目录

[第一天 2](#_Toc471662118)

[1：定义 2](#_Toc471662119)

[2：JS历史 3](#_Toc471662120)

[3：JS组成 5](#_Toc471662121)

[4：script标签 6](#_Toc471662122)

[5：js初体验 7](#_Toc471662123)

[6：如何输出标签符号 7](#_Toc471662124)

[7：实体符号列表 8](#_Toc471662125)

[8：js语句 8](#_Toc471662126)

[9：js注释 9](#_Toc471662127)

[10：js变量 9](#_Toc471662128)

[11：变量常用命名法 9](#_Toc471662129)

[12：保留关键字 10](#_Toc471662130)

[13：数据类型 11](#_Toc471662131)

[14：运算符 11](#_Toc471662132)

[15：运算符的优先级 12](#_Toc471662133)

[应用： 13](#_Toc471662134)

[16：强制转换 13](#_Toc471662135)

[17：自动转换 16](#_Toc471662136)

[18：八进制和十六进制介绍 18](#_Toc471662137)

[19：代码规范 19](#_Toc471662138)

[应用： 19](#_Toc471662139)

[综合应用： 19](#_Toc471662140)

# 第一天

## 1：定义

**javascript是一种弱类型、动态类型、解释型的脚本语言。**

弱类型：类型检查不严格，偏向于容忍隐式类型转换。

强类型：类型检查严格，偏向于不容忍隐式类型转换。

动态类型：运行的时候执行类型检查。

静态类型：编译的时候就知道每个变量的类型。

解释型：程序不需要编译，程序在运行的时候才翻译成机器语言，每执行一次都要翻译一次，因此效率比较低，但是跨平台性好。

编译型：程序在执行之前需要一个专门的翻译过程，把程序编译为机器语言的文件，运行时直接使用编译的结果就行了。

标记语言：标记语言的存在就是用来被读取(浏览)的，而其本身是没有行为能力的，在标记语言里你会看到<和>这些尖括号，这是用来写出“层次”和”属性”的，换句话说，它是被动的。并不具备与访问者互动的能力。

编程语言：它是具有逻辑性和行为能力，这是主动的。说通俗一点，它是有思想的。

脚本语言：它介于标记语言和编程语言之间，脚本语言不需要编译，可以直接用，由解释器来负责解释。

## 2：JS历史

**1：Netsape发明了javascript。**

1994年，网景公司(Netscape)发布了Navigator浏览器0.9版。这是历史上第一个比较成熟的网络浏览器，轰动一时。但是这个版本的浏览器只能用来浏览，不具备与访问者互动的能力。比如，网页上有一栏用户名需要填写，浏览器无法判断用户是否真的填写了，只有让服务器去判断。如果没有填写就返回错误，要求用户重新填写。太浪费时间和服务器资源了。更何况在那个用调制解调器上网的年代。

**2：JS之父Brendan Eich（布兰登 · 艾奇）**

网景公司急需一种网页脚本语言，使得浏览器可以与网页互动。当时工程师Brendan Eich就是负责开发这种新语言。



**3：为什么叫JavaScript**

1995年Sun公司将Oak语言改名为Java，正式向市场推出。Sun公司大肆宣传，允诺这种语言可以”一次编译，到处运行”。很有可能成为未来市场的主宰。

网景公司动了心，决定与Sun公司结成联盟。它不仅允许JAVA程序以applet(小程序)的形式，直接在浏览器中运行。甚至还考虑直接将Java作为脚本语言嵌入网页，只是因为这样使HTML网页过于复杂，后来才不得不放弃。

总之，因为Sun公司完全介入网页脚本语言的决策。Js后来就是网景公司和Sun两家公司一起携手推向市场的，这种语言被命名为java+script不是偶然的。

**4：JS和JAVA语言没有任何关系**

JS之父并不喜欢JAVA，为了应付公司安排的任务，他只用了十天就把JS设计出来了。因为设计时间太短，语言的一些细节考虑的并不太严谨。

设计思路：

1：借鉴C语言的基本语法。

2：借鉴JAVA语言的数据类型和内存管理。

3：借鉴Scheme语言，将函数提升到”第一等公民的地位”。

4：借鉴Self语言，使用基于原型(prototype)的继承机制。

所以，JS实际是两种语言风格的混合产物：(简化的)函数式编程+(简化的)面向对象编程。

**5：JS标准化---ECMAScript**

因为javascript1.0的成功，netscape在navigator 3.0中发布1.1版。此时微软决定进军浏览器，在IE3.0种搭载了javascript的克隆版，叫JScript。在微软进来以后，有三种不同的javascript版本存在，即navigator3.0的javascript、IE的JScript、以及Cenvi中的ScriptEase。

由于javascript没有一个标准来统一其语法和特性，3种不同版本的恰恰突出了这个问题，这个语言的标准化显然势在必行。

1997年，javascript1.1作为一个草案提交给欧洲计算机制造协会(ECMA)。最后定义为ECMAScript的全新脚本语言。从此，浏览器开始努力将ECMAScript作为Javascript的基础。

## 3：JS组成

js组成：ECMAScript + BOM + DOM

尽管ECMAScript是一个重要标准，但它并不是JS的唯一部分。

一个完整的JS实现是由3个部分组成的。

ECMAScript(核心) + 浏览器对象模型(BOM)+ 文档对象模型(DOM)

ECMAScript描述了该语言的语法和基本对象。

BOM描述了与浏览器进行交互的方法和接口。

DOM描述了处理网页内容的方法和接口。

## 4：script标签

<script>标签用于定义客户端脚本。它既可以包含脚本语句，也可以通过src属性指定外部脚本文件。

属性：

language：用来指定<script>标签中的脚本类型，即javascript。**已废弃**，大多数浏览器已经忽略它了，所以不要在使用。

type：它也是用来指定<script>标签中的脚本类型，即text/javascript。它也是type的默认值，所以可以忽略指定。

src：指定外部的脚本文件。**如果指定该属性，script标签包含的JS脚本不会执行。**

**不可以使用单标签，即<script type=“text/javascript”/>。**

**script标签可以放在页面中的任何位置，但是我们一般放在head和body中。**

## 5：js初体验

document.write：输出内容到页面，可同时输出多段文本。

eg：document.write(‘我叫’, ‘thomas!’);

document.writeln：输出内容到页面，但是会在表达式后面写一个换行符。

eg：document.writeln(‘我叫’, ‘thomas’);

writeln() = write() + write(‘\n’);

**为什么换行没起作用？**

在HTML的解析中，只会把<br/>解析成换行，\n会被解析成空白符，空白符就被解析成一个空格。但是在pre、textarea标签内，HTML不会解析空白符。

**如何让输出内容换行？**

1：输出带有<pre>标签的内容。

2：输出带有<br/>标签的内容。

**注：如果页面加载完毕后再执行document.write，这个HTML页面将被覆盖。**

## 6：如何输出标签符号

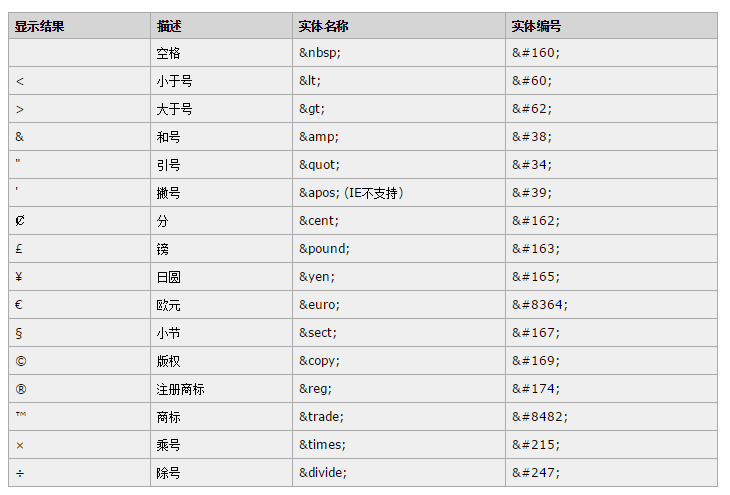
为了正确的显示预留字符，我们必须在HTML源代码中使用字符实体。如果直接使用<或者>，浏览器会误认为标签。

HTML实体格式：&entity\_name; 或者 &#entity\_number;

**实体名称对大小写敏感。**

使用实体名称易于记忆，但是浏览器也许并不支持所有的实体名称(对实体数字的支持就很好)。

## 7：实体符号列表

****

## 8：js语句

JS语句就是向浏览器发出的命令。该语句告诉浏览器应该做什么。

分号（英文）用于分割JS语句，所以每条语句都应以分号结束。使用分号我们可以在一行内编写多条语句。但是在JS中分号来结束语句是可选的。

JS代码是JS语句的序列。浏览器会按照编写的顺序来执行每条语句。

JS代码块：JS语句通过代码块的形式进行组合。块由左花括号开始，右花括号结束。它的作用就是使语句一起执行。

**JS语句严格区分大小写。**

## 9：js注释

注释可用于提高代码的可读性。Javascript不会执行注释，用户也不会看到注释，注释只是方便开发者更好的理解JS代码。

单行注释：以//开头。

多行注释：以/\*开头，以\*/结尾。

文档注释：以/\*\*开头，以\*/结尾。

## 10：js变量

变量是存储信息的容器，用var关键词来声明(创建)变量。

变量命名规则：以字母、数字、下划线和$组成，但是不能以数字开头。

JS语句和JS变量都是严格区分大小写。**不能用拼音来命名。**

我们可以在一条语句中声明多个变量，该语句以var开头，并使用英文逗号分隔。

我们可以通过=给变量赋值，然后通过alert查看变量内容。

## 11：变量常用命名法

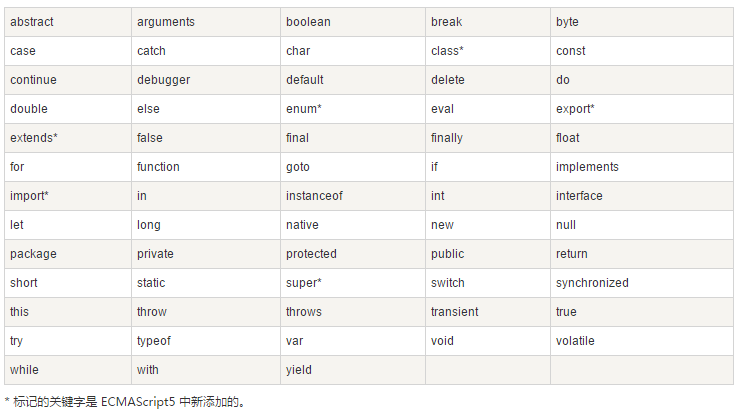
驼峰命名法（Camel）：第一个单词的首字母小写，后面的单词首字母大写。

帕斯卡命名法（Pascal）：所有单词的首字母大写。

匈牙利命名法（Hungarian）：在变量名最前面添加相应小写字母来标识变量的类型，后面的单词首字母大写。



## 12：保留关键字



## 13：数据类型

原始数据类型：字符串类型（string）、数值类型（number）、布尔类型（boolean）、undefined、null。

引用数据类型：对象（object），函数（function）。

字符串类型：必须要由成对的单引号或双引号包起来。内容可以是任意文本，只要不包含包围字符串的引号就可以。**如果一定要包含，则用反斜杠转义。**

数值类型：可以带小数点，也可以不带小数点。

布尔类型：只有两个值：true，false。

undefined：表示声明的变量没有值。

null：用来表示尚未存在的对象**。**

**对象、函数后面再详细讲解。**

**我们可以通过typeof来查看变量的类型。**

## 14：运算符

算术运算符：+、-、\*、/、%(取余)、++（递增）、--(递减)

赋值运算符：=、+=、-=、\*=、/=、%=

**+运算符：可用于把字符串变量连接起来。**

**当用/运算符的时候，Math.round的用法(四舍五入)。**

**parseInt：将字符串转换成整数。**

**parseFloat：将字符串转换成浮点数，也就是保留小数点。**

**Number必须为数值，parseInt必须是第一位是数字。**

关系(比较)运算符：>、<、>=、<=、==、===、!=、!==

**===与==的区别：**

对于string和number等基本类型，不同类型之间比较，==比较会将变量隐式转换成同一类型的值进行比较。而===如果类型不同，其结果就是不等。

如果是array和object等引用类型，==和===没有区别，如果两个的变量是同一个对象就为true，否则为false。

基本类型和引用类型进行比较，==会将引用类型转换成基本类型，再进行值比较。而===因为类型不同，结果为false。

逻辑运算符：&&、||、!

三目运算符：? :

## 15：运算符的优先级

. [] ()：字段访问、数组下标、函数调用以及表达式分组

++、--、!、delete、new、typeof

\*、/、%

+、-、+ 加、减、字符串连接

关系运算符、instanceof（判断某个对象是否是某个类的一个实例）

逻辑运算符

赋值运算符

**()作用：可以表达式分组、改变运算符的优先级、函数调用。**

**new运算符：用来创建对象。**

## 应用：

1：为抵抗洪水，战士连续作战89小时，编程计算共多少天零多少小时？

2：小明要到美国旅游，可是那里的温度是以华氏度为单位记录的。他需要一个程序将华氏温度（80度）转换为摄氏度，并以华氏度和摄氏度为单位分别显示该温度。  
提示：摄氏度与华氏度的转换公式为：摄氏度 = 5/9.0\*(华氏度-32)保留3位小数。

## 16：强制转换

强制转换主要指使用Number、String和Boolean三个构造函数，手动将各种类型的值，转换成数字、字符串或者布尔值。

1：Number强制转换

**参数为原始类型值的转换规则：**

原始类型的值主要是字符串、布尔值、undefined和null，它们都能被Number转成数值或NaN。

NaN：not a number，当数学计算得不到数字结果时，该值就是NaN。

isNaN：判断变量是否为NaN。

// 数值：转换后还是原来的值

Number(324) // 324

// 字符串：如果可以被解析为数值，则转换为相应的数值

Number('324') // 324

// 字符串：如果不可以被解析为数值，返回NaN

Number('324abc') // NaN

// 空字符串转为0

Number('') // 0

// 布尔值：true 转成1，false 转成0

Number(true) // 1

Number(false) // 0

// undefined：转成 NaN

Number(undefined) // NaN

// null：转成0

Number(null) //0

Number函数将字符串转为数值，要比parseInt函数严格很多。基本上，只要有一个字符无法转成数值，整个字符串就会被转为NaN。

**参数为对象的转换规则：**

简单的规则是，Number方法的参数是对象时，将返回NaN。

Number({a: 1}) // NaN

Number([1, 2, 3]) // NaN

**实际上，Number背后的真正规则复杂得多，内部处理步骤如下：**

1：调用对象自身的valueOf方法。如果返回原始类型的值，则直接对该值使用Number函数，不再进行后续步骤。

2：如果valueOf方法返回的还是对象，则改为调用对象自身的toString方法。如果返回原始类型的值，则对该值使用Number函数，不再进行后续步骤。

3：如果toString方法返回的是对象，就报错。

2：String强制转换

**参数为原始类型值的转换规则：**

数值：转为相应的字符串。

字符串：转换后还是原来的值。

布尔值：true转为"true"，false转为"false"。

undefined：转为"undefined"。

null：转为"null"。

**参数为对象的转换规则：**

String方法的参数如果是对象，返回一个类型字符串；如果是数组，返回该数组的字符串形式。

**String内部处理步骤如下：**

1：先调用对象自身的toString方法。如果返回原始类型的值，则对该值使用String函数，不再进行以下步骤。

2：如果toString方法返回的是对象，再调用valueOf方法。如果返回原始类型的值，则对该值使用String函数，不再进行以下步骤。

3：如果valueOf方法返回的是对象，就报错。

3：Boolean强制转换

**参数为原始类型值的转换规则：**

转换规则相对简单：除了以下六个值的转换结果为false，其他的值全部为true。

undefined、null、-0、0或+0、NaN、''（空字符串）

**参数为对象的转换规则：**

所有对象（包括空对象）的转换结果都是true。

## 17：自动转换

自动转换的规则：预期什么类型的值，就调用该类型的转换函数。比如，某个位置预期为字符串，就调用String函数进行转换。如果该位置即可以是字符串，也可能是数值，那么默认转为数值。

由于自动转换具有不确定性，而且不易除错，建议在预期为布尔值、数值、字符串的地方，全部使用Boolean、Number和String函数进行显式转换。

**1：自动转换为布尔值**

当JavaScript遇到预期为布尔值的地方（比如if语句的条件部分），就会将非布尔值的参数自动转换为布尔值。系统内部会自动调用Boolean函数。

if ( !undefined && !null && !0 && !NaN && !'' ) { console.log('true'); } // true

// 写法一 expression ? true : false

// 写法二 !! expression

**2：自动转换为字符串**

当JavaScript遇到预期为字符串的地方，就会将非字符串的数据自动转为字符串。系统内部会自动调用String函数。字符串的自动转换，主要发生在加法运算时。当一个值为字符串，另一个值为非字符串，则后者转为字符串。

'5' + 1 // '51' '5' + true // "5true"

'5' + false // "5false" '5' + {} // "5[object Object]"

'5' + [] // "5" '5' + function (){} // "5function (){}"

'5' + undefined // "5undefined“

'5' + null // "5null"

**3：自动转换为数值**

当JavaScript遇到预期为数值的地方，就会将参数值自动转换为数值。系统内部会自动调用Number函数。

'5' - '2' // 3 '5' \* '2' // 10

true - 1 // 0 false - 1 // -1

'1' - 1 // 0 '5' \* [] // 0

false / '5' // 0 'abc' - 1 // NaN

除加法运算符有可能把运算子转为字符串，其他运算符都会把运算子自动转成数值。

+'abc' // NaN -'abc' // NaN +true // 1 -false // 0

特殊：

**null == undefined // true**

**null == false // false**

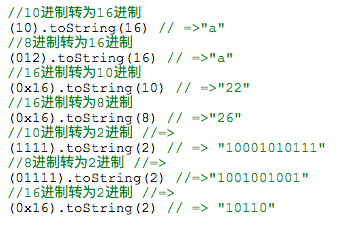
**undefined == false // false**

## 18：八进制和十六进制介绍

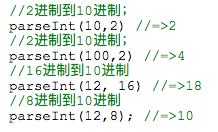
八进制：一种以8为基数的计数法。采用0，1，2，3，4，5，6，7八个数字，逢八进1。js以数字0开始表明该数字是八进制。

十六进制：计算机中数据的一种表示方法。它由0-9，A-F组成，字母不区分大小写。与10进制的对应关系是：0-9对应0-9；A-F对应10-15；js以0x开始表明该数字是十六进制。

js的进制转换，分为2进制，8进制，10进制，16进制之间的相互转换，我们直接利用对象.toString()即可实现：



如果要处理2进制到10进制，16进制到10进制，8进制到10进制， 需要用了parseInt这个方法：



**进制转换**：如果要实现进制之间的转换， 可以利用parseInt方法， 先转化为10进制， 然后再利用toString(参数)， 转化成不同的进制。

## 19：代码规范

1：代码缩进。

2：+、=等运算符两边的空格。

3：语句结束以分号结尾。

4：小括号的嵌套，要加空格。

5：双引号、单引号的嵌套。

## 应用：

1：计算两个文本框的和。

2：var k=0; alert(k++ + ++k +k +k++)。

3：掌握逻辑运算的意义。

4：掌握八进制、十六进制的写法。

5：掌握NaN的含义。

6：掌握Number函数的转换结果。

## 综合应用：

计算两个文本框的加减乘除。

要求：1) 使用parseInt方法类型转换。

2) 计算结果使用Math.round方法四舍五入。