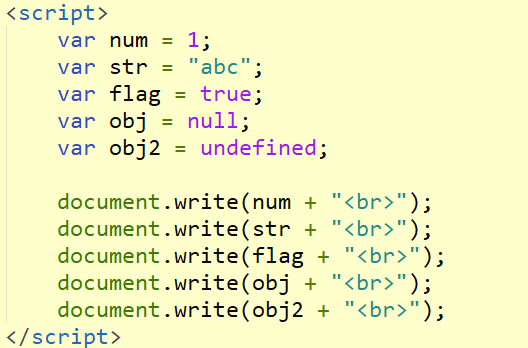
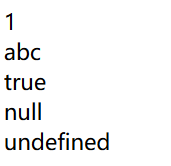
只能存储固定类型的数据

弱类型：在开辟变量存储空间时，不定义空间将来的存储数据类型

可以存放任意类型的数据

* 语法：

var 变量名 = 初始化值;

* document.write() : 将数据写出到页面的body当中, 没有自动换行.

注意 : js当中, 变量可以出现重名的问题, 现象是后者对前者进行覆盖.

var a = 10;

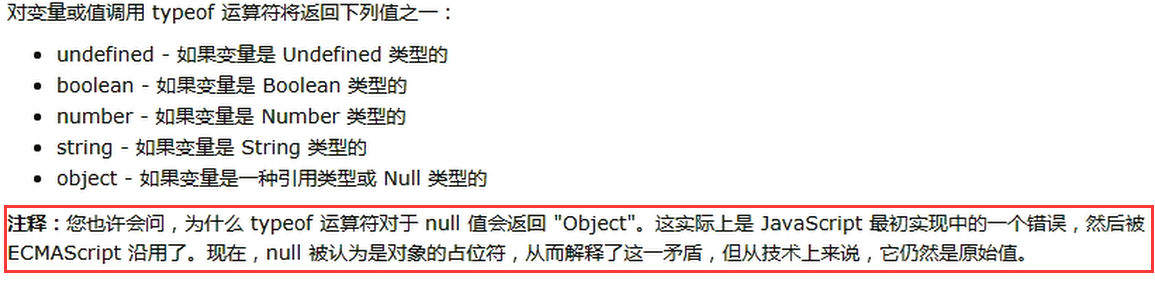
var a = 20;

alert(a); // 20

### 3.3 JavaScript\_语法\_变量\_typeof

* typeof运算符：获取变量的类型。

理解 : 照妖镜.



### 3.4 JavaScript\_语法\_一元运算符

* 一元运算符：只有一个运算数的运算符 (++, --, (正号 +))

++ --: 自增(自减)

\* ++(--) 在前，先自增(自减)，再运算

\* ++(--) 在后，先运算，再自增(自减)

* 注意：在JS中，如果运算数不是运算符所要求的类型，那么js引擎会自动的将运算数进行类型转换

其他类型转number：  
          string转number：按照字面值转换。如果字面值不是数字，则转为NaN（不是数字的数字）  
          boolean转number：true转为1，false转为0

### 3.5 JavaScript\_语法\_算数&赋值运算符&比较运算符

* 算数运算符 & 赋值运算符 
* 比较运算符

1. 类型相同：直接比较

字符串：按照字典顺序比较。按位逐一比较，直到得出大小为止。

var str1 = “abc”;

var str2 = “acd”;

document.write(str1 > str2 + “<br>”); false

2. 类型不同：先进行类型转换，再比较

===：全等于。在比较之前，先判断类型，如果类型不一样，则直接返回false

document.write(123 === “123”); (需要类型和数值完全一样)

### 3.6 JavaScript\_语法\_逻辑运算符

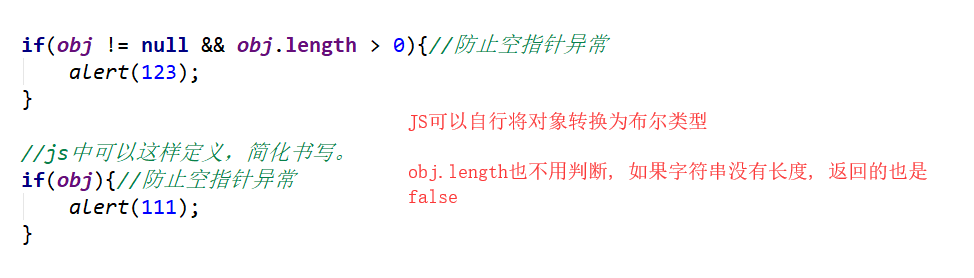
* && || !
* 其他类型转boolean：

1. number：0或NaN为假，其他为真

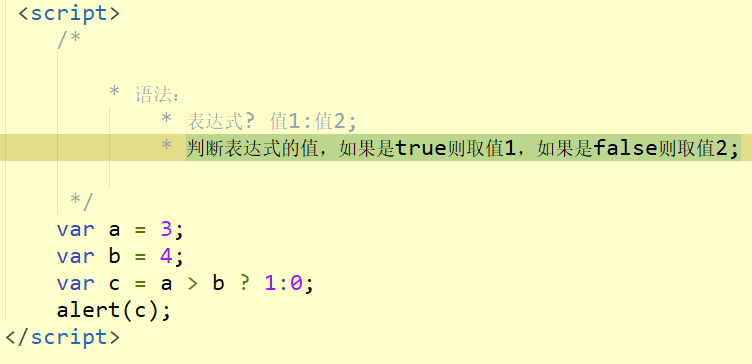
2. string：除了空字符串("") str.length = 0，其他都是true

3. null&undefined:都是false

4. 对象：所有对象都为true, 对象为null的时候, 才是false



### 3.7 JavaScript\_语法\_三元运算符



### 3.8 JavaScript\_语法\_特殊语法

1. 语句以;结尾，如果一行只有一条语句则 ;可以省略 (不建议)

2. 变量的定义使用var关键字，也可以不使用

\* 用： 定义的变量是局部变量

\* 不用：定义的变量是全局变量(不建议)

理解 : 可以将全局变量看做为是一个静态的成员变量.

虽然类似Java的静态变量, 会有单独的静态区域存储, 但并不会优先加载

### 3.9 JavaScript\_语法\_流程控制语句

1. if...else...

2. switch:

\* 在java中，switch语句可以接受的数据类型： byte int shor char,枚举(1.5) ,String(1.7)

\* switch(变量):

case 值:

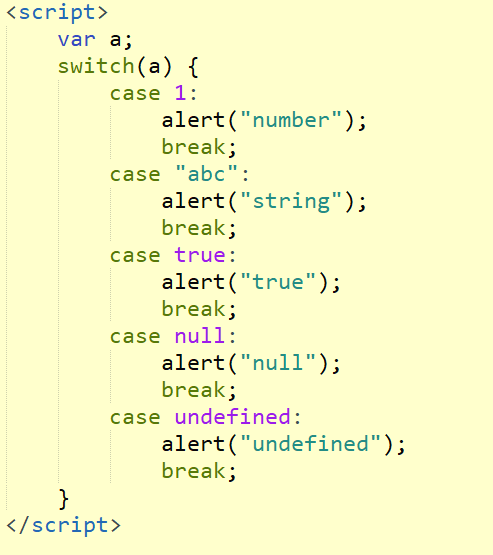
\* 在JS中,switch语句可以接受任意的原始数据类型

3. while

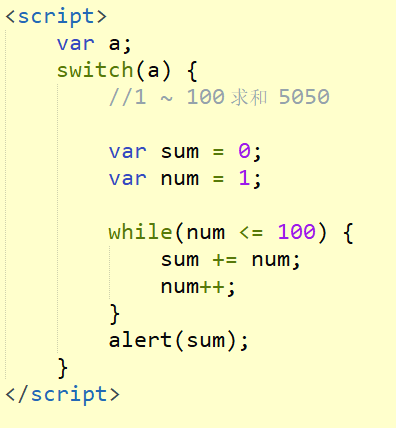
4. do...while

5. for

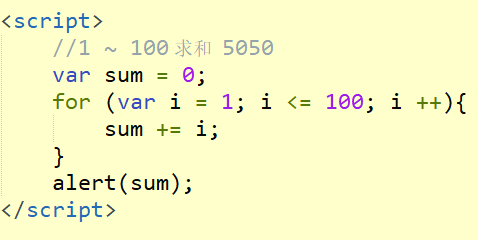
* switch示例代码 :



* while循环示例代码 :

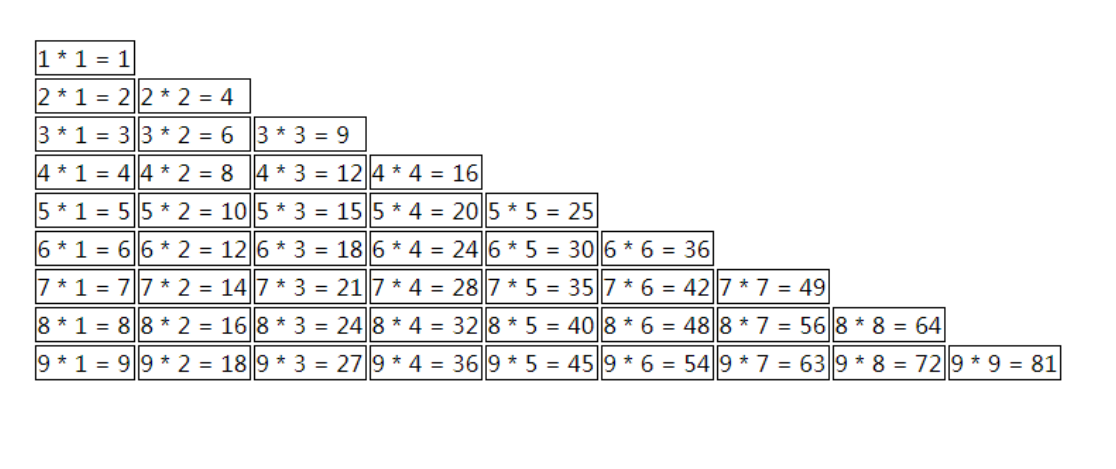


* for循环示例代码 :



### 3.10 JavaScript\_语法\_练习\_99乘法表

* 需求



* 示例代码 :

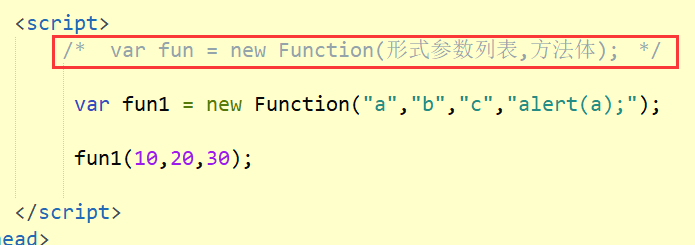


## 4. EMCAScript常见对象

### 4.1 JavaScript\_对象\_Function

#### 4.1.1 创建的三种格式

* 第一种 : 不用记

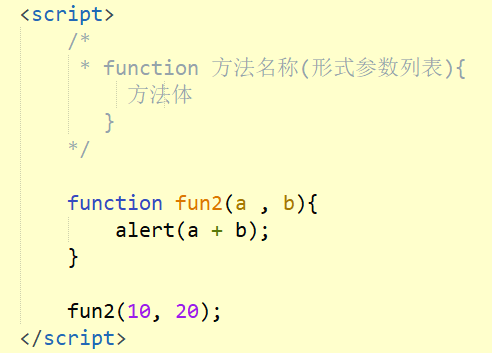


* 第二种 : 重点记
* 定义方法的思路

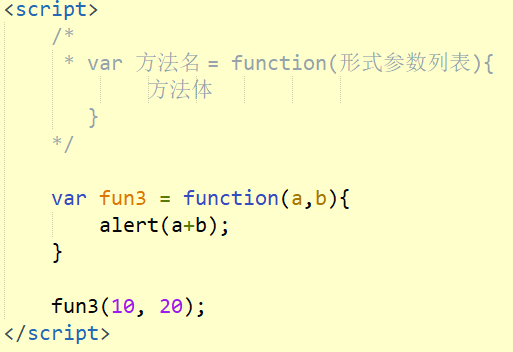
1. function

2. 起个名

3. 写出需要的参数.



* 第三种 : 重点记



#### 4.1.2 方法的属性

* length:代表形参的个数

#### 4.1.3 JS方法的特点

1. 方法定义时，形参的类型不用写,返回值类型也不写。

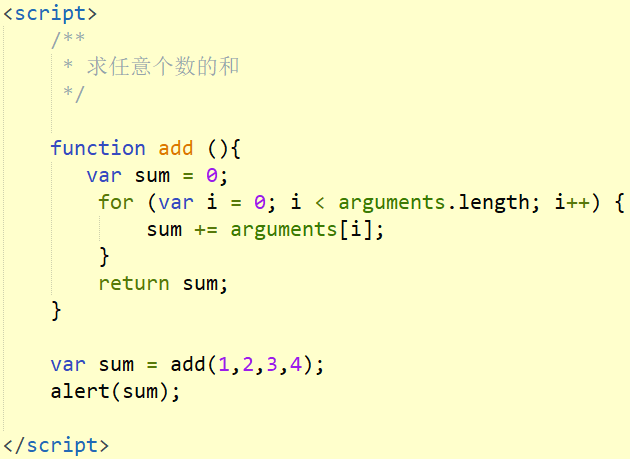
2. 方法是一个对象，如果定义名称相同的方法，会覆盖

|  |
| --- |
| var fun1 = function(){  alert(“1”);  }  fun1 = function(){  alert(“2”);  } |

3. 在JS中，方法的调用只与方法的名称有关，和参数列表无关

|  |
| --- |
| function method(a, b){  }  调用方法 :  method(10,20,30); |

4. 在方法声明中有一个隐藏的内置对象（数组），arguments,封装所有的实际参数



#### 4.1.4 调用格式

* 方法名称(实际参数列表);

### 4.2 JavaScript\_对象\_Array

#### 4.2.1 创建数组的三种写法

1. var arr = new Array(元素列表);

对比 : int[] arr = new int[]{11,22,33};

2. var arr = new Array(默认长度);

对比 : int[] arr = new int[3];

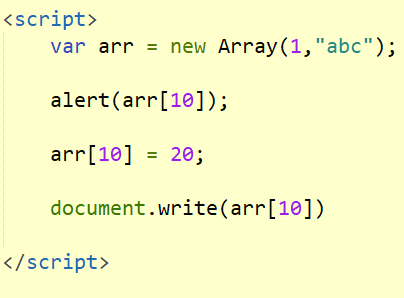
3. var arr = [元素列表];

对比 : int[] arr = {11,22,33};

#### 4.2.2 数组属性和特点

* 属性 : length:数组的长度
* 特点 :

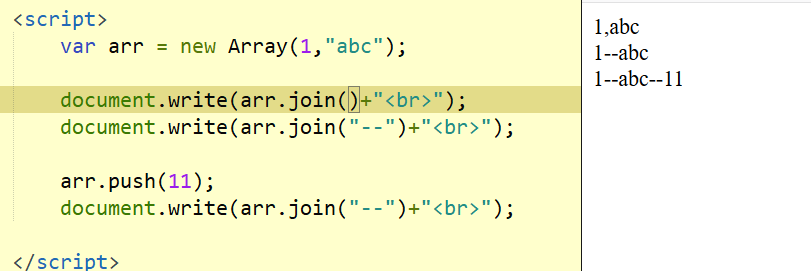
1. JS中，数组元素的类型可变的。  
 2. JS中，数组长度可变的。



#### 4.2.3 数组中的方法 :



* 示例代码 :



### 4.3 JavaScript\_对象\_Date

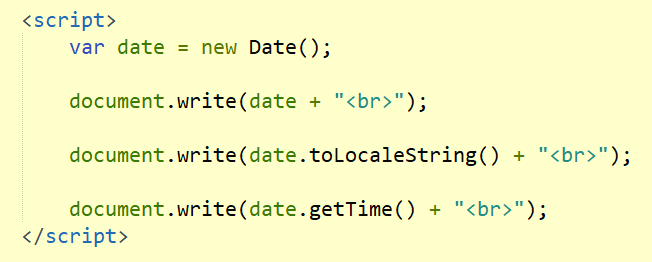
1. 创建：

var date = new Date();

2. 方法：

toLocaleString()：返回当前date对象对应的时间本地字符串格式

getTime():获取毫秒值。返回当前日期对象描述的时间到1970年1月1日零点的毫秒值差



### 4.4 JavaScript\_对象\_Math

1. 创建：

\* 特点：Math对象不用创建，直接使用。 Math.方法名();

2. 方法：

random():返回 0 ~ 1 之间的随机数。 含0不含1

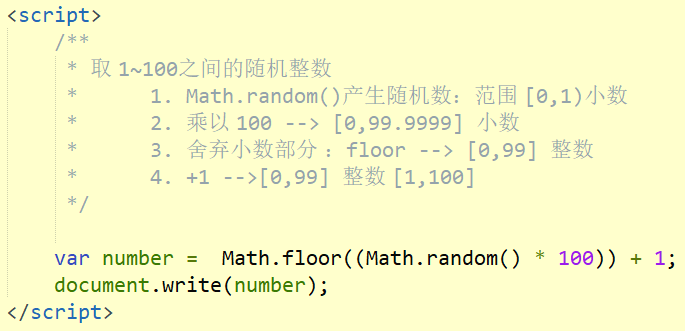
ceil(x)：对数进行上舍入。

floor(x)：对数进行下舍入。

round(x)：把数四舍五入为最接近的整数。

3. 属性：

PI



### 4.5 JavaScript\_对象\_RegExp1(语法规则)

* 正则表达式：定义字符串的组成规则。
* 单个字符:[]

如： [a] [ab] [a-zA-Z0-9\_]

\* 特殊符号代表特殊含义的单个字符:

\d:单个数字字符 [0-9]

\w:单个单词字符[a-zA-Z\_0-9]

* 量词符号：

?：表示出现0次或1次

\*：表示出现0次或多次

+：出现1次或多次

{m,n}:表示 m<= 数量 <= n

\* m如果缺省： {,n}:最多n次

\* n如果缺省：{m,} 最少m次

### 4.6 JavaScript\_对象\_RegExp2(JS对象创建)

* 开始结束符号:

^:开始

$:结束

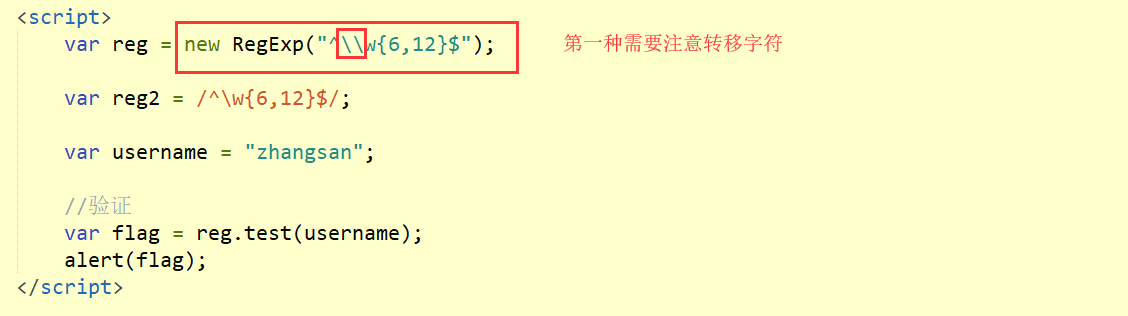
1. 创建

1. var reg = new RegExp("正则表达式");

2. var reg = /正则表达式/;

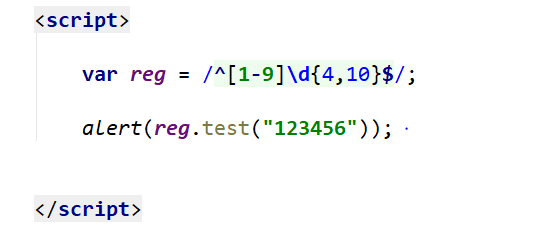
2. 方法

1. test(参数):验证指定的字符串是否符合正则定义的规范



* 注意 :

1. 如果使用的是简化写法创建正则对象, 那么编写正则语句的时候, 不能加入转移字符.  
  
 2. 使用JS正则表达式对象, 必须要有开始和结束标记符.



#### 4.6.1 练习

A. QQ正则

[1-9]\\d{4,10}

B. 手机号正则

[1][34578]\\d{9}

C. 邮箱正则

\\w+@[a-zA-Z0-9]+\\.[a-z]{2,3}

### 4.7 JavaScript\_对象\_Global

* 特点：全局对象，这个Global中封装的方法不需要对象就可以直接调用。  方法名();
* URL编码 : 传智播客 =  %E4%BC%A0%E6%99%BA%E6%92%AD%E5%AE%A2

encodeURI():url编码

decodeURI():url解码

encodeURIComponent():url编码,编码的字符更多

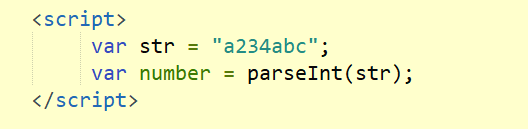
decodeURIComponent():url解码

|  |
| --- |
|  |



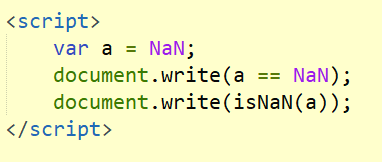
parseInt():将字符串转为数字

\* 逐一判断每一个字符是否是数字，直到不是数字为止，将前边数字部分转为number



isNaN():判断一个值是否是NaN

\* NaN六亲不认，连自己都不认。NaN参与的==比较全部问false



eval():讲 JavaScript 字符串，并把它作为脚本代码来执行。

