网易微专业之《前端开发工程师》

学习笔记

开始时间：2015.12.28

《JavaScript 程序设计》

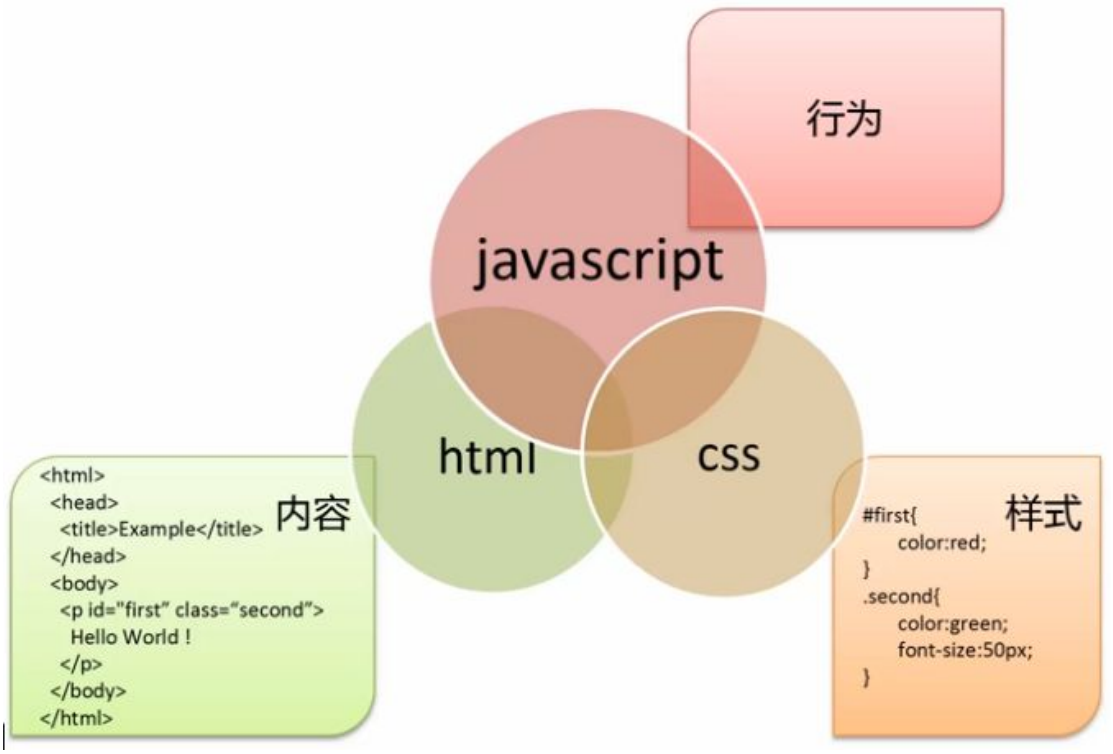
基础篇

一、JS 介绍

JavaScript，就是我们通常所说的 JS，是一种嵌入到 HTML 页面中的脚本语言，由浏览器一边解释一边执行。

HTML、CSS 和 JavaScript 的关系如下：

“HTML 是网页的结构，CSS 是网页的外观，而 JavaScript 是页面的行为。”

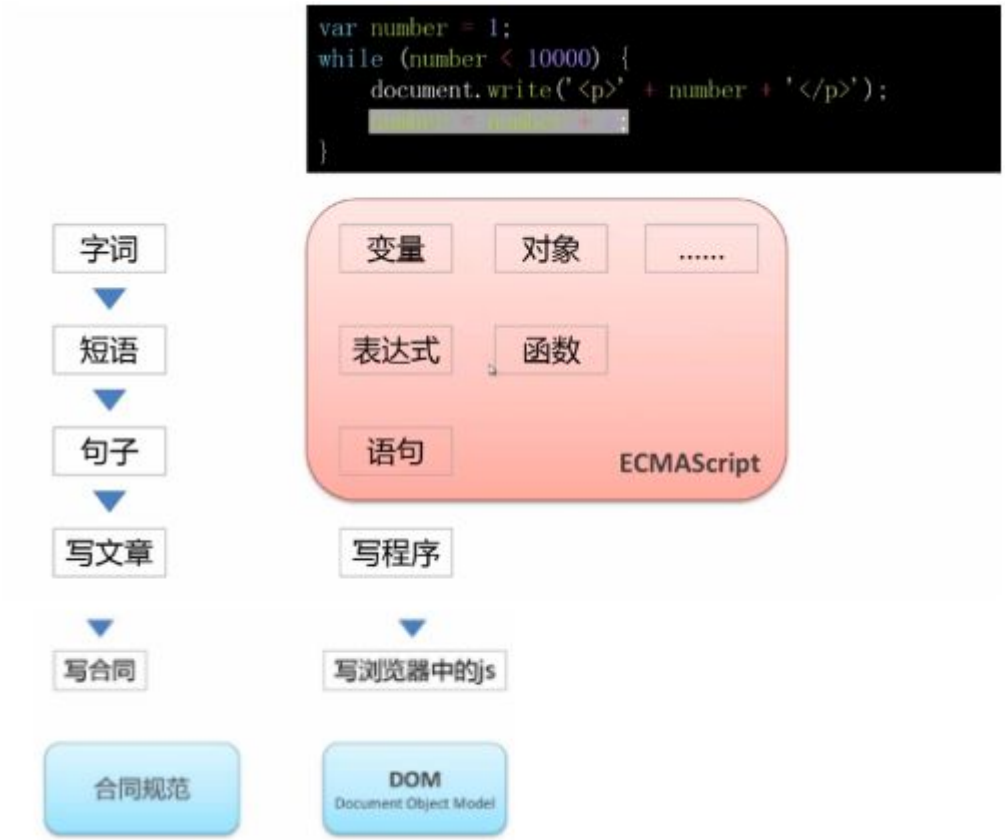


JS 特性：

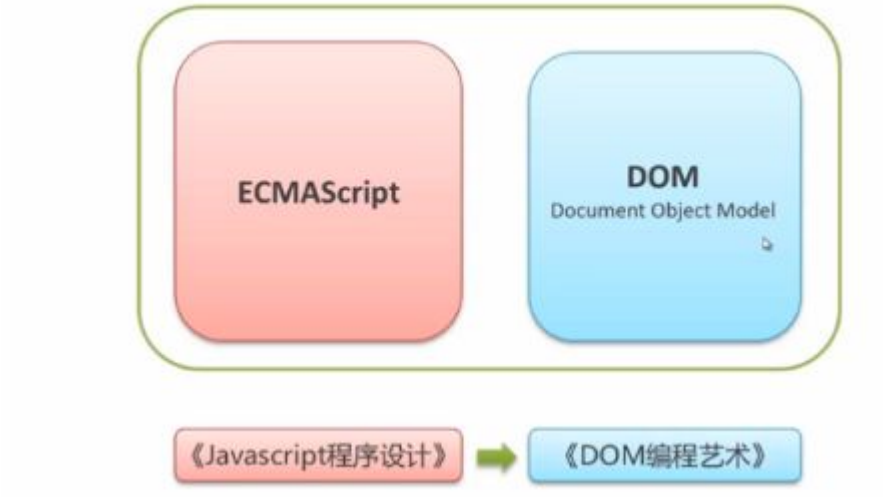
- 运行环境：Chrome,Firefox,IE……

- 解释型。

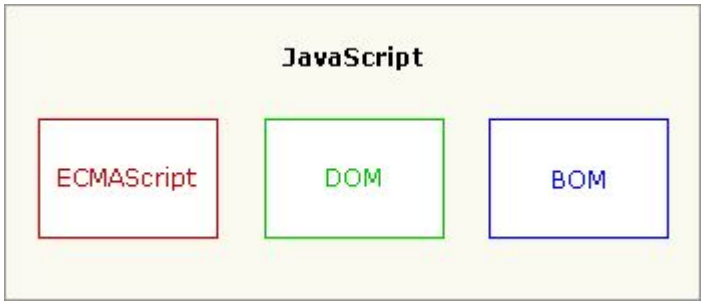
如何写程序（以写文章相类比）



浏览器中的 JS



JavaScript 是由 ECMAScript，DOM 和 BOM 三者组成的。



JavaScript 的核心 ECMAScript 描述了该语言的语法和基本对象；

ECMAScript:实际上是一种脚本在语法和语义上的标准,它描述（语法、类型、语句、关键字、保留字、运算符、对象），定义脚本语言的所有属性、方法和对象。

DOM 描述了处理网页内容的方法和接口；

DOM（文档对象模型,**Document Object Model**）是 HTML 和 XML 的应用程序接口（API）。DOM 将把整个页面规划成由节点层级构成的文档。HTML 或 XML 页面的每个部分都是一个节点的衍生物。

DOM 通过创建树来表示文档，从而使开发者对文档的内容和结构具有空前的控制力。用 DOM API 可以轻松地删除、添加和替换节点。

BOM 描述了与浏览器进行交互的方法和接口。

BOM（浏览器对象模型,**Browser Object Model**），可以对浏览器窗口进行访问和操作。使用 BOM，开发者可以移动窗口、改变状态栏中的文本以及执行其他与页面内容不直接相关的动作。

BOM 主要处理浏览器窗口和框架，不过通常浏览器特定的 JavaScript 扩展都被看做 BOM 的一部分。这些扩展包括：

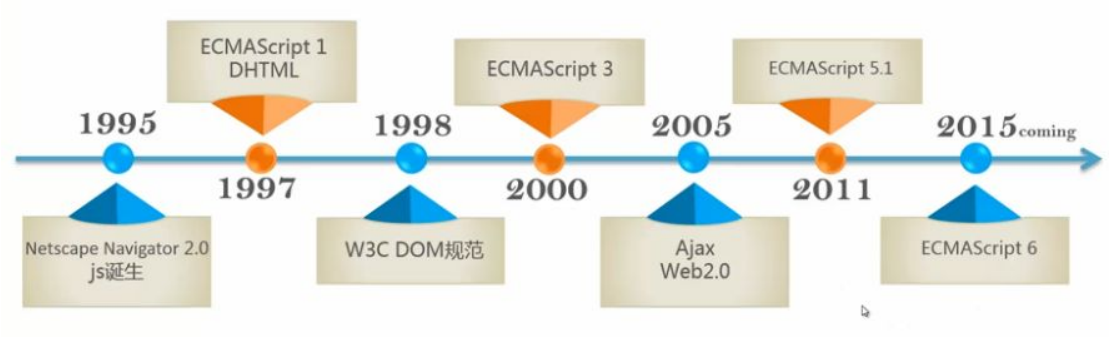
- 弹出新的浏览器窗口
- 移动、关闭浏览器窗口以及调整窗口大小
- 提供 Web 浏览器详细信息的定位对象
- 提供用户屏幕分辨率详细信息的屏幕对象
- 对 cookie 的支持
- IE 扩展了 BOM，加入了 ActiveXObject 类，可以通过 JavaScript 实例化 ActiveX 对象

由于没有相关的 BOM 标准，每种浏览器都有自己的 BOM 实现。有一些事实上的标准，如具有一个窗口对象和一个导航对象，不过每种浏览器可以为这些对象或其他对象定义自己的属性和方法。

ECMA/DOM/BOM 三者浏览器兼容性：

- ECMA　　几乎没有兼容性问题
- DOM　　有一些操作不兼容
- BOM　　没有兼容问题（完全不兼容）

历史：



二、JS 调试

javascript 调试最常用的是全局方法 `console.log()`，可以用它输出想要跟踪查看的变量，或者单纯地输出一些字符串说明调试语句所在的代码段有没有被执行到。

浏览器的开发者工具，在控制台中可以查看到当前页面的输出信息（通过 `console.log()`），以及 javascript 错误。错误信息还会说明错误位置（文件名，行，列），错误类别，错误说明。

如何打开浏览器调试面板？

F12/Ctrl+Shift+!/右键“审查元素”

变量输出：alert(变量)；或者：console.log(变量)；

Ctrl+Shift+O:查找函数；

Ctrl+O:查找文件

ESC:调出 Console 面板；

FireFox 调试工具：firebug

三、JS 基本语法

每一种计算机编程语言都有自己的数据结构，JavaScript 脚本语言的数据结构包括：标识符、常量、变量、保留字等。

（一）JS 数据结构

1. 直接量（常量）

常量，顾名思义就是指不能改变的量。常量的指从定义开始就是固定的，一直到程序结束。常量主要用于为程序提供固定和精确的值，包括数值和字符串，如数字、逻辑值真（true）、逻辑值假（false）等都是常量。

2. 变量

变量，顾名思义，就是指在程序运行过程中，其值是可以改变的。

- **变量的命名：**变量的名称实际上是一个标识符，因此命名一个变量时也要遵循标识符的命名规则。
- **变量的声明与赋值：**在 JavaScript 中，使用变量之前需要先声明变量。

“所有的 JavaScript 变量都由关键字 var 声明。”

声明变量：var 变量名；

在声明变量的同时，也可以对变量进行赋值。一个关键字 var 也可以同时声明多个变量名，变量名之间必须用英文逗号“,”隔开。

3. 标识符：

标识符，说白了，就是一个名字。在 JavaScript 中，变量和函数等都需要定义一个名字，这个名字就可以称为“标识符”。

JavaScript 语言中标识符最重要的 3 点就是：

- 以字母、下划线或者美元符合（$）开头
- 由字母、下划线、美元符合（$）和数字组成
- **变量名不能包含空格、加号、减号等符号；**
- 不能使用 JS 关键字和保留字

以下图片中的 age、add、num1、num2、name 等都是标识符。

```
var age = 12;
function add (num1, num2) {
    return num1 + num2;
}
var student = {
    name:"小明"
}
```

4. 关键字和保留字

JavaScript 关键字是指在 JavaScript 语言中有特定含义，成为 JavaScript 语法中一部分的那些字。JavaScript 关键字是不能作为变量名和函数名使用的，也就是说变量的名称或者函数的名称不能跟系统的关键字重名。使用 JavaScript 关键字作为变量名或函数名，会使 JavaScript 在载入过程中出现编译错误。

- 关键字

break	do	instanceof	typeof
case	else	new	var
catch	finally	return	void
continue	for	switch	while
debugger*	function	this	with
default	if	throw	delete
in	try		

- 保留字

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
Char	final	native	synchronized
Class	float	package	throws
Const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

（二）JS 基本语法

1. 大小写敏感

JavaScript 是严格区分大小写的。

```
var age = 10;
var Age = 20;


document.write(age); //10
document.write(Age); //20
```

2. 语句：

赋值语句，条件语句，循环语句

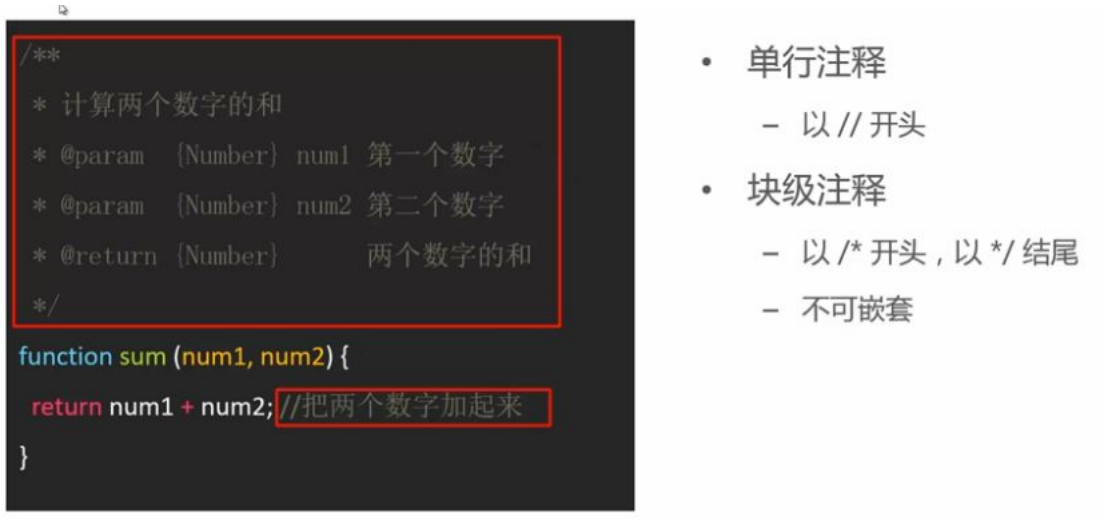
```
var sex = 1;

if (sex == 1){
   document.write('男');
} else {
   document.write('女');
}
...
while (number < 10000) {
   document.write('<p>' + number + '</p>');
   number = number + 1;
}
...
```

共同点：

- 以分号结尾；
- 语句可以嵌套；
- 多个语句可以放在一个代码块中，以{}符合包括起来。

3. 注释

在编写 JavaScript 代码时，我们经常要在一些关键代码旁做一下注释，这样做的好处很多，比如：方便理解、方便查找或方便项目组里的其它程序员了解你的代码，而且可以方便以后你对自己代码进行修改。



总结：

HTML 注释：<!-- -->

CSS 注释：/* */

4. 命名规范

- 可读性——能看懂
- 规范性——符合规则

匈牙利命名法

- 类型前缀
- 首字母大写

类型	前缀	类型	实例
数组	a	Array	aItems
布尔值	b	Boolean	bIsComplete
浮点数	f	Float	fPrice
函数	fn	Function	fnHandler
整数	i	Integer	iItemCount
对象	o	Object	oDiv1
正则表达式	re	RegExp	reEmailCheck
字符串	s	String	sUserName
变体变量	v	Variant	vAnything

如 aDiv[]，表示数组；oDiv，表示一个对象；sDiv,表示一个字符串。通过这样的“类型前缀+首字母大写”的命名方式，提高代码可读性。

四、JS 基本类型

每一种计算机语言除了有自己的数据结构外，还具有自己所支持的数据类型。JavaScript 跟传统编程语言不同，它采用的是弱数据方式，也就是说一个数据不必首先做声明，可以在使用或赋值时再确定其数据类型，当然也可以先声明该数据类型。

javascript 中一共有 5 种简单数据类型（也称为**基本数据类型**），和 1 种复杂数据类型（也称为**引用数据类型**）

JavaScript 数据类型包括：

1. **Number(数字型)**
2. **String**（字符串行）
3. **Boolean**（布尔型）
4. **Object** （引用数据类型/对象型）
5. **Null**（空值）
6. **Undefined**（未定义值）

```
var typeA = undefined, //Undefined类型 这里写成 var typeA; 也是一样的
    typeB = null, //Null类型 空
    typeC = false, //Boolean类型 逻辑值
    typeD = "twitter", //String类型 字符串
    typeE = 1.5, //Number类型 数字

    typeF = new Object(), //Object类型 对象
    typeG = ["elem1", "elem2"], //Array类型 数组
    typeH = new Date(2013, 7, 19), //Date类型 日期
    typeI = /^\stext\s$/, //RegExp类型 正则表达式
    typeJ = function(){}; //Function类型 函数
```

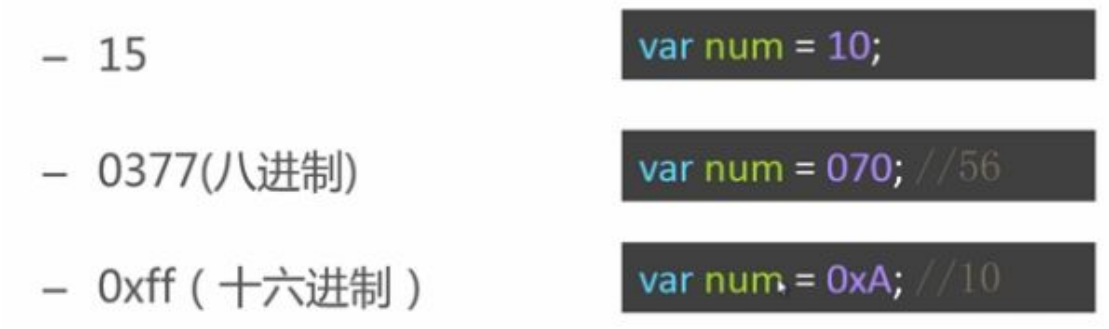
javascript 是以 Object 为基础的语言，除基本数据类型外，其他所有的引用数据类型，本质上都是 Object。

1. Number

数字（Number）是最基本的数据类型。

- **整型数据**

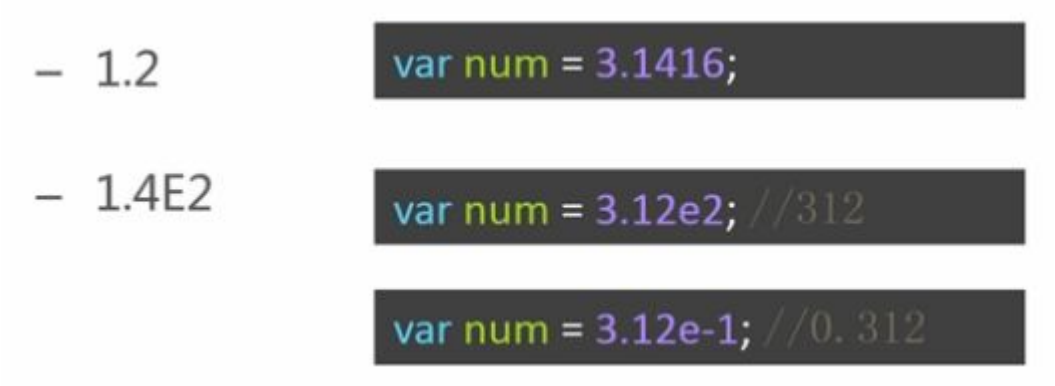
整型数据指的是数据形式是十进制整数（也包括十六进制和八进制），整数可以为正数、0 或负数。例如，“0、4、-5、1000”这些都是“整型数据”。



- **浮点型数据**

浮点型数据是指带有小数的数据。

浮点数还可以使用指数法表示，即实数后跟随字母 e 或 E，后面加上正负号，其后再加一个整型指数。这种计数法表示的数值等于前面的实数乘以 10 的指数次幂。



- **特殊值：**

NaN(Not a Number)：即“非数字”。当在程序中由于某种原因发生计算错误后，将产生一个没有意义的数字，此时 JavaScript 返回的数字值就是 NaN。

NaN==NaN，返回的结果为 false.

Infinity：如 var num = 1/0; //Infinity

2. String（字符串型）

字符串是由 Unicode 字符、数字、标点符号等组成的序列，它是 JavaScript 用来表示文本的数据类型。程序中的字符串型数据是包含在单引号或双引号中的，由单引号定界的字符串中可以含有双引号，由双引号定界的字符串中也可以含有单引号。

如：var name=”hello”；

var month=’july’；

var num=’3.14’；

3. Boolean

布尔型数据类型只有 2 个：真（true）和假（false）。

0 可以看作 false，1 可以看做 true。

布尔值通常在 JavaScript 程序中用来比较所得的结果，例如：n==1；

这行代码测试了变量 n 的值是否和数值 1 相等。如果相等，比较的结果就是布尔值 true，否则结果就是 false。

什么是真、什么是假：

- 真：true、非零数字、非空字符串、非空对象
- 假：false、数字 0、空字符串、空对象、undefined

如：

```
var sex=true;
if(sex) {document.write(‘男’);}
else {document.write(‘女’);}
```

4. Object

Object 是一组无序的名值对的集合。

Object 是一组数据和功能的集合。一个简单的 Object 的示例：

```
var myObject = {}; //和 new Object() 相同

myObject.name = "yuki"; //属性

myObject.sayHello = function() { //方法

    alert("Hello! My name is " + this.name);};

    alert(myObject.name); // "yuki"

    myObject.sayHello(); // "Hello! My name is yuki"
```

这里 name 是 myObject 的一个属性，而 sayHello 是它的一个方法。可以看到，访问属性，或者调用方法，都是通过点语法.来实现的。这种使用方法你一定非常熟悉，jQuery 就是全局创建了一个名为 jQuery（如果不冲突，还有别名$）的 Object，然后把所有的方法都定义在了这个 Object 中。

5. Null

整型、浮点型这些数据在定义的时候，系统都会分配一定的内存空间。JavaScript 中的关键字 null 是一个特殊的值，它表示空值，系统没有给它分配内存空间。

如果试图引用一个没有定义的变量，则返回一个 null 值。这里要非常强调一点：null 不等同于空的字符串（""）或 0，因为空的字符串（""）或 0 是存在的，但是 null 表示其不存在的。

- 值：null
- 出现场景：表示不存在。如：var car=null;

6. Undefined

如果一个变量虽然已经用 var 关键字声明了，但是并没有对这个变量进行赋值，而无法知道这个变量的数据类型，因此这个变量的数据类型是 undefined，表示这是一个未定义数据类型的变量。

值：undefined

出现场景:

- 已声明未赋值的变量。如：

```
var a;

console.log(a); //undefined
```

- 获取对象不存在的属性：

```
var obj={a:1,b:2};

console.log(obj.c); //undefined
```

null 与 undefined 的区别是 ,null 表示一个变量被赋予了一个空值 ,而 undefined 则表示该变量尚未被赋值。

类型识别：

Typeof

在 JavaScript 中，typeof 运算符用于返回它的操作数当前所容纳的数据的类型，这对于判断一个变量是否已被定义特别有用。

```
var num;
typeof num; //undefined
```

```
var flag = true;
typeof flag; //boolean
```

```
var num = 1;
typeof num; //number
```

```
var cat = null;
typeof cat; //object
```

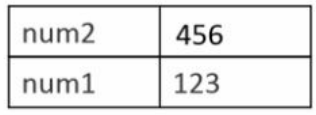
```
var num = '1';
typeof num; //string
```

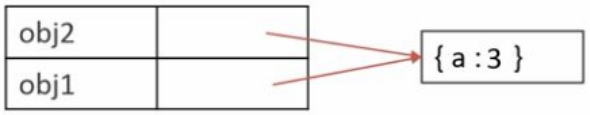
```
var cat = {name : 'kitty'};
typeof cat; //object
```

原始类型和引用类型

原始类型	引用类型
Number	Object
String	
Boolean	
Undefined	
Null	

原始类型与引用类型的区别：






```
var num1 = 123;
var num2 = num1;
num2 = 456;
console.log(num1);//123
```

```
var obj1 = {a:1};
var obj2 = obj1;
obj2.a = 3;
console.log(obj1.a);//3
```

五、JS 操作符与表达式

（一）JS 操作符（运算符）

- 一元操作符（++，--）
- 算术操作符（+，-，*，/,%）
- 关系操作符（>,<, >=, <=, ）
- 相等操作符（ ==, !=, ===, !== ）
- 逻辑操作符（ ！， &&， || ）
- 赋值操作符（=）
- 条件操作符（？：）
- 逗号操作符（，）
- 对象操作符（new, delete, ., [], instanceof, in ）
- 位操作符（ ~, &， |, ^, <<,>>,>>> ）

1.一元操作符：

- ++
- --

```
var age = 29;
++age; //30
age++; //31
```

自增运算符：++

“++”是自增运算符，它指的是在原来值的基础上加 1，i++表示“i=i+1”。该运算符有 2 种情况：

◆ **（1）i++：“i++”指的是在使用 i 之后，使 i 的值加 1。**

举例：

```
i=1;  j=i++;
```

上面执行的结果：j 的值为 1，i 的值为 2。

其实上面代码等价于下面这一段代码：

```
i=1;j=i;i++;
```

◆ **（2）++i：“++i”指的是在使用 i 之前，先使 i 的值加 1。**

举例：

```
i=1;j=++i;
```

上面的执行结果：j 的值为 2，i 的值为 2。

其实上面代码等价于下面这一段代码：

```
i=1;i++;j=i;
```

自减运算符

“--”是自减运算符，它指的是在原来值的基础上减 1，i--表示“i=i-1”。该运算符同样有 2 种情况：

（1）i--

（2）--i

举例：

i=6;j=i--;//j 的值为 6，i 的值为 5

i=6;j=--i;//j 的值为 5，i 的值为 5

2.算术操作符

+	加	4+6 //返回值 10
-	减	7-2 //返回值 5
*	乘	2*3 //返回值 6
/	除	12/3 //返回值 4
%	求余	7%4 //返回值 3

3.关系操作符

运算符	描述	示例
<	小于	1<4 //返回 true
>	大于	2>5 //返回 false
<=	小于等于	8<=8 //返回 true
>=	大于等于	3>=5 //返回 false

4.相等操作符

==	是否等于	0==false;//true;2==true;//false; “==0;//true
!=	是否不等于	5!=6 //返回 true
=== 和 !==		var num=”4”； num===4; //返回 false; num!==4; //返回 true

案例：

var a=5;var b='5';

alert(a==b); //true 先转换类型，然后比较

//alert(a===b); //false 不转换类型，直接比

var a='12';

var b='5';

alert(a+b); //结果为 125，不是 17 1.字符串连接 2.数字相加

alert(a-b); //结果为 7 1.数字相减

5.逻辑操作符

逻辑运算符通常用于执行布尔运算，它们常常和比较运算符一起使用来表示复杂比较运算，这些运算涉及的变量通常不止一个，而且常用于 if、while 和 for 语句中。

JavaScript 中常用的算术运算符

运算符	描述	示例
&&	逻辑与，若两边表达式的值都为 true，则返回 true；任意一个值为 false，则返回 false	(8>5)&&(4<6)，返回 true； (8<5)&&(4<6)，返回 false
||	逻辑或，只有表达式的值都为 false，才返回	(8<5)||(4<6)，返回 true；

JavaScript 中常用的算术运算符

运算符	描述	示例
	false，其他情况返回 true	(8<5)&&(4>6)，返回 false
!	逻辑非，若表达式的值为 true，则返回 false；若表达式的值为 false，则返回 true	!(9>2) 返回 false ;!(9<2) 返回 true

案例：

!	Var flag=true; alert(!flag);//false;
	Alert(!0);//true
	Alert(![]);//false
	Alert(!" ");//true
	Alert(!!1);//true

&&	Var result=true && false;
	Var result=true && 3; //3
	Var result=1&&3; //3
	Var result=[] && " "; //" "
	Var result=false && 3; //false
	Var result=" " && 3; //" "
	Var result=null && true; //null
	Var num=0; var result=" " && num++;//num0

||	Var result=true || false;
	Var result=1 || 3;//1
	Var result=[] || " ";//[]
	Var result=false || 0;//0
	Var result=" " || 3;//3
	Var result=null || true;//true
	Var num=0;var result=3 || num++;//num 0

6.赋值操作符

JavaScript 中的赋值运算可以分为 2 种：简单赋值运算和复合赋值运算。

简单赋值运算是将赋值运算符（=）右边表达式的值保存到左边的变量中。

复合赋值运算结合了其他操作（如算术运算操作）和赋值操作。

运算符	示例
=	author="helicopter"
+=	a+=b 等价于 a=a+b

运算符	示例
-=	a-=b 等价于 a=a-b
*=	a*=b 等价于 a=a*b
/=	a/=b 等价于 a=a/b
%=	a%=b 等价于 a=a%b
&=	a&=b 等价于 a=a&b（&是逻辑与运算）
|=	a|=b 等价于 a=a|b（|是逻辑或运算）
^=	a^=b 等价于 a=a^b（^是逻辑异或运算）

7.条件操作符

条件运算符是 JavaScript 支持的一种特殊的运算符。

“？：”又被称作“三目运算符”、“三元运算符”，类似于 if...else...条件语句。

语法：

条件 ？表达式 1：表达式 2;

说明：如果“条件”为 true，则表达式的值使用“表达式 1”的值；如果“条件”为 false，则表达式的值使用“表达式 2”的值。

```
var result;
var sex = true;
if(sex){
  result = '男';
} else {
  result = '女';
}
```

```
var sex = true;
var result = sex ? '男' : '女';
```

8.逗号操作符

9.对象操作符

New	Var cat=new Object();

delete	Var cat={name:'kitty',age:2};
	Alert(cat.name);//'kitty'
	Delete cat.name;
	Alert(cat.name);//undefined

.	获取对象的值
	Var cat={name:'kitty',age:2};
	Alert(cat.name);//'kitty'
	Alert(cat['name']);//'kitty'

[]	

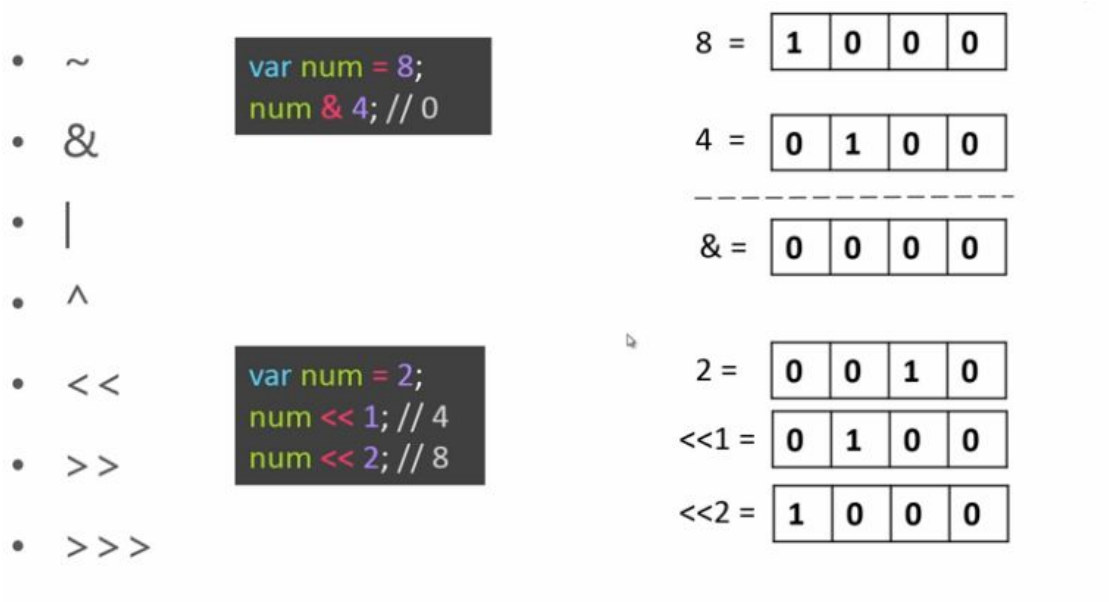
instanceof	判断某个变量是否是某个对象的实例：
	Var cat={name:'kitty',age:2};
	Alert(cat instanceof Object);//true
	Alert(cat instanceof Number);//false

in	判断某个属性是否在对象中：
	Var cat={name:'kitty',age:2};
	Alert('name' in cat);//true
	Alert('run' in cat);//false

10.位操作符

操作符	用法	描述
按位与	a & b	如果两个操作数对应位都是 1 的话则在该位返回 1。
按位或	a | b	如果两个操作数对应位都是 0 的话则在该位返回 0。
按位异或	a ^ b	如果两个操作数对应位只有一个 1 的话则在该位返回 1。
求反	~ a	反转操作数的每一位。

左移	a << b	将 a 的二进制形式左移 b 位。右面的空位补零。
算术右移	a >> b	将 a 的二进制形式右移 b 位。忽略被移出的位。
逻辑右移	a >>> b	将 a 的二进制形式右移 b 位。忽略被移出的位，左侧补入 0。



位操作符：

执行位操作时，操作符会将操作数看作一串二进制位(1 和 0)，而不是十进制、十六进制或八进制数字。例如，十进制的 9 就是二进制的 1001。位操作符在执行的时候会以二进制形式进行操作，但返回的值仍是标准的 JavaScript 数值。

从原理上讲，位逻辑操作符的工作流程是这样的：

将操作数转换为 32 位的整型数值并用二进制表示。

第一操作数的每一位与第二操作数的对应位配对：第一位对第一位，第二位对第二位，以此类推。

对每一对位应用操作符，最终结果按位组合起来。

例如，9 的二进制表示为 1001，15 的二进制表示为 1111。所以如果对这两个数应用位逻辑操作符，结果应该像下面这样：

15 & 9 结果为 9 (1111 & 1001 = 1001)

15 | 9 为 15 (1111 | 1001 = 1111)

15 ^ 9 为 6 (1111 ^ 1001 = 0110)

移位操作符

移位操作符需要两个操作数：第一个是要进行移位的数值，第二个指定要对第一个数移位的数目。移位的方向由使用的操作符决定。

移位操作符将把两个操作符转换为 32 位整型数值，并返回与左操作数类型相同的结果。

<< (左移)

该操作符将把第一个操作数向左移若干位。移出的位将被忽略。右侧空位补零。

例如，9<<2 结果为 36，因为 1001 向左移两位变成 100100，这是 36。

>> (算术右移)

该操作符将把第一个操作数享有移若干位。移出的位将被忽略。左侧的空位补上与原来最左面位相同的值。

例如，9>>2 结果为 2，因为 1001 右移两位变成 10，这是 2。反之，-9>>2 结果为 -3，因为要考虑到符号位。

>>> (逻辑右移)

该操作符将把第一个操作数享有移若干位。移出的位将被忽略。左侧的空位补零。

例如，19>>>2 结果为 4，因为 10011 右移两位变成 100，这是 4。对于非负数，算术右移和逻辑右移结果相同。

11.操作符优先级

下表列出了 JavaScript 运算符，并按优先级顺序从高到低排列。具有相同优先级的运算符按从左至右的顺序计算。

运算符	说明
. [ ] ( )	字段访问、数组索引、函数调用和表达式分组
++ -- - ~ ! delete new typeof void	一元运算符、返回数据类型、对象创建、未定义的值
* / %	相乘、相除、求余数
+ - +	相加、相减、字符串串联
<< >> >>>	移位
< <= > >= instanceof	小于、小于或等于、大于、大于或等于、是否为特定类的实例
== != === !==	相等、不相等、全等，不全等
&	按位“与”
^	按位“异或”
|	按位“或”

&&	逻辑“与”
||	逻辑“或”
?:	条件运算
= OP=	赋值、赋值运算（如 += 和 &=）
,	多个计算

圆括号用于改变由运算符优先级确定的计算顺序。这就是说，先计算完圆括号内的表达式，然后再将它的值用于表达式的其余部分。

```
var num =  5 + 4  * 3;
```

```
4 + 0 || 3; //4
!false && [];//[]
4 > 3 ? 5 : 7+10;//5
4 == '4'  && 3;//3
```

(二)表达式

表达式是一个语句的集合，计算结果是个单一值。

在 JavaScript 中，常见的表达式有 4 种：

- （1）赋值表达式；
- （2）算术表达式；
- （3）布尔表达式；
- （4）字符串表达式；

赋值表达式

在 JavaScript 种，赋值表达式的语法格式一般如下：

变量 赋值运算符 表达式;

赋值表达式在计算过程中是按照自右而左结合的。其中有简单的赋值表达式，如 n=1；也有定义变量时，给变量赋初始值的赋值表达式，如 var str="我爱 Javascript";还有使用比较复杂的赋值运算符连接的赋值表达式，如 n+=6。

算术表达式

算术表达式就是用算术运算符连接的 JavaScript 语句。如 a+b+c、20-12、m*n、m/n、sum%3 等，都是合法的算术运算符的表达式。

算术运算符的两边必须都是数值，若在“+”运算中存在字符或字符串，则该表达式将是字符串表达式.因为 JavaScript 会自动将数值型数据转换成字符串型数据。

布尔表达式

布尔表达式一般用来判断某个条件或者表达式是否成立，其结果只能为 true 或 false。

字符串表达式

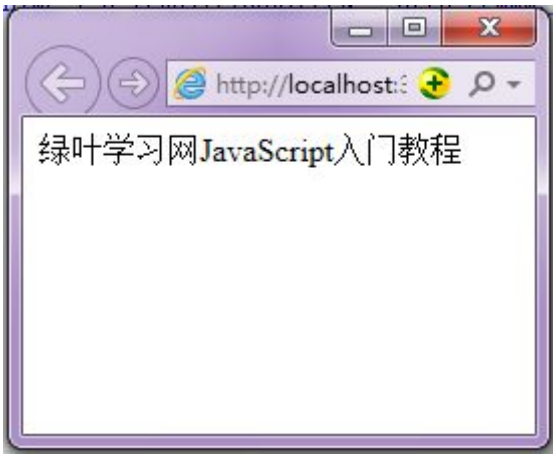
字符串表达式指的就是操作字符串的语句。

“数字+字符串”返回值一定是字符串，因为 JavaScript 会自动将数值型数据转换成字符串型数据。

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
    <script type="text/javascript">
        var a="绿叶学习网";
        var b="JavaScript";
        var c="入门教程";
        var str = a+b+c;
```

```
        document.write(str);
    </script>
</head>
<body>
</body>
</html>
```

在浏览器预览效果如下：



参考资源：

1. JavaScript 中的相等性判断
https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Equality_comparisons_and_sameness

2. 运算符优先级：

https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

3. JS 操作符、运算符汇总表格：

http://www.jb51.net/article/31518.htm

4. JS 运算符优先级：

https://technet.microsoft.com/zh-cn/library/z3ks45k7