# JS基础

## 1 JS简介

### 1.1 JavaScript与ECMAScript

ECMAScript是一种由Ecma国际前身为欧洲计算机制造商协会,英文名称是European Computer Manufacturers Association，制定的标准。

JavaScript是由公司开发而成的，公司开发而成的一定是有一些问题，不便于其他的公司拓展和使用。所以欧洲的这个ECMA的组织，牵头制定JavaScript的标准，取名为ECMAScript。

**简单来说ECMAScript不是一门语言，而是一个标准。符合这个标准的比较常见的有：JavaScript、Action Script（Flash中用的语言）。**就是说，你JavaScript学完了，Flash中的程序也会写了。

ECMAScript在2015年6月，发布了ECMAScript 6版本，语言的能力更强。但是，浏览器的厂商不能那么快的去追上这个标准。这些新的特性，我们就业班的深入，也会给大家介绍。

### 1.2 网页规范（前端标准）

结构 html

表现 css

行为 js(页面中的动态效果)

网页设计原则：结构、样式、行为---分离！

### 1.3 JS定义

JS是一门运行在客户端的解释型脚本编程语言。

特点：

1-解释执行 源代码无需经过编译，一行一行解析执行

2-基于对象 封装、继承、多态

构成：

JS = ECMAScript + DOM + BOM + 高级

ECMAScript（前身为欧洲计算机制造商协会）

JavaScript的语法规范

DOM（Document Object Model 的简称）

JavaScript操作网页上元素的API

BOM（Browser Object Model 的简称）

JavaScript操作浏览器部分功能的API

### 1.4 名词解释

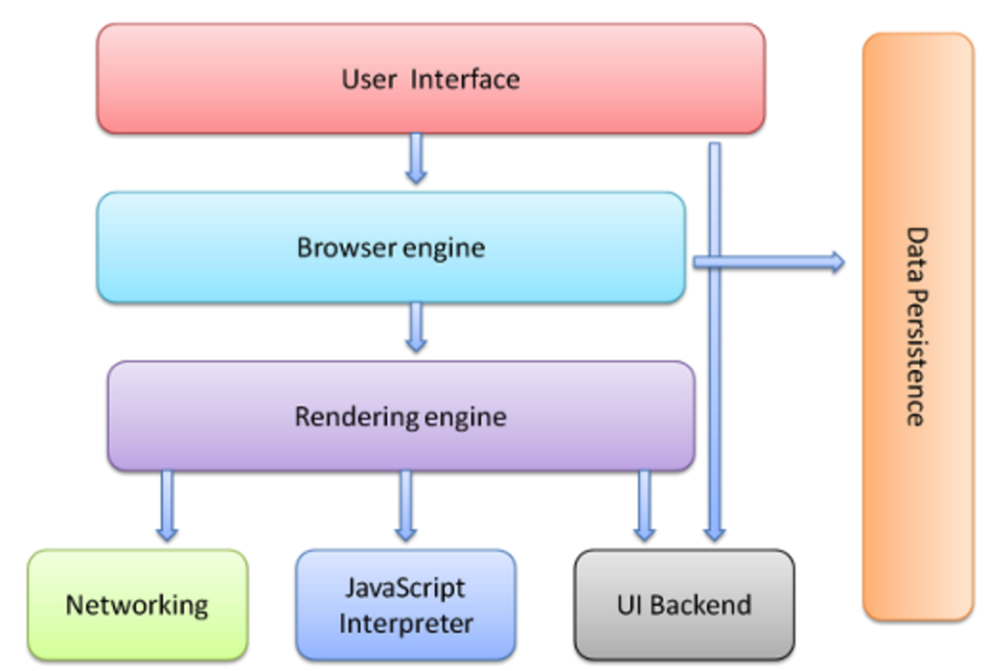
解释执行： 源代码无需经过编译，一行一行解析执行

编译执行： 源码编译为可执行文件，执行该文件

脚本语言： 弥补编译语言的不足而存在的，作为补充语言，不用编译。

弱类型语言： 简单理解定义一个变量，可以有多种数据类型。（var temp）

### 1.5 浏览器工作原理



### 1.6 JS书写位置

1-内嵌式：

|  |
| --- |
| <**head**>  <**meta charset="UTF-8"**>  <**script type="text/javascript"**>  </**script**> </**head**> |

2-外联式：在HTML文档中引入外部新建的JS文

|  |
| --- |
| <**script type="text/javascript" src="1.js"**> </**script**> |

此外还有内嵌式，极度不推荐：<button onclick=”fn()”>登录</button>

注意：推荐使用外联式，可以将js文件合并到一个文件中。如果使用内嵌式， 建议JS代码写在body标签后，以确保JS执行的正确性，

a链接的新写法：

<a href=”javascript:;”></a> 点击不执行的意思。

<a href=”javascript:void(0);”></a>

一般在tab栏切换中用到。

### 1.7 JS显示消息

1-使用注释： // /\* \*/

2-弹窗显示： alert(“ “);

3-网页显示： document.write( );

4-控制台显示： console.log( ); console.warn(); console.error();

5-判断消息： confirm(“你确定退出吗?YES/No!”);

6-接收用户消息： prompt(“请输入你的姓名”);

## 2 变量

### 2.1 变量的定义

变量就是用来存储数据的容器。

变量的定义与赋值：var i = 3;

多个变量定义赋值：var a = 3, b = 4;

注意：如果给同一个变量赋值，后面的值会覆盖之前赋的值。

使用逗号运算符可以将多个变量同时定义：var n1,n2;

### 2.2 变量命名规范

驼峰命名规则：getElementById/matherAndFather/aaaOrBbbAndCcc

遵从规则：

1.变量命名必须以字母或是下标符号”\_”或者”$”为开头。

2.变量名长度不能超过255个字符。

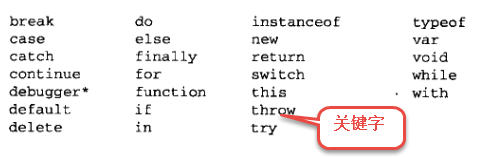
3.变量名中不允许使用空格，首个字不能为数字。

4.不用使用脚本语言中保留的关键字及保留符号作为变量名。

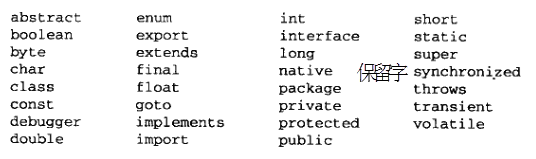
5.变量名区分大小写。(javascript是区分大小写的语言)

6.汉语可以作为变量名。但是不建议使用！！！（low）

关键字：



保留字：



## 3 数据类型

### 3.1 数据类型划分

一个变量的数据类型，可以确定该变量在内存中的存储方式。JS是一种弱类型语言，变量是什么值，就是什么数据类型。

根据数据类型的作用可以划分为两大类：简单数据类型、复杂数据类型

简单数据类型（值类型）

四种： 字符串 数字 布尔 未定义 空

String Number Boolean undefined null

复杂数据类型（引用型）

Object、function、Array、Date、RegExp、Error.......

注意：如果一个变量赋值为null ，那么该变量的数据类型为object

### 3.2 字面量

固定的值，让你从“字面上”理解其含义。

数值字面量

var age = 18; // 数值字面量，18为字面值

### 3.3 数据类型查看

三种查看写法：typeof() typeof name 或者 typeof(name)

### 3.4 Number类型

1、数字类型Number

变量赋的值是数字。数字可以是：小数，负数，正数，整数。

2、Number的范围

由于内存的限制，ECMAScript 并不能保存世界上所有的数值。

能表示的最大值是±1.7976931348623157乘以10的308次方；

能表示的最小值是±5 乘以10的-324次方。

获取方式：

最大值：Number.MAX\_VALUE，这个值为： 1.7976931348623157e+308

最小值：Number.MIN\_VALUE，这个值为： 5e-324

无穷大：Infinity

无穷小：-Infinity

3、Number的表示方式

|  |
| --- |
| **var *n1*** = 11; *//十进制* **var *n2*** = 012; *//八进制： 0开头，0-7之间* **var *n3*** =0xacf; *//十六进制：0x开头，0-F之间* |

进制计算

二进制转换十进制 110 ：0\*20 + 1\*21 + 1\*22 = 6

1. Nan

a) NaN 非数值（Not a Number的简写）

console.log(“abc”/18); //结果是NaN

Undefined和任何数值计算为NaN;

NaN 与任何值都不相等，包括 NaN 本身

b) isNaN() :任何不能被转换为数值的值都会导致这个函数返回 true

（isNaN译为是否符合一个标准，即不是一个数字的标准，如果符合了那么就不是一个数字，不符合就是一个数字）

isNaN(NaN);// true

isNaN(“blue”); // true

isNaN(123); // false

### 3.4 String类型

字符串类型（string）：赋的值是使用双引号或者单引号包含起来

转义即无法输出的字符，先输出/，在输出字符，如下所示

\\ 转一个\

\’ 转单引号

\” 转双引号

\r 回车

\n 换行

注意：用引号时，可单可双，唯独不可一单一双。可用.length看有几个字符。字符串在内存中不会立刻消失，只能二次赋值，原有的字符在一定时间内被垃圾回收器回收。

### 3.5 Boolean类型

Boolean类型有两个字面量：true和false，只能用小写。

true、除0数字、“something”、Object(任何对象)为true

false、0 、“”、undefined 、null为false

### 3.6 undefined类型

undefined类型：变量没有被赋值，默认值是undefined

如果定义一个变量，没有给该变量赋值，那么该变量的默认值为undefined，同时该变量的数据类型为undefined类型。我们称之为：变量未初始化。

null和undefined有最大的相似性。看看null == undefined的结果(true)也就更加能说明这点。但是null ===undefined的结果(false)。

和数字运算时，

10 + null 结果为：10；

10 + undefined 结果为：NaN。

任何数据类型和undefined运算都是NaN;

任何值和null运算，null可看做0运算。

### 3.7 特别提示

Boolean类型中：true数值为1；false为0；

null的数值类型为0；

undefined无数值类型或者为NaN;

## 4 数据类型转换

### 4.1 数据类型判断

typeof(变量) ==> 获取到当前变量的数据类型

|  |
| --- |
| **var *n*** = **true**; alert(typeof (***n***)); |

### 4.2 隐式类型转换

☞变量在参与运算的过程中，实现的数据类型的改变

### 4.3 强制类型转换

4.3.1 数字转字符串

使用String() 或者 变量名.toString()

|  |
| --- |
| **var *n1*** = 32; **var *n2*** = String(***n1***); **var *n3*** = ***n1***.toString(); **console**.log(typeof (***n1***)); **console**.log(typeof (***n2***)); **console**.log(typeof (***n3***)); |

注意：Null和undefined无toString方法。

4.3.2 字符串转数字

☞Number() 转换：

◆如果变量的值属于非数字的字符串，转换会失败，结果为NaN;

◆如果变量的值为数字的字符串，通过该方法可以将原来的数据类型转

换为数字类型，并且保留原来值的格式

☞parsInt()转换：

◆非数字的字符串通过该方法转后的结果为NaN

◆如果一个变量的值为前面为数字那么通过该方法可以转换为数字类

◆通过该方法只会保留数据的整数部分，没有进行四舍五入运算

☞parsInt()转换：将字符串转化为数字

◆如果变量的值为非数字的字符串，那么转换后的结果为NaN

◆如果变量的值为以数字开头的字符串，那么可以转换成功

◆如果变量的值为数字的字符串，通过该方法可以保留原来数字的格式。

parseInt(19.11); 值为19

parseInt(19.99); 值为19

parseInt(“25px”); 值为25

parseInt(值，进制); 括号内的值是\*\*进制的形式，输出结果十进制。

parseInt(10,2); 值为2

案例：

var a = “15.15abc” , b = “10.15”, c = “10.0abc”;

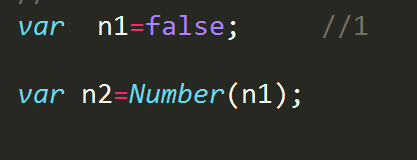
alert(parseInt(a) + Number(b) + parseFloat(c));

结果为：35.15

4.3.3 转换布尔

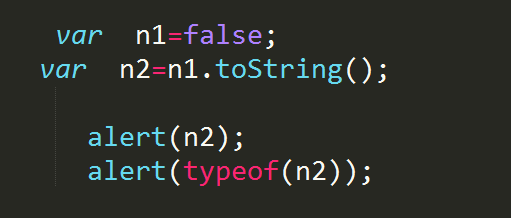
☞使用 Boolean() 可以实现强制类型转换

☞将布尔类型转化为数字类型通过Number方法



任何数据类型都可以转换成boolean类型

4.3.4 布尔转换字符串



4.3.5 paseInt()与parseFloat()

空字符串parseInt()和parseFloat()返回NaN，Number("")返回0

parseInt()：取整，如果变量首字符为字母结果为NaN。

123 = parseInt(“123.123aaaa”);

parseFloat()：取浮点数（没有小数取整），如果变量首字符为字母结果为NaN。

123.123 = parseFloat(“123.123aaaa”);

## 5 操作符

1. 算数运算符（+—\*/...）
   1. 一元运算符：正号、负号、++、--、平方等一个变量就能运算
   2. 二元运算符：+-\*/%等两个变量才能运算
   3. 三元运算符： 值1？值2：值3；
2. 逻辑运算符（ ||&& ! ）（或且非）
3. 比较运算符（<、>、==、>=...）
4. 赋值运算符（=、+=、-=、\*=、/=、%=）

### 5.1 比较运算符

> < >= <= !=

通过比较运算符，得出的结果为布尔类型的结果（true(真),false（假））

注意：

☞ “=” 赋值运算 （不能用来判断是否相等）

☞ “==” 等于 （判断内容是否相等，不考虑数据类型）

☞ “===” 全等于 （判断是否相等，考虑全部）

☞ “!=” 不等于 （判断内容是否不相等）

☞ “!==” 不全等于 （判断内容和数据类型是否不相等）

### 5.2 算术运算符

+ (加运算)

◆如果两个数字类型的变量相加，结果为数字。

◆如果是两个非数字类型的变量相加，加号只是连接作用

- （减运算）

◆如果两个数字类型的变量相减，最后的结果为数字。

◆如果两个数字类型的字符串相减，最后的结果为数字

例如： Var n1=”123”; var n2=”456”;

◆如果和非数字的字符串相减结尾为NaN

NaN： not a numebr

\*（乘运算）

/ （除运算）

◆如果一个数字除以数字0 ，得到的结果为infinity (无穷大)

% （取余 获取余数）

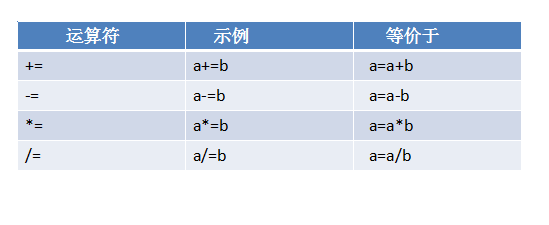
() 优先级（有小括号先计算小括号里面的值）

NaN： 不是一个数字 属于number数据类型

isNaN:

Infinity： 无穷大数字 属于number数据类型

### 5.3 带赋值的运算符



### 5.4 逻辑运算符

☞ || 或运算

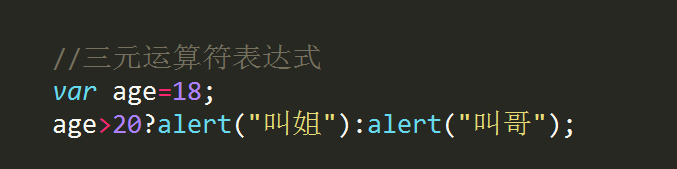
☞ && 且运算（条件同时满足）

☞ ! 非运算 （获取运算结果的反面）

注意： 使用逻辑运算符，必须连接的是布尔类型的结果

### 5.5 三元运算符

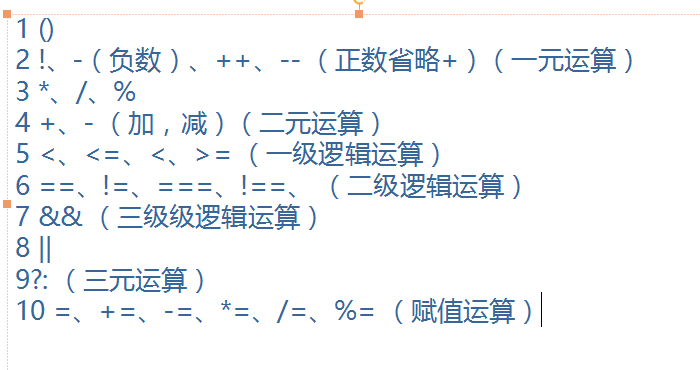
语法：条件表达式? 条件满足执行代码一 : 条件不满足执行冒号后面的代码



### 5.6 优先级

1 ()

2 !、-（负数）、++、-- （正数省略+）（一元运算）

3 \*、/、%

4 +、- （加，减）（二元运算）

5 <、<=、<、>= （一级逻辑运算）

6 ==、!=、===、!==、 （二级逻辑运算）

7 && （三级级逻辑运算）

8 ||

9?: （三元运算）

10 =、+=、-=、\*=、/=、%= （赋值运算）

## 6 自增自减

以var i=1为例：

i++; i的值在原来的基础上加1

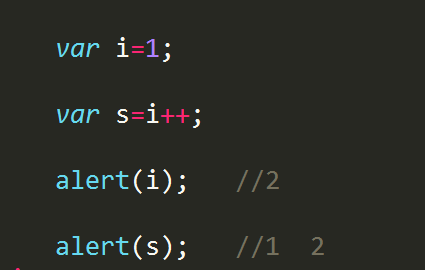
++i; i 的值在原来的基础上加1

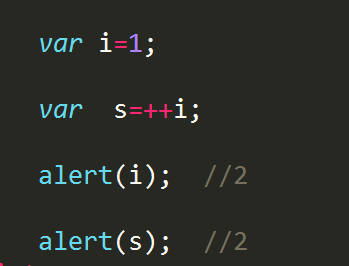
☞i++ 与++ i的区别

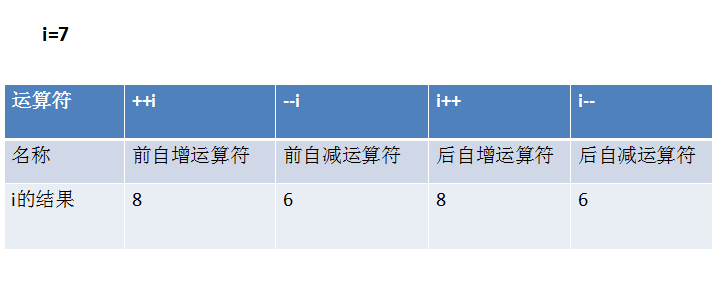
◆i++ 当i++参与运算中，先将i原来的值赋值给变量，然 后自己再加1 （先赋值后计算）

◆++i 当++i参与运算中，先将变量加1，然后将加1后的值

赋值给另外一个变量 （先计算后赋值）







## 7 流程控制

1. 顺序结构（程序正常执行流程从上往下，从左往右）（不用研究）

赋值运算时为从右往左执行，且左侧为属性或变量，不取值！！！

1. 选择结构
   1. If语句
   2. Switch语句
2. 循环结构
   1. for循环
   2. while循环
   3. do...while循环

### 7.1 if判断语句

1. If语句用法有三种
2. if(条件1){程序1}
3. if(条件1){程序1}else{程序2}
4. if(条件1){程序1}else if(条件2){程序2}...else{程序n}

if判断时会把（）内的值强行转换成boolean类型进行判断。

1. 三目运算（也叫三元运算）（目或者元代表几个表达式）

三目运算可以替代部分if...else...功能，运算简单，使用方便，代码清晰。

表达式1？值1：值2

注意： if。。。。else。。。 代码语句执行一次

### 7.2 嵌套判断

1. if(条件表达式){  
    条件为true
2. }else if (条件表达式) {  
    }else if(条件表示式){
3. }else {
4. }

注意： if条件判断中可以再嵌套其他条件判断语句。

### 7.3 switch判断语句

switch (值1) {

case value1:

程序1；

break; // break 关键字会导致代码执行流跳出 switch 语句

case value2:

程序2；

break;

default:

程序3；

}

注意：

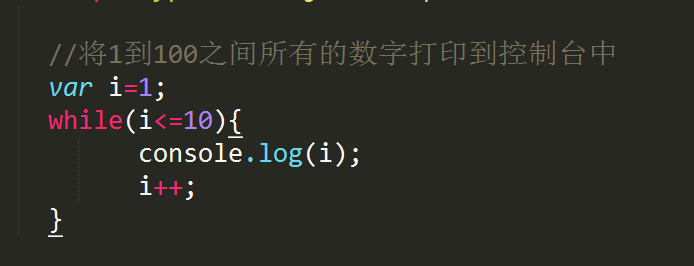
break可以省略，如果省略，代码会继续执行下一个case

switch 语句在比较值时使用的是全等操作符，因此不会发生类型转换

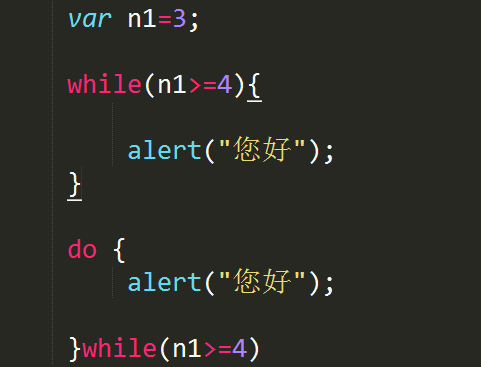
（例如，字符串 "10" 不等于数值 10）。

switch中的变量数据类型必须和case后面值的数据类型一致

### 7.4 while循环



### 7.5 do while循环



do while 循环执行顺序： 代码首先执行do中的代码语句.然后判断条件是否满足。如果条件满足，那么继续执行do中的代码语句。否则不再执行。

注意：如果条件不满足的情况下，do while循环会比whlie循环多执行一次。

### 7.6 for循环

☞语法结构

for(变量的初始化;条件表达式; 变量自增){

程序代码

}

☞执行顺序

◆变量初始化，然后判断条件是否成立（true）

◆如果条件成立，那么执行for中的代码语句，然后执行变量自增

◆变量自增后，然后继续判断条件是否成立如果成立继续执行代码，如

果不成立那么循环体代码不再执行。

### 7.7 break与continue

☞在循环中使用了break，代码立马结束本循环（循环不再执行）

☞continue语句以后的代码不再执行，当程序遇到continue会立马结束本次循环，进入到一下次循环。

