# JavaScript基础入门

第1天课堂笔记

# 目录

[JavaScript基础入门 1](#_Toc32730)

[目录 2](#_Toc27575)

[一、JavaScript简介 4](#_Toc8692)

[1.1 JavaScript用途 4](#_Toc23668)

[1.2 JavaScript诞生 4](#_Toc30014)

[1.3 JavaScript从丑小鸭到金凤凰 5](#_Toc29233)

[1.4 JavaScript非常好学 6](#_Toc1237)

[1.5 学习方法 6](#_Toc21028)

[二、Hello World！ 7](#_Toc13689)

[2.1 JavaScript书写的位置 7](#_Toc22755)

[2.2 alert()语句 7](#_Toc14551)

[2.3 控制台 8](#_Toc25364)

[2.4 行文特性 9](#_Toc19458)

[2.5 注释 9](#_Toc18550)

[三、字面量 11](#_Toc8910)

[3.1 数字的字面量 11](#_Toc1114)

[3.2 字符串 14](#_Toc12011)

[四、变量 15](#_Toc7881)

[4.1 整体感知 15](#_Toc10481)

[4.2 变量必须先声明，才能使用 16](#_Toc17378)

[4.3 变量的赋值 17](#_Toc12993)

[4.4 变量声明的提升 18](#_Toc9024)

[4.5 不写var的情况 18](#_Toc26078)

[4.6 用逗号来隔开多个变量的定义 18](#_Toc16224)

[4.7 区分变量还是直接量 19](#_Toc30132)

[五、变量的类型 20](#_Toc21855)

[5.1 概述 20](#_Toc3295)

[5.2 typeof关键字 20](#_Toc1929)

[5.3 number类型 21](#_Toc1527)

[5.4 string类型 21](#_Toc2455)

[5.5 undefined类型 22](#_Toc25452)

[5.6 加号 22](#_Toc17283)

[六、变量类型的转换 24](#_Toc18282)

[5.1 string → number 24](#_Toc18527)

[5.2 number → string 26](#_Toc29681)

[七、数学运算符 27](#_Toc10394)

# 一、JavaScript简介

## 1.1 JavaScript用途

前端三层：

结构层 HTML 从语义的角度描述页面的结构

样式层 CSS 从审美的角度装饰页面

行为层 JavaScript 从交互的角度提升用户体验

JavaScript用来制作web页面交互效果，提升用户体验。

|  |  |  |  |
| --- | --- | --- | --- |
| 轮播图 | 选项卡TAB栏 | Ajax表单验证 | 炫酷HTML5页面 |

**JavaScript在今天，如果你要离开他，寸步难行，网页将无法正常使用。**

## 1.2 JavaScript诞生

在1995 年 Netscape（网景公司，我们要非常感谢网景公司，首先提出的交互的概念，首先提出的用户体验的概念，在老师的心中，Brendan Eich能够和乔布斯平起平坐），一位名为 Brendan Eich 的工程师创造了 JavaScript，随后在 1996 年初，JavaScript 首先被应用于 Netscape 2 浏览器上。最初的 JavaScript 名为 LiveScript（活力脚本），后来因为 Sun Microsystem 的 Java 语言的兴起和广泛使用，Netscape 出于宣传和推广的考虑，将它的名字从最初的 LiveScript 更改为 JavaScript——尽管两者之间并没有什么共同点。这便是之后混淆产生的根源。（**也就是说，JavaScript和Java没有任何关系，就是为了“榜大牌”的，和北大青鸟和北大的关系一样**）。

几个月后，Microsoft 随着 IE 3 推出了一个与之基本兼容的语言 JScript（注意微软的人生哲学，当它发现别人的东西很好的时候，就必须拧巴的推出自己的，然后自己的又被市场排斥，又开始兼容别人的）。又几个月后，Netscape 将 JavaScript 提交至 Ecma International（一个欧洲标准化组织）， ECMAScript 标准第一版便在 1997 年诞生了，随后在 1999 年以 ECMAScript 第三版的形式进行了更新，从那之后这个标准没有发生过大的改动。由于委员会在语言特性的讨论上发生分歧，ECMAScript 第四版尚未推出便被废除，但随后于 2009 年 12 月发布的 ECMAScript 第五版引入了第四版草案加入的许多特性。第六版标准已经于2015年六月发布。

**ECMAScript是JavaScript的标准**，现在我们学习的版本的ECMAScript4、5。

Brendan Eich (JavaScript之父)



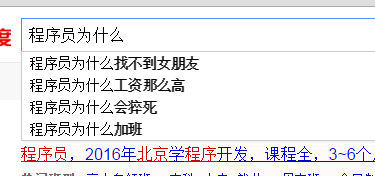
出任Mozilla CTO

## 1.3 JavaScript从丑小鸭到金凤凰

**2003年：**牛皮鲜，页面上漂浮的广告、弹窗广告；所以当时的浏览器就推出一个功能，禁用广告，实际上本质就是禁用JavaScript。页面上的特效，都特别俗，比如鼠标后面跟随的星星，然后工程师对JS的感觉就是不用学习，生搬硬套，大量的“效果宝盒”软件，一套就有各种特效了。没有人琢磨语言特性。

**2004年：**谷歌打开了Ajax这个潘多拉的盒子，从此JavaScript被人重视，很多人开始学习JS语言。当时问世了两本JS巨作《犀牛书》、《高级程序设计》。

google suggest



**2007年：**三层分离，iPhone发布，人们开始重视用户体验。大家发现了，JavaScript是web页面中制作交互效果唯一的语言，所以把JS的重视程度，提到了相当高的一个地位。招聘信息里面开始出现独立的“JS工程师”职位了，之前都是后台工程师捎带脚写写JS。

**2008年**：Chrome浏览器发布，V8引擎加快了JS的解析，之前的浏览器解析JS的时候卡顿卡顿的，动画效果是蹦蹦的。在Chrome里，它的引擎叫做V8，运行JS很流畅。

**2009年**：jQuery变得流行，解决了浏览器兼容问题，制作页面效果变得简单，越来越多的初学者愿意学习JavaScript。

**2010年**：Canvas画布技术得到众多浏览器支持，可以用Canvas替代Flash了，并且能制作小游戏，比如偷菜、停车小游戏。我们有这个课程，做Flappy Bird、俄罗斯方块、消消乐、接触物理抛物线引擎。

**2011年**：Node.js得到广泛应用，实际上就是把JavaScript运行在了服务器上，单线程非阻塞，能够让企业用最小的成本实现后台网站，比如之前4万的服务器都卡，现在用了node之后，4000的机器都很流畅。

**2012年**：HTML5+CSS3的流行，也带火了JavaScript。

**2013年**： hybrid app模式开始流行。就是做手机app的时候，老板们发现要雇佣三队人马，iOS、安卓、windows phone。花三份工资，并且产品还不好迭代。所以人们发明了用网页技术开发手机App的技术，叫做web app。hybrid app就是混合app，同时结合web技术和原生开发技术。省钱，好迭代。

**2015年**：ECMA6发布，叫做ECMA2015。重量级的改变，把语言的特性颠覆性的一个增强。

## 1.4 JavaScript非常好学

好学的点是什么：

JavaScript是有界面效果：不像C语言，黑底白字，很枯燥的。

JavaScript的语法来源于C和Java：有C和Java的经验同学好学很多。

JavaScript是弱变量类型的语言，动态数据类型语言。

JavaScript运行在宿主环境中，不关心内存，垃圾回收。

不好学的点是什么：

**兼容性问题**： IE8是个怪胎，很多东西都不一样，所以就要写兼容写法，不怕造轮子，多写几遍

**花式写法很多，抽象**：从简单入手，细细品味代码

**太多细节**： 为知笔记认真写自己的笔记，自己做实验;

老师的课程已经精挑细选了细节，并不盲目堆砌

## 1.5 学习方法

● 要多去“品”程序，多去思考内在逻辑，读懂每一行代码！

● JS机械重复性的劳动几乎为0，基本都是创造性的劳动。HTML、CSS都是重复的劳动，margin、padding挤来挤去。

● 永远不要背程序，每一个程序都必须自己会写，有一个每天的作业，重复老师的上课案例。

# 二、Hello World！

这事儿吧，挺有意思，就是学习任何的语言，我们都喜欢在屏幕上直接输出一点什么，当做最简单、最基本的案例。输出什么大家随意，但是很多人都习惯输出“hello world”，世界你好。感觉自己很有情怀的样子。

## 2.1 JavaScript书写的位置

|  |
| --- |
| 1. **<script type="text/javascript">** 2. alert("Hello World!"); 3. **</script>** |



JavaSript程序，要写在HTML页面中，**运行页面的时候，这个页面上的JS也就一起运行了**。也就是说，js的运行必须有宿主环境，最最常见的宿主环境，就是浏览器。

JS写在<script>标签对儿里面，type属性就是类型，text/javascript就是纯文本的javascript；**注意，这个type必须写准确，可以不写，但是要写，一定要写准确，否则失效。如果爱，请深爱。**

## 2.2 alert()语句

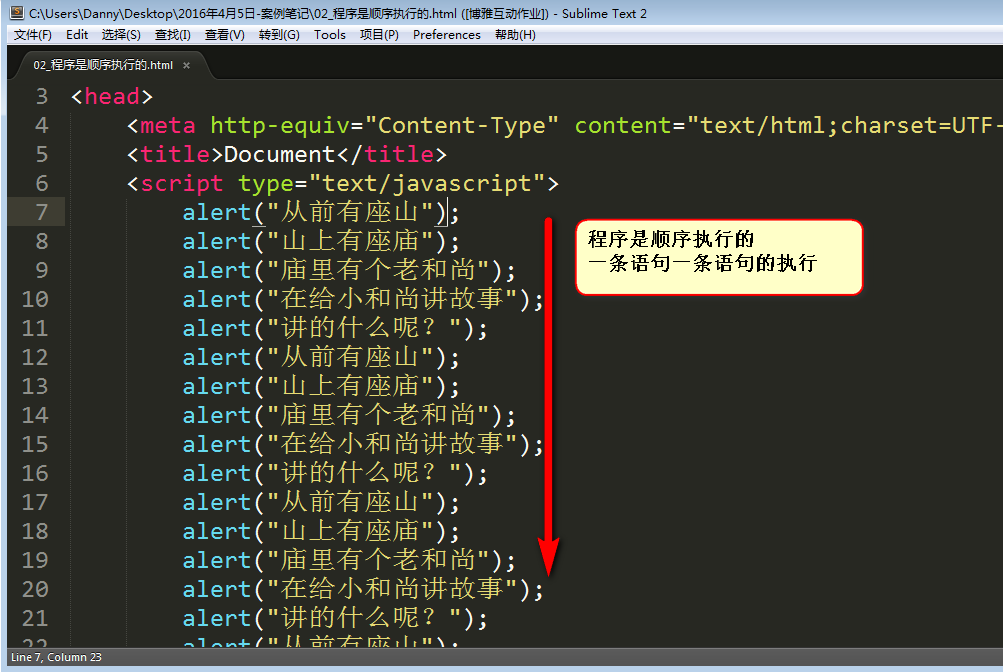
alert就是英语中“警报”的意思，能够让页面弹出警告框。

实际上这条语句，在运行一个内置函数，运行函数就要加圆括号。不要纠结这个东西，**学习所有的程序，都是从模仿开始的**。前期，不要去纠结语法的机理是什么，只需要进行模仿。你只需要知道，**你模仿了，你就能实现效果**。随着课程的深入，你将知道每条语句的确切机理。

|  |
| --- |
| 1. alert("你好啊！"); |

这是一条**语句**，**每条语句之后用;结尾**。注意，所有的符号都是英文的符号，不要使用中文！

**程序是顺序执行的**，任何程序都是这样。



语句都要遵循**语法**，alert的语法比较简单，就是后面的圆括号**。**

## 2.3 控制台

控制台是Chrome浏览器“检查”里面的功能，快捷键是F12。英文叫做console。

**程序的所有未捕获的错误，都会在控制台中输出**。控制台是调试程序的一个利器。

比如下面的语句用了中文的符号，有语法错误：

|  |
| --- |
| 1. alert(“你好”)； |



有些时候我们想在控制台中自己输出点什么，那么就要使用语句：

|  |
| --- |
| 1. console.log(); |

console就是控制台，log就是日志的意思，这里是动词，表示打印。

还是那句话，只需要模仿就行了，后面的课程你将知道console是一个内置对象，log是它的一个方法。

比如：

|  |
| --- |
| 1. console.log("你好啊"); |

养成打开控制台调试程序的习惯。

## 2.4 行文特性

**JavaScript语句和语句之间的换行、空格、缩进都不敏感。**

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("你好");alert("帅"); 3. alert("哈哈哈哈"); 4. </script> |

等价于：

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("你好"); 3. alert("帅"); 4. alert("哈哈哈哈"); 5. </script> |

语句后面的分号，不是必须写的，如果语句是一行一行写的，那么分号是没有必要的。但是压缩页面的时候，语句结尾的**分号**，非常重要。

**我们把页面做好之后，通常都会进行压缩，用软件把所有的空格、换行都去掉。此时，语句末尾的分号显得非常重要，如果去掉分号，将不能执行。**

不能执行：

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("你好")alert("帅")alert("哈哈哈哈") 3. </script> |

正确写法：

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("你好")**;**alert("帅")**;**alert("哈哈哈哈")**;** 3. </script> |

## 2.5 注释

给人看的东西，对读程序是一个提示作用。复习一下HTML的注释：

|  |
| --- |
| 1. <!--我是注释--> |

CSS注释：

|  |
| --- |
| 1. /\*我是注释\*/ |

JavaScript里面的注释：

单行注释：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **//我是一行注释** 3. **//再来一行** 4. alert("你好"); 5. </script> |

或者多行注释：

|  |
| --- |
| 1. **/\*** 2. 我是一行注释 3. 再来一行 4. 哈哈，我又写了一行 5. **\*/** |

注释的符号，不能嵌套，错误的示范：

|  |
| --- |
| 1. /\* 2. 我是一行注释 3. **/\*** 4. **\*/** 5. 再来一行 6. 哈哈，我又写了一行 7. \*/ |

sublime中注释的快捷键，就是ctrl+/

# 三、字面量

字面量：英语叫做literals，有些书上叫做直接量。看见什么，它就是什么。

我们先来学习数字的字面量，和字符串的字面量。剩余的字面量类型，我们日后遇见再介绍。

## 3.1 数字的字面量

数字的字面量，就是这个数字自己，并不需要任何的符号来界定这个数字。

JavaScript中，数字的字面量可以有三种进制：

10进制：普通的数字就是十进制

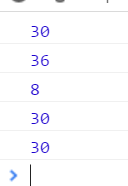
8进制：如果以0开头、或者以0o开头、或者以0O开头的都是八进制，八进制只能用0~7来表示

16进制：如果以0x开头的都是十六进制。

八进制举例：

|  |
| --- |
| 1. <script type="text/javascript"> 2. //以0开头，所以就是八进制；显示的时候会以十进制显示 3. //3\*8+6=30 4. console.log(**0**36); //30 5. console.log(**0**44); //36 6. console.log(**0**10); //8 7. console.log(**0o**36); //30 8. console.log(**0O**36); //30 9. </script> |

运行结果：



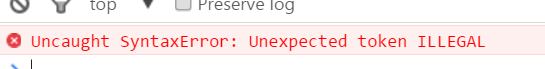
注意，八进制只能出现0~7这8中字符，如果表示不合法，那么JS将自动的认为你输入错了，从而用十进制进行显示：

|  |
| --- |
| 1. console.log(088); //以0开头，按理说是八进制，但是后面的数字错了，所以以十进制显示 |



但是以0o开头、0O开头的数字，如果后面写错了，控制台报错！

|  |
| --- |
| 1. console.log(**0o**88); |



再看一下16进制：

|  |
| --- |
| 1. console.log(**0x**ff); 2. console.log(**0x**2b); 3. console.log(**0x**11); |

运行结果：



如果后面有错误的写法，那么控制台报错：

|  |
| --- |
| 1. console.log(**0x**2m); |



总结一下，下面的输出结果都是15：

|  |
| --- |
| 1. console.log(15); 2. console.log(017); 3. console.log(0o17); 4. console.log(0O17); 5. console.log(0xf); |

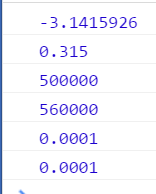
下面的输出结果都是负15：

|  |
| --- |
| 1. console.log(-15); 2. console.log(-017); 3. console.log(-0o17); 4. console.log(-0O17); 5. console.log(-0xf); |

小数的字面量也很简单，就是数学上的点。计算机世界中，小数称为“浮点数”。

**允许使用e来表示乘以10的几次幂**：

|  |
| --- |
| 1. console.log(-3.1415926); //-3.1415926 2. console.log(.315); //.315 如果整数位数是0，可以不写 3. console.log(5e5); //500000 4. console.log(5.6e5); //560000 5. console.log(1e-4); //0.0001 6. console.log(.1e-3); //0.0001 |



**只有十进制有小数的字面量，八进制、十六进制没有小数的字面量。**

最后学习两个特殊的字面量：

**Infinity 无穷大**

试图输出一个非常大的数字：

|  |
| --- |
| 1. console.log(3e45645645645645); |

控制台显示 表示无穷大



至于多大的数字能生成无穷大，不同浏览器不一样，不要管。

可以是负的无穷大：

|  |
| --- |
| 1. console.log(-3e45645645645645); |



我们可以直接在控制台输出无穷大，注意大小写：

|  |
| --- |
| 1. console.log(**Infinity**); |

NaN 英语全名叫做not a number，不是一个数。比较哲学的是，这个“不是一个数”是一个数字字面量。

|  |
| --- |
| 1. console.log(0/0); |



总结一下，数字字面量有整数字面量（十进制、16进制、八进制），浮点数字面量（要记住e），Infinity，NaN

## 3.2 字符串

字符串是一个术语，就是人类说的语句、词。

**字符串的字面量，必须用双引号、单引号包裹起来。字符串被限定在同种引号之间；也即，必须是成对单引号或成对双引号。**

|  |
| --- |
| 1. console.log(**"**今天天气很好**"**); |

|  |
| --- |
| 1. // 必须是同种引号，这是错误的： 2. console.log(**'**哈哈**"**); |

如果一个数字，用引号引起来，那么就是字符串了：

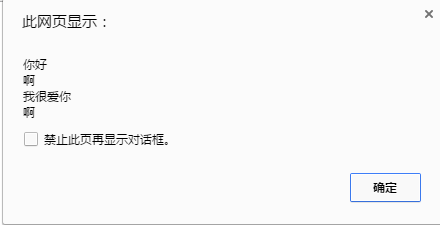
|  |
| --- |
| 1. console.log(“3”); |

转义字符，

\n 回车换行

\\ \

|  |
| --- |
| 1. alert("你好**\n**啊**\n**我很爱你**\n**啊"); |



# 四、变量

变量（Variables），和高中代数学习的x、y、z很像，它们不是字母，而是蕴含值的符号。

它和直接量不同，直接量5，就是数字5；直接量”你好”就是字符串“你好”。**现在这个变量不一样了，你看见一个a，实际上它不是字母a，而是里面蕴含的不同的值。**

## 4.1 整体感知

|  |
| --- |
| 1. <script type="text/javascript"> 2. **//定义一个变量** 3. var a; 4. **//赋值** 5. a = 100; 6. **//输出变量a** 7. console.log(a); 8. </script> |



**我们使用var关键字来定义变量**，**所谓的关键字就是一些有特殊功能的小词语，关键字后面要有空格**。

var就是英语variables变量的缩写，表示定义一个变量。一旦你

|  |
| --- |
| 1. var a; |

你的电脑内存中，就会开辟一个空间，来存储这个变量a。

现在就可以给这个变量赋值，**JS中给变量赋值用等号，等号右边的值赋给左边**。

|  |
| --- |
| 1. a = 100; |

现在a变量的值就是100。所以我们输出

|  |
| --- |
| 1. console.log(a); |

控制台就会输出a变量的值，就是



## 4.2 变量必须先声明，才能使用

使用一个变量，必须先进行一个var，才能使用。var这个过程可以叫做声明declaration，也可以叫做定义definition。

现在我们直接运行语句：

|  |
| --- |
| 1. console.log(b); //这个b没有被var过，所以要报错 |



因为b没有被定义，所以b现在不是一个变量，系统不认识这个b的。抛出引用错误。

**变量的名称是标识符**（identifiers），**任何标识符的命名都需要遵守一定的规则**：

在JavaScript语言中，一个标识符(identifier)可以**由字母、下划线（\_）、美元（$）符号、数字（0-9）组成，但不能以数字开头**。也就是说，一个标识符必须由字母、下划线、美元符号开头，后续可以有字母、下划线、美元符号、数字。因为JavaScript语言是区分大小写的，所以**A和a不是同一个变量**。并且**不能是JavaScript保留字、关键字**。

保留字，是系统里面的有用途的字，为了不造成误会，不能成为标识符的名字。

保留字，不用背：

abstract、boolean、byte、char、class、const、debugger、double、enum、export、extends、fimal、float  
goto、implements、import、int、interface、long、mative、package、private、protected、public、short、static、super、synchronized、throws、transient、volatile

合法变量名的举例：

|  |
| --- |
| 1. // 以下都是合法的变量名 2. var a; 3. var A; 4. var b333; 5. var \_abc; 6. var $; 7. var $o0\_0o$; 8. var \_\_\_\_\_\_\_\_; 9. var \_; |

非法的：

|  |
| --- |
| 1. var 123a; //不能数字开头 2. var 12\_a; //不能数字开头 3. var abc@163; //不能有特殊符号，符号只能有\_和$ 4. var abc￥; //不能有特殊符号，符号只能有\_和$ 5. var var; //不能是关键字 6. var class; //不能是保留字 |

变量的名字的长度实际上每个浏览器的规范不同，不用在意。

## 4.3 变量的赋值

**变量的赋值用等号，等号就是赋值符号，在JS中等号没有其他的含义，等号就表示赋值。**

|  |
| --- |
| 1. var a; //定义 2. a = 100; //赋初值 3. console.log(a); |

运行结果：



可以定义和赋初值一起写：

|  |
| --- |
| 1. **var a = 100;** 2. console.log(a); |

运行结果：



等号右边的值给左边，等号右边的值不变。

|  |
| --- |
| 1. var a = 100; 2. var b = 200; 3. a = b; **//在改变a变量的值，a的值变为200。b的值不变。** 4. console.log(a); 5. console.log(b); |

运行结果：



你会发现，变量只需要var一次，后面改变变量的值的时候，就不需要var了，可以直接改变它的值。

|  |
| --- |
| 1. var a = 100; 2. a = 50;  **//不需要写var** 3. console.log(a); |

**如果一个变量，仅仅被var了，但是没有被赋初值呢，此时这个变量的值就是undefined;**

|  |
| --- |
| 1. var m; 2. console.log(m); //输出undefined |

运行结果：



实际上我们已经var了这个m，已经定义了这个m，只不过这就是浏览器的一个规矩，如果这个变量没有被赋初值，那么这个变量就视为没有“定义完成”。值就是undefined。

## 4.4 变量声明的提升

这是js特有的一个特点，其他语言都没有这个特点。

我们现在先去改变变量的值，然后定义变量，由于JS有一个机制，叫做变量声明的提升，所以现在程序在执行前会已经看见这个程序中有一行定义变量，所以就会提升到程序开头去运行。

|  |
| --- |
| 1. <script type="text/javascript"> 2. a = 100; 3. var a; **//这行定义变量会自动提升到所有语句之前** 4. console.log(a); 5. </script> |



**记住，js只能提升变量的声明，而不能提升变量的赋初值**

|  |
| --- |
| 1. console.log(a); 2. var a = 100; |



等价于：

|  |
| --- |
| 1. var a;  **//自动升级** 2. console.log(a); 3. a = 100;  **//赋初值还留在原地** |

## 4.5 不写var的情况

|  |
| --- |
| 1. abc = 123; 2. console.log(abc); |



定义abc的时候没有写var，程序没有报错，说明这个abc变量真的已经被定义成功了。现在你看不出来var和不var的区别，感觉都是成功的，但是以后你就会知道**不写var定义了一个全局变量，作用域是不能控制的**。

## 4.6 用逗号来隔开多个变量的定义

|  |
| --- |
| 1. var a = 7 , b = 9 , c = 10; |

逗号这个表示法，只能用于变量的连续定义，不要瞎用。

## 4.7 区分变量还是直接量

变量的时候，不能加引号。如果变量加上引号，就变成字符串了。

|  |
| --- |
| 1. var a = 100; 2. console.log(**"**a**"**); |

而不是输出100,



引号是“字符串”的定界符，所以在双引号里面的a已经失去了变量的意思，就是一个字符串a。

# 五、变量的类型

## 5.1 概述

**基本类型5种**

number 数字类型

string 字符串类型

undefined undefined类型，变量未定义时的值，这个值自己是一种类型

boolean 布尔类型，仅有两个值true 和 false，讲if语句时我们细说

null null类型，这个值自己是一种类型，遇见时我们细说

**引用类型**

讲到再说

至于什么是引用类型，我们到时候见到再说。

## 5.2 typeof关键字

这个关键字，用来检测一个变量的类型。

|  |
| --- |
| 1. var a = 100; 2. console.log(**typeof** a); |



## 5.3 number类型

所有的数字都是number类型

|  |
| --- |
| 1. <script type="text/javascript"> 2. //下面定义的变量都是number类型 3. var a = 100; 4. var b = 234243245345; 5. var c = -345345435435; 6. var d = 345.3245234; 7. var e = .5e6; 8. var f = 0xff; 9. var g = 017; 10. var h = Infinity; 11. var i = NaN; 13. console.log(typeof a); 14. console.log(typeof b); 15. console.log(typeof c); 16. console.log(typeof d); 17. console.log(typeof e); 18. console.log(typeof f); 19. console.log(typeof g); 20. console.log(typeof h); 21. console.log(typeof i); 22. </script> |

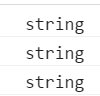


**JS中所有的数字都是number类型的**，不在细分为整形int、浮点型float这些乱七八糟的东西。

**number类型的东西：所有数字（不分正负、不分整浮、不分大小、不分进制）、Infinity、NaN。**

## 5.4 string类型

|  |
| --- |
| 1. var m1 = "哈哈"; 2. var m2 = "123"; 3. var m3 = ""; //空字符串，也是字符串 4. console.log(typeof m1); 5. console.log(typeof m2); 6. console.log(typeof m3); |



**这个变量是什么类型，和赋的值有关系，而和定义的时候是没有关系的。定义的时候，都是用var关键字定义的。**

|  |
| --- |
| 1. var mm = 123; **//number** 2. console.log(typeof mm); 3. mm = "哈哈"; **//改为string是合法的** 4. console.log(typeof mm); |



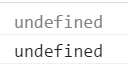
我们说js这个语言叫做**动态数据类型**，原因就在这里，就是赋值的时候就决定这个变量的类型了。甚至可以改变这个变量的值，为其他类型的值。**变量的类型是自动检测的，而不是人规定的。**

## 5.5 undefined类型

之前说过，一个变量只var过，没有赋初值，它的默认值是undefined;

这个undefined是自成一家，自己是自己的类型，它的类型也叫作undefined。undefined类型的值就只有undefined一个。

|  |
| --- |
| 1. var nn; //我只定义了，没有赋初值，所以就是undefined，类型也是undefined 2. console.log(nn); 3. console.log(typeof nn); |



## 5.6 加号

加号是一个运算符，我们先提前说一下。

加号两边都是数字的时候，那么就是数学加法；两边不都是数字的时候，那么就是连字符，结果是字符串。

|  |
| --- |
| 1. console.log("你" + "好"); |



|  |
| --- |
| 1. console.log(1 + **"**2**"**); |

，因为2被引号引起来了，所以就是字符串，所以+就是连字符



|  |
| --- |
| 1. var a = "1"; 2. var b = "2"; 3. console.log(a + b); |



|  |
| --- |
| 1. var a = 1; 2. var b = 2; 3. console.log("a" + b); //a2 |



多个加号，从左算到右：

|  |
| --- |
| 1. var a = 1; 2. var b = 2; 3. var c = 3; 4. console.log("a" + a + b + "c"); |



|  |
| --- |
| 1. var a = 1; 2. var b = 2; 3. var c = 3; 4. console.log(a + b + "c"); |

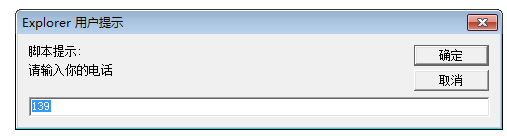


# 六、变量类型的转换

## 5.1 string → number

先来学习一个语句，这个语句和alert差不多，也是弹窗，弹的是输入框：

|  |
| --- |
| 1. prompt("请输入你的电话","139"); |



这些小功能，就叫做程序给我们提供的API，每个API都有自己不同的语法。

|  |
| --- |
| 1. prompt(“提示文本”,”默认值”); |

默认值可以省略。

可以把用户输入的值，存入变量：

|  |
| --- |
| 1. var a = prompt("请输入你的电话","139"); 2. alert("哈哈，你输入的电话是" + a); |

用prompt接收的任何东西都是字符串，哪怕用户输入了一个数字，也是字符串的数字。

有一些方法可以将内存中表示一个数字的字符串转换为对应的数字

parseInt()和parseFloat()。

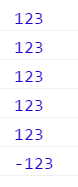
**parseInt就是将一个string转为一个整数，不四舍五入，直接截取整数部分。如果这个string有乱七八糟的东西，那么就截取前面数字部分。**

|  |
| --- |
| 1. var a = "123"; 2. var b = parseInt(a); **//parseInt就把字符串123转为数字123了** 3. console.log(b); 4. console.log(typeof b); |



下面的实例结果都是123，最后一个是-123：

|  |
| --- |
| 1. parseInt("123") //123 2. parseInt("123.6") //123 3. parseInt("123年都会很爱你") //123 4. parseInt("123年11月") //123 5. parseInt("123px") //123 6. parseInt("-123.99999999") //-123 |



parseInt()不仅仅能够进行一个转为整数，更能进行一个进制的转换，**把任何进制的数字，都换为10进制**。

进制和要转换的字符串，用逗号隔开。

下面的运算结果都是15：

|  |
| --- |
| 1. parseInt(15,10) 2. parseInt(17,8) 3. parseInt(1111,2) 4. parseInt("0xf",16) 5. parseInt("f",16) 6. parseInt(16,9) 7. parseInt("15e6",10) 8. parseInt("15\*6",10) |

parseInt如果不能转，那么就返回NaN

|  |
| --- |
| 1. parseInt("Hello", 8); 2. parseInt("546", 2); 3. parseInt("三百六十五"); |

得到NaN的方法又多了一种，之前讲的是6/0得到Infinity。0/0得到NaN。

parseFloat就是将字符串转为浮点数

尽可能的将一个字符串转为浮点数，浮点数之后如果有乱七八糟的内容，直接舍弃。

|  |
| --- |
| 1. <script type="text/javascript"> 2. var a = "123.67.88"; 3. var b = **parseFloat(**a**)**; 4. console.log(b); 5. </script> |



也就是说，数字类型都是number，不分整数和浮点数，但是转换的时候分。

## 5.2 number → string

将一个数字，与一个空字符串进行连字符运算，那么就是自动转为字符串了。

|  |
| --- |
| 1. var a = 123; 2. var b = a **+ ""**; 3. console.log(b); 4. console.log(typeof b); |



# 七、数学运算符

运算符叫做operator，也可以叫做操作符。运算符有很多种，一元运算符、二元运算符；数学运算符、逻辑运算符……我们今天先学习数学运算符，比较简单

+ 加法

- 减法

\* 乘法

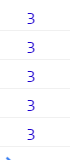
/ 除法

% 取余数

( ) 括号

下面的结果都是3：

|  |
| --- |
| 1. console.log(1 + 2); 2. console.log(8 - 5); 3. console.log(1.5 \* 2); 4. console.log(12 / 4); 5. console.log(13 % 5); //得几不重要，要的是余数 |



取余数这个运算，实际上也是除，要的是余数：

|  |
| --- |
| 1. //取余数 2. console.log(12 % 3); //0 3. console.log(121 % 11); //0 4. console.log(5 % 8); //5 5. console.log(8 % 5); //3 6. console.log(5 % 5); //0 |

默认的计算顺序，先乘除，后加减。**乘除取余是平级，先遇见谁，就算谁**。

|  |
| --- |
| 1. console.log(1 + 2 \* 3); //7 2. console.log(1 + 2 \* 3 % 3); //1 3. console.log(1 + 2 % 3 \* 3); //7 |

我们可以用小括号来改变计算先后顺序，注意没有中括号和大括号，一律用小括号。

|  |
| --- |
| 1. var a = 4 \* (3 + (1 + 2) \* 3); 2. alert(a); |

特殊的数字运算，很多公司在考，考你对面试的重视程度，因为这个知识实战中一辈子用不到。

JS中的数学运算符，就这么几个，如果你学过C，那么我提醒你，没有乘方^。如果你学过java那么我提醒你，没有取整除法\。

乘法怎么算？



|  |
| --- |
| 1. Math.pow(3,4); |

这是一个新的API，记住就行了，后面的课程将会告诉你，Math是一个内置对象，pow是它的一个方法。

Math就是数学，pow就是power乘方。



|  |
| --- |
| 1. var a = **Math.pow(**3,2 + **Math.pow(**3,6**));** |

根号就是

|  |
| --- |
| 1. Math.sqrt(81); |

今天遇见的所有API：

alert("哈哈");

prompt("请输入数字","默认值");

console.log("哈哈");

parseInt("123",8);

parseFloat("123");

Math.pow(3,4);

Math.sqrt(81);