# DAY01 JavaScript介绍、 对象、变量和运算符

2023年4月14日 11:42

## 一.javascript的概述

1.34岁的系统程序员Brendan Eich(布兰登·艾奇)

2.Javascript：java(蹭热度) + script(脚本)

脚本语言：比企业级(java,c#,python)语言简单,而且是目前世界上最流行的语言，总的语言榜一般排在6，7位的样子。

3.将函数提升到"第一等公民"(first class)的地位。

4.JavaScript能干很多事情，考虑是否合适。

## 二.JavaScript的核心组成

### 1.ECMAscript

ECMA:欧洲的计算机协会组织

定义了javascript的语法规范,描述了语言的基本语法和数据类型。

**2.BOM**：浏览器对象模型，有一套成熟的可以操作浏览器(和网页内容没有关系)

javascript是一门轻量级，解释型的语言

轻量级：比企业级(java,c#,python)语言简单，是脚本语言

解释型：依靠浏览器进行编译解释，没有浏览器目前无法测试

**3.DOM**：文档对象模型，有一套成熟的可以操作页面元素的 API，通过 DOM 可以操作页面中的元素(专门操作网页内容)

html

css

## 三.前端(浏览器端，客户端)

用户可见的内容都是前端做的。

核心技术：html css javascript(vue,react,小程序)

### JavaScript核心工作

#### 一.前端核心工作组成部分

1.数据渲染（根据数据设计html结构，渲染页面，展示页面.....）

2.用户体验（网页效果，表单验证，其他功能.....）

3.性能优化（精灵图，懒加载 ，代码的压缩合并，图片的压缩合并，css的预编译.....）

## 二.JavaScript代码的书写位置(类似于css的写法)

1.外链式JS代码（推荐）

<script src="js文件的路径"></script>

2.内嵌式JS代码，内嵌式的js代码会在页面打开的时候直接触发

注意：

script 标签可以放在 head 里面也可以放在body里面

尽量将js代码写在body标签的内部的下面

这个和页面的加载顺序有关。

3.行内式 JS 代码（不推荐）写在标签上的 js 代码需要依靠事件（行为）来触发

```
<div class="box" onclick="alert('我是行内的js代码')">我是box里面的内容</div>
```

# JavaScript里面的注释和分号

## 一.JS里面的注释

学习一个语言，先学习一个语言的注释，因为注释是给我们自己看的，也是给开发人员看的.

写好一个注释，有利于我们以后阅读代码.

```
//双斜杠叫做单行注释
/*
  多行注释
  多行注释
  多行注释
  多行注释
*/
```

## 二.JS代码里面的分号

分号表示语句的结束，如果一行写多条语句，必须添加分号，一行写一条语句可以省略分号

注意：要添加分号尽量做到都添加，否则可以省略。

# 对象的认识

**核心的名称解读：**

一.类：表示一类事物，类是抽象(不具体)，不会占据内存，核心的意义是用来生成对象的

例如：人类，动物类，植物类

二.对象：类下面的具体的事物，真实存在，占据内存的。

例如：刘德华

三.属性和方法：对象一般都有属性和方法

属性：修饰对象的，描述的是对象的特点(私有的)

刘德华的身高体重肤色...都是属性

方法：对象具有的功能(公有的)

刘德华会走，会跑，会唱歌，会演戏...

## 一.JavaScript的对象

1.window对象：浏览器(类)下面的对象,javascript最大的对象,系统(JS语法)提供的。

window对象下面的属性和方法在使用的时候，允许省略window不写。

### 1.1.alert方法

功能是弹出一个信息框，里面展示自定义的一些内容，阻止后续代码继续执行。

通过点符号调用对象的属性和方法

属性和方法的区别是方法后面多一个括号，括号里面可以输入一些不同的内容。

```
window.alert(1 + 1);
window.alert(123 + 456);
window.alert('hello，你好1');
window.alert('hello，你好2');
alert('可以省略window不写，不影响效果');
```

### 1.2.prompt方法

功能是弹出一个输入框，类似于表单，里面输入的内容都是字符串(数字字母中文放在一个引号里面)

括号里面都可以添加提示文字

```
window.prompt('请输入你的年龄：');
```

### 1.3.confirm方法

功能是弹出一个选择框，提供选择使用的。

括号里面都可以添加提示文字

确定返回真的含义

取消返回假的含义

window.confirm('你确定要删除吗?');

2.console对象，window对象下面的子对象，浏览器控制面板下面的一个常用的对象。

console面板下面允许执行以及编写JavaScript代码

javascript代码调试非常好一个平台

2.1.log方法：将括号里面的内容输出到console平台下面，同时输出多个。

```
console.log(1 + 2);//3
console.log(1 + 3);//4
console.log(1 + 4);//5
console.log(1 + 2, 2 + 3, 4 + 5);//3 5 9
```

注意：所有的编程都是以数值编程为基础，前期会学习数学计算方面的一些代码。

**变量**

**1.变量的概念**

特定的标识符(自定义的名称)存储经常使用的一些值

特定的时间存储特定值的一个标识符

语法： var 变量名 = 值

**2.初始化**

第一次给变量赋值，叫做初始化

```
var n1 = 10;//初始化
n1 = 20;//改变
console.log(n1);
```

**3.变量的特点 - 记住这5个特点**

3.1.变量的值是可以改变的

3.2.变量是松散类型的，可以用来保存任何类型的数据

解读：其他的计算机语言声明变量必须先告诉变量名你存储的数据是什么类型的(比如是数字，还是字符串)

JavaScript的变量不需要告诉变量名你存储的是什么类型

任何类型都可以存储

3.3.变量提升(重点)

变量在声明之前也可以使用，但是值统一是undefined(未定义，未初始化)

代码正常的执行顺序是从上往下。

```
console.log(num);//undefined
var num = 100;
console.log(num);//100
```

3.4.同时声明多个

同时声明多个变量，采用简洁的写法，注意中间是逗号分隔，分号表示代码的结束。

```
var a = 1, b = 2, c = 3; //同时声明多个变量，采用简洁的写法
```

3.5.写入内存

变量会写入内存，整个script代码块内，都可以随意的使用变量

```
console.log(username);
```

**变量的命名规则(必须遵守) - 重点**

1.一个变量名称可以由 数字、字母、英文下划线（_）、美元符号（$）组成

2.严格区分大小写

3.不能由数字开头，会报错

4.不要出现空格，会报错

5.不能是 保留字 或者 关键字

关键字:作者约定的具有特殊用途的单词

保留字:未来的关键字

```
var var = 1;
```

**变量的命名建议遵守 - 重点**

1.变量名尽量有意义（语义化）

2.遵循小驼峰命名规则，由多个单词组成的时候，从第二个单词开始首字母大写

**变量的数据类型以及检测 - 重点**

为什么学习数据类型：

声明变量，给变量赋值的内容一定要属于下面的类型之一。

**1.变量有那些数据类型 - 面试题**

1.1.基本类型(简单类型)：number(数字) string(字符串) boolean(布尔值) null(空对

象）undefined(未赋值）Symbol(后面讲）BigInt(后面讲)

1.2.复杂类型(引用类型)：object(对象)

**2.数据类型**

2.1.number(数字类型)

了解二进制数字：0-1组成，0b开头的数字

```
var num1 = 0b110;
console.log(num1);//6
```

了解八进制数字：0-7组成，0开头的数字

```
var num2 = 016;
console.log(num2);//14
```

十进制数字:0-9组成，生活中的数字

十六进制数字:0-9和a-f组成，0x开头的数字

```
var num3 = 0xabc;
console.log(num3);//2748
```

NaN：当计算无法得到结果，输出NaN(不是一个数字not a number)

```
console.log('a' - 'b');//NaN
```

2.2.string(字符串类型)

被引号包裹的所有内容（可以是单引号也可以是双引号）

2.3.boolean(布尔类型)

只有两个值（true真 或者 false假）

2.4.特殊类型：null(空对象)

声明变量无法确认变量存储的是什么类型的值，可以赋值为null

只有一个，就是null，表示空的意思(内存占位)

```
var element = null;//element无法确定初始值，将null赋值给element
```

2.5.特殊类型：undefined(未初始化,未定义)

未初始化 - 变量

```
var num;
console.log(num);//undefined,未初始化
```

未定义 - 属性

```
console.log(window.hehe);//undefined 未定义 hehe属性在window对象上面没有定义。
```

**如何检测变量的数据类型。**

1.检测变量类型的方法：typeof( 变量名 ) 或者 typeof 变量名

注意：typeof返回变量值的类型，typeof输出的值又是什么类型(string类型)。

规律：typeof两次，结果一定是string

**数学运算符（+ - * / %)**

+ - * / %都具有隐式转换，将字符串格式的数字转换成真正的数字

1.+号(三层含义)

1.1.加法运算，两边任意一边都不能有字符串，具有隐式转换(系统自动转换)

```
console.log(1 + 3);//4
console.log(true + 1);//2 将true转换成数字1  进行运算
console.log(true + false);//1  1 + 0 = 1  false转换成数字0
```

1.2.拼接符号，两边任意一边有字符串，将符号两边的值拼接起来，最终形成一个字符串(重要的应用)

```
console.log('1' + 100); //1100
console.log(true + '100'); //true100
var username = '张三';
console.log('我的名字叫' + username);//核心应用，username是变量  我的名字叫张三
```

1.3.正号使用，将任何类型的值转换成数字，如果无法转换，输出NaN(不是一个数字)

数字格式的字符串(字符串里面的都是数字)，将其转换成真正的数字。

```
console.log(+true);//1
console.log(+'hehe');//NaN

var num = '100';
console.log(num);// 字符串100
console.log(+num);// 数字100
```

利用prompt输入两个数字，计算它们的和。

prompt输入的值一定是字符串格式

```
var n1 = prompt('请输入一个数字：');
var n2 = prompt('请再输入一个数字：');
console.log(typeof n1);//string
console.log(typeof n2);//string
console.log(+n1 + +n2);
```

### 1.4.特殊的测试

IEEE754标准：小数误差造成的计算问题

涉及下面的这些运算。

都会选择将数字扩大10倍，将运算结果缩小10倍。

```
console.log(0.1 + 0.2);//0.30000000000000004
console.log(0.2 + 0.3);//0.5
console.log(0.2 + 0.4);//0.6000000000000001
```

### 1.5.特殊的测试

```
console.log(+null);//0
console.log(+undefined);//NaN
console.log(null == undefined);//true 作者的约定
```

### 2.%:求余数，求模

+ - * / %都具有隐式转换，将字符串格式的数字转换成真正的数字

```
console.log(9 % 2);//1
console.log(2 % 9);//2 如果前面的数比后面的数小，结果就是前面的数字
console.log('9' % 2);//1
console.log('9' % '2');//1

// 应用：num是任意的正整数，那么下面的结果肯定在0，1，2，3，4范围内。
// console.log(num % 5);

// 应用：求偶数以及整数的思维
// console.log(4 % 2 == 0);//true
// console.log(4.2 % 1 == 0);//false
// console.log(4 % 1 == 0);
```

### 3.- * / 正常的操作

```
console.log(3 - 2);//1
console.log(3 - 20);//-17
console.log('3' - '20');//-17
console.log('3' * '20');//60
console.log('3' / '20');//0.15
console.log(10 / 0);//Infinity无穷的意义
```

总结：

重点关注：+ 和 %

**赋值运算符(=)**

javascript使用=运算符来给变量或者属性赋值（最低优先级）

就是把=右边的赋值给等号左边的变量名

```
var num = 10;//真正的含义：将右边的数字10赋值左边的变量num
num = num + 20;//右边的结果赋值给左边。
console.log(num);//30
```

**复合的赋值运算符(+= -= *= /= %=)**

复合的赋值运算符使用更快捷，更优

**1.+=最重要的符号**

迭代：将上一次的结果，当作下一次的初始值。

```
var num = 1;
num += 2;// 相当于num = num + 2;
console.log(num);//3

var str = 'zhangsan';
str = 'lisi';//重新修改变量的值
console.log(str);//lisi

var str = 'zhangsan';
str += 'lisi';//迭代  str = str + 'lisi'===> str = 'zhangsan' + 'lisi'
console.log(str);//zhangsanlisi
```

**2.其他的-= *= /= %=**

```
var num = 1;
num -= 1;
num *= 10;
num /= 10;
console.log(num);//0
```

**变量不同类型之间的转换**

**1.其他数据类型转成数值**

1.1.Number(变量):Javascript天生具有的函数。

可以把一个变量强制转换成数值类型

可以转换小数，会保留小数

可以转换布尔值

遇到不可转换的都会返回NaN

### Number和正号+的区别

一个是函数，一个是符号

大部分情况下，正号(+)可以取代Number函数

### 1.2.window.parseInt(变量) - 非常重要

从第一位开始检查，是数字(字符串格式的数字)就转换成数字，直到一个不是数字的内容就结束

开头就不是数字(字符串格式的数字)，那么直接返回NaN

不认识小数点，只能保留整数

### 1.3.window.parseFloat(变量) - 重要

从第一位开始检查，是数字(字符串格式的数字)就转换，直到一个不是数字的内容

开头就不是数字(字符串格式的数字)，那么直接返回 NaN

认识一次小数点

## 2.其他数据类型转成字符串

### 2.1.变量.toString():Javascript天生具有的方法 - 重点

有一些数据类型不能使用toString()方法，比如undefined和null

### 2.2.String(变量):Javascript天生具有的函数。

任何类型的值转字符串

### 2.3.利用拼接空字符串实现 - 重点

```
var num = 100;
console.log(num + '');//100变成字符串
```

## 3.其他数据类型转成布尔

### 3.1.Boolean(变量):Javascript天生具有的函数。

任何类型的值转布尔值

在 js 中，只有 ''、0、null、undefined、NaN，这些是 false

字符串非空即真，数字非零即真

```
// 特殊情况
// console.log(Boolean('0'));//true
// console.log(Boolean(' '));//true(包含空格)
```

### 总结：

熟悉提供三个转换函数：Number() String() Boolean() （进行强制转换）

重点关注：parseInt() parseFloat()

特殊关注：toString()

**关系运算符（结果为布尔值）**

<、>、<=、>=、==( 相等 )、===（全等）、!=(不相等) !==（不全等）

**1.注意点**

结果为布尔值

比较的核心一般都是基于数字(特殊情况除外)

```
// console.log(5 >= 3);//true  解读5大于或等于3
```

**2.特殊情况(字符串和unicode编码) - 重点**

2.1.如果是字符串格式的数字和真正的数字比较，还是基于数字,将字符串格式的数字转换成真正的数字

```
console.log('6' > 5);//true
```

2.2.如果字符串和字符串进行比较，比较是通过unicode编码进行比较

unicode编码:全世界大部分国家的文字都对应有一个数字编码。

0-9 unicode 48-57

A-Z unicode 65-90

a-z unicode 97-122

```
// console.log('zhangsan' > 'lisi');//true
// console.log('10000' > '9');//false
```

2.3.如果是字符串和数字比较，还是基于数字，将字符串转换成NaN，NaN和任意类型的值比较都是false

2.4.布尔值，null,undefined和数字进行比较，还是基于数字。

```
// console.log(true > 0); //true  1>0
// console.log(false < 1); //true  0<1
// console.log(null < 1);// true  0<1
// console.log(undefined < 1);//false   NaN<1
```

**3.==( 相等 )、!=(不相等) ===（全等）、!==（不全等） - 重点**

3.1.== 和 != 表示比较的是值,和数据类型没有关系，具有隐式转换(系统自动转换)

3.2.===（全等）、!==（不全等）

比较的值和类型必须相等，没有任何相关的转换。

```
// 特殊的NaN
// console.log(NaN == NaN);//false
// console.log(NaN === NaN);//false
```

**逻辑运算符 - 非常重要**

在 js 中，只有 ''、0、null、undefined、NaN，这些是 false

### 1.逻辑与(&&) - 和 - and

操作数转成布尔值进行运算符，但是结果不一定是布尔值。

符号两边的操作数，如果第一个是真的，继续执行第二个操作数，结果就是第二个操作数

符号两边的操作数，如果第一个是假的，直接代码短路，结果就是第一个操作数

```
// console.log(5 && 6);//6
// console.log(0 && 6);//0
```

短路

即如果第一个操作数能够决定结果，那么就不会再对第二个操作数求值

```
// console.log(0 && hehehehehhehehehehe);//0
// console.log('' && hehehehehhehehehehe);//''(console面板下面不可见的)
```

### 2.逻辑或(||) - 或 - or

操作数转成布尔值进行运算符，但是结果不一定是布尔值。

符号两边的操作数，如果第一个是真的，直接代码短路，输出结果

如果第一个是假的, 继续执行第二个操作数，结果是第二个操作数

```
// console.log(1 || 2 || 3 || 4);//1
// console.log(0 || 1);//1
// console.log(0 || false);//false
```

规律(重点理解)

逻辑与碰到假的就结束，逻辑或碰到真的就结束(除非都是假或者都是真除外)

```
// console.log(3 && false);//false
// console.log(3 && 4 && 0 && 5);//0
// console.log(null && 4 && 0 && 5);//null

// console.log(false || 0 || 5 || null);//5
// console.log(1 || 0 || 5 || null);//1
```

### 3.逻辑非！

逻辑非操作符由一个叹号!表示，无论这个值是什么数据类型，逻辑非操作符首先会将它的操作数转换为一个布尔值，然后再对其求反。

## 一元运算符

只能操作一个值的操作符叫做一元操作符，是 ECMAScript 中最简单的操作符。

++ -- 本身的含义对操作数进行加1或者减1

### 1.++

前置：符号在操作数的前面

后置：符号在操作数的后面

前置和后置指的是当前的操作数在参与赋值(运算)的时候会有区别，自身都是+1

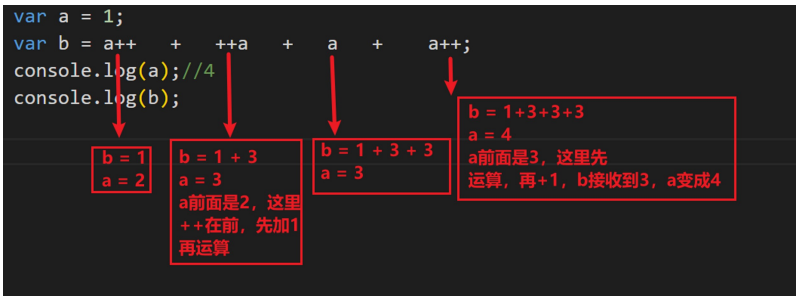
```
// var a = 1;
// var b = a++; //先将a赋值给b，然后a再+1      b = 1   a = 2
// var c = ++a; //先将a+1,然后再赋值给c        c = 3   a = 3

// 特殊情况
// var num = 1;
// console.log(num++);//1
// console.log(num);//2

// var num1 = 1;
// console.log(++num1);//2
// console.log(num1);//2


// 案例:

// var a = 1;
// var b = a++ + ++a + a + a++;
// console.log(a);//4
// console.log(b);//10
```




**NaN的概念及应用**

**1.当数学计算无法得到数字结果，该变量的值为NaN(not a number)**

```
// console.log('a' - 'b');//NaN
// console.log(typeof NaN);//number  理解虽然意义是表示"不是一个数字",但还是属于数字类型
// console.log(NaN == NaN);//false
```

**2.isNaN(num):**

该方法判断num变量的值是否是NaN，结果是布尔值，如果num不是一个数字输出

true

```
// console.log(isNaN(NaN));//true
// console.log(isNaN('abc'));//true
// console.log(isNaN('100px'));//true
```

**类，对象，属性，方法**

类->对象(属性和方法)

**1.document对象(文档对象),是window的子对象。**

document是html文件中最大的一个对象，document对象里面包含html

```
// console.log(document);
```

1.1write()方法:将括号里面的信息输出到浏览器内容区域。同时能够解读标签

```
// document.write('我是document.write输出的内容');
// document.write('<h1>我是document.write输出的内容</h1>');
// document.write('<h1 style="color:red;">我是document.write输出的内容</h1>');
```