JavaScript基础入门

第1天课堂笔记（本课程共3天）

# 目录

[目录 2](#_Toc6666)

[一、简单好学，富有表现力 —— JavaScript简介 3](#_Toc22180)

[1.1 JavaScript的用途 3](#_Toc4082)

[1.2 JavaScript历史背景介绍 3](#_Toc29797)

[1.3 JavaScript和ECMAScript的关系 3](#_Toc15590)

[1.4 今天的JavaScript：承担更多责任 4](#_Toc19720)

[1.5 JavaScript非常好学 5](#_Toc15908)

[1.6 我们的课程 6](#_Toc11003)

[二、JavaScript是前台语言，而不是后台语言 7](#_Toc30179)

[三、开始写第一个JavaScript程序 9](#_Toc11029)

[3.1 程序书写的位置 9](#_Toc11058)

[3.2 alert语句 9](#_Toc26482)

[3.3 语法规则 10](#_Toc27102)

[3.4 注释 11](#_Toc13890)

[四、认识数字和字符串 - 直接量 12](#_Toc10887)

[五、变量 14](#_Toc1045)

[5.1 整体感知 14](#_Toc19524)

[5.2 变量的命名规范 15](#_Toc19807)

[5.3 变量的定义和赋值 16](#_Toc29832)

[5.4 区分变量和字符串 16](#_Toc6477)

[六、变量的类型 17](#_Toc20272)

[6.1 数值型 17](#_Toc10278)

[6.2 字符串型 18](#_Toc14291)

[6.3 连字符和加号 18](#_Toc22988)

[七、变量值的传递 20](#_Toc24275)

[八、运算符和表达式 23](#_Toc13588)

[8.1 数学运算符 23](#_Toc20420)

[8.2 乘方和开根号 25](#_Toc32524)

[九、变量格式转换 27](#_Toc11289)

[9.1 用户的输入 27](#_Toc2423)

[9.2 字符串→数字 27](#_Toc17470)

# 一、简单好学，富有表现力 —— JavaScript简介

## 1.1 JavaScript的用途

**JavaScript用来制作web页面交互效果，提升用户体验。**

简单列出几个JavaScript能够制作的页面效果，它能干什么：

|  |  |  |  |
| --- | --- | --- | --- |
| 轮播图 | Tab栏（选项卡） | 地图 | 表单验证 |

web前端三层来说：

结构层 HTML 从语义的角度，描述页面结构

样式层 CSS 从审美的角度，美化页面

行为层 JavaScript 从交互的角度，提升用户体验

## 1.2 JavaScript历史背景介绍

布兰登·艾奇（Brendan Eich，1961年～），1995年在网景公司，发明的JavaScript。



一开始JavaScript叫做LiveScript，但是由于当时Java这个语言特别火，所以为了傍大牌，就改名为JavaScript。如同“北大”和“北大青鸟”的关系。“北大青鸟”就是傍“北大”大牌。

同时期还有其他的网页语言，比如VBScript、JScript等等，但是后来都被JavaScript打败，所以现在的浏览器中，只运行一种脚本语言就是JavaScript。

## 1.3 JavaScript和ECMAScript的关系

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。

## 1.4 今天的JavaScript：承担更多责任

**2003年之前，JavaScript被认为“牛皮鲜”**，用来制作页面上的广告，弹窗、漂浮的广告。什么东西让人烦，什么东西就是JavaScript开发的。所以浏览器就推出了屏蔽广告功能。

**2004年JavaScript命运开始改变了，那一年谷歌公司，开始带头使用Ajax技术了，Ajax技术就是JavaScript的一个应用**。并且，那时候人们逐渐开始提升用户体验了。

百度的智能感应



再比如：

网易的Ajax验证



2007年乔布斯发布了iPhone，这一年开始，用户就多了上网的途径，就是用移动设备上网。

**JavaScript在移动页面中，也是不可或缺的。**并且这一年，互联网开始标准化，按照W3C规则三层分离，人们越来越重视JavaScript了。

聚划算的手机页面

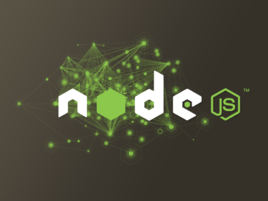


2010年的时候，人们更加了解HTML5技术了，**HTML5推出了一个东西叫做Canvas（画布），工程师可以在Canvas上进行游戏制作，利用的就是JavaScript**。

canvas制作的水果忍者



2011年，Node.js诞生，使JavaScript能够开发服务器程序了。



**今天，JavaScript工程师是绝对的吃香，能够和iOS、Android工程师比肩，毫不逊色的。**

现在，公司都流行WebApp，就是用网页技术开发手机应用。什么意思呢？手机系统有iOS、安卓、windows phone。那么公司比如说开发一个“携程网”APP，就需要招聘三队人马，比如iOS工程师10人，安卓工程师10人，windows工程师10人。共30人，工资开销大。并且，如果要改版，要改3个版本。所以，现在公司，都用web技术，用html+css+javascript技术来开发app。好处是不用招聘那么多工程师，只需要几个前端开发工程师即可。并且也易于迭代，就是网页一改变，所有的终端都变了。

## 1.5 JavaScript非常好学

JavaScript在“对初学者友好的语言”排行榜中排名第一。

JavaScript是有界面效果的，比如你学习C语言，白底黑字。而JavaScript有绚丽的效果，效果是可见的。

JavaScript是弱变量类型的语言，变量只需要用var来声明。Java中变量的声明，要根据变量是什么类型的来声明：

|  |
| --- |
| 1. int a; 2. float a; 3. double a; 4. String a; 5. boolean a; |

JavaScript中，只用一个：

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

JavaScript不用关心一些其他的事情，比如内存的释放，指针。程序员只需要关心自己的业务，不需要关系这些鸡毛蒜皮的破事儿。

## 1.6 课程安排

JavaScript分为几个部分：

**● 语言核心 - 基础班只学习语言核心，变量、表达式、运算符、函数、if语句、for语句**

● DOM - 控制HTML中的元素，比如让盒子移动、变色、轮播图。

● BOM - 控制浏览器的一些东西，比如让浏览器自动滚动。

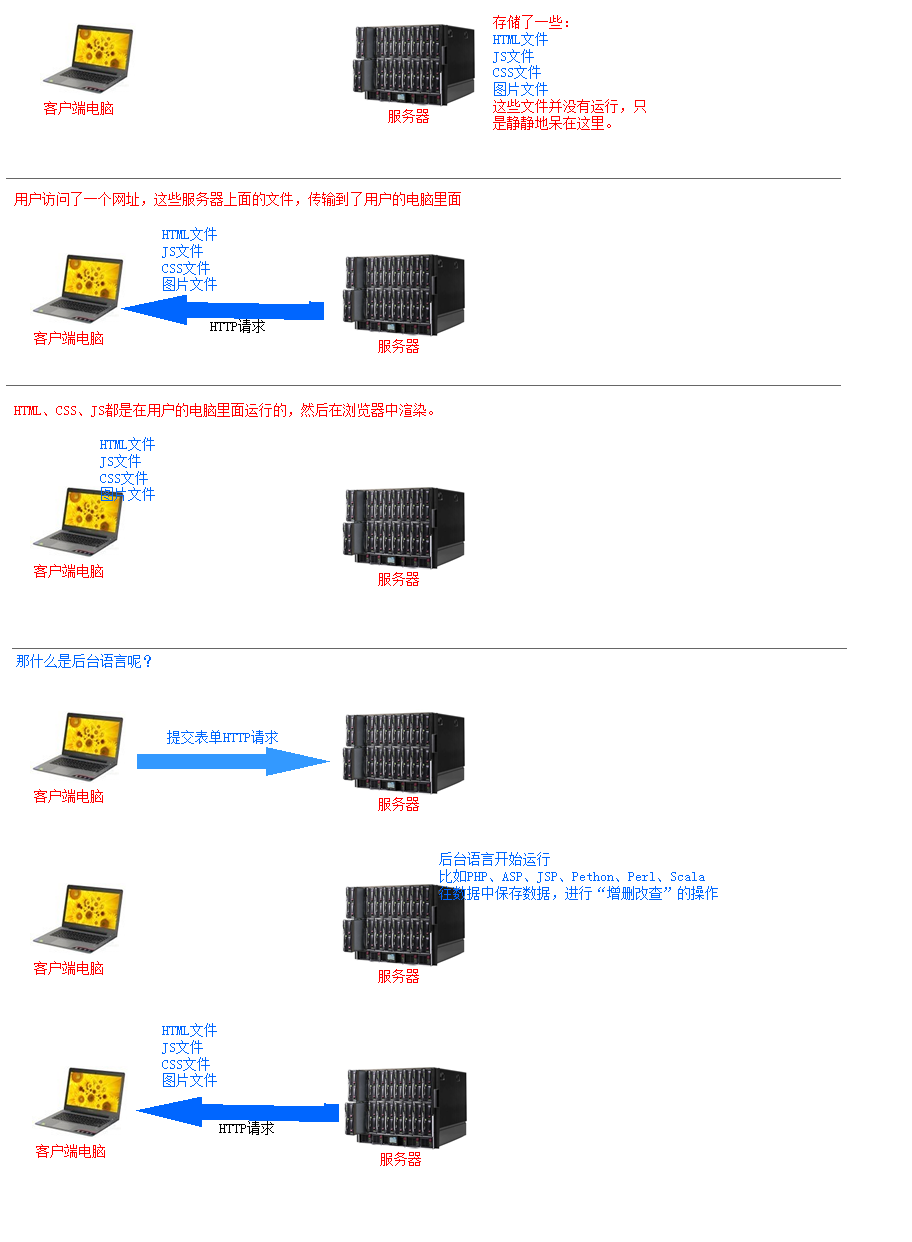
JavaScript的学习方法和HTML、CSS有着非常大的区别：

● 要多去“品”程序，多去思考内在逻辑。 HTML、CSS好比富士康，人力密集型；JS好比发条手表，很精密，令人啧啧称奇。

● JS机械重复性的劳动几乎为0，基本都是创造性的劳动。而不像HTML、CSS中margin、padding都是机械重复劳动。

● 永远不要背程序，每一个程序都必须自己会写。

# 二、JavaScript是前台语言，而不是后台语言



**JavaScript运行在客户的电脑里面，而不是服务器上**，所以我们称为“前台语言”。JavaScript就是一个简单的制作页面效果的语言，不能操作数据库。就是服务于页面的交互效果、美化、绚丽。

“后台语言”是运行在服务器上的，比如PHP、ASP、JSP等等，这些语言都能够操作数据库，都能够对数据库进行“增删改查”操作。

（Node.js除外，先别管Node.js是什么）。

● 比如一个图书馆，要开发“图书借阅程序”，能够记录每个学生借了什么书，有没有按时归还，不能用JS开发！因为，设计数据库的数据记录。

● 比如，一个公司要开发“订餐系统”，每天上午11:00统计所有的员工想吃什么？不能用JS开发。因为涉及数据库的数据记录。

● 比如，一个公司的网页想要做的漂亮、有交互效果，绚丽。用JS开发。

# 三、开始写第一个JavaScript程序

## 3.1 程序书写的位置

在页面中，**<script type=”text/javascript”></script>标签对儿，里面就是书写JavaScript程序的地方**。

|  |
| --- |
| 1. <script type="text/javascript"> 2. </script> |

text表示什么？纯文本。JavaScript也是一个纯文本的语言。

## 3.2 alert语句

学习第一个语句，就是alert语句。

|  |
| --- |
| 1. <script type="text/javascript"> 2. **alert("今天天气真好");** 3. </script> |

alert就是英语里面的“警报”的意思。用途就是弹出“警告框”：



如果我们想弹出两次警告框，那么就要写两条语句：

|  |
| --- |
| 1. **alert("**今天天气真好**");** 2. **alert("**哈哈哈哈**");** |

学习程序，是有规律可循的，就是程序是有相同的部分，这些部分就是一种规定，不能更改的，我们成为：语法。

**至于为什么alert后面有一个圆括号，为什么里面又有引号，我们现在先不管。因为你知道，只要我按照这个语法书写，功能就会实现。**

世界上不管什么编程语言，都有一个规定，**程序是一句一句执行，执行完上面的语句，才能之后下面的语句**：

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("今天天气真好"); 3. alert("哈哈哈哈"); 4. </script> |

## 3.3 语法规则

JavaScript对换行、缩进、空格不敏感：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **alert("今天蓝天白云");** 3. **alert("哈哈哈，我很高兴");** 4. </script> |

等价于

|  |
| --- |
| 1. <script type="text/javascript"> 2. **alert("今天蓝天白云");** 3. **alert("哈哈哈，我很高兴");** 4. </script> |

等价于：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **alert("今天蓝天白云");alert("哈哈哈，我很高兴");** 3. </script> |

**每一条语句末尾要加上分号，虽然分号不是必须加的，但是为了程序今后要压缩，如果不加分号，压缩之后将不能运行了。**

比如，你不加分号，今后我们的程序一旦压缩，就不能使用了：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **alert("今天蓝天白云")alert("哈哈哈，我很高兴")** 3. </script> |

所有的符号，都是英语的。比如括号、引号、分号：



第二行语句所有的符号都是中文的，不对！！

还有一个常见错误，就是script标签写错了：

|  |
| --- |
| 1. <script type="text/jav**e**script"> 2. alert("今天蓝天白云"); 3. alert("哈哈哈，我很高兴"); 4. </script> |

## 3.4 注释

程序中，为了便于理解、教学，我们可以写一些给人看的东西，这些内容就是注释，程序不会运行。

//表示注释：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **//**alert语句表示弹出窗口 3. alert("你好"); 4. </script> |

也可以：

|  |
| --- |
| 1. <script type="text/javascript"> 2. alert("你好"); **//alert语句表示弹出窗口** 3. </script> |

注释可以很多行，但是比较麻烦：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **//**alert语句表示弹出窗口 3. **//**一定要注意用英语的符号啊！ 4. **//**千万不要再错了！ 5. alert("你好"); 6. </script> |

等价于：

|  |
| --- |
| 1. <script type="text/javascript"> 2. **/\*** 3. alert语句表示弹出窗口 4. 一定要注意用英语的符号啊！ 5. 千万不要再错了！ 6. **\*/** 7. alert("你好"); 8. </script> |

总结一下：

// 单行注释

/\*

多行注释

多行注释

\*/

sublime中，单行注释的快捷键是ctrl+/ 。 多行注释的快捷键是ctrl+shift+/

# 四、认识数字和字符串 - 直接量

**“直接量”也称为“字面量”，就是看见什么，它就是什么。**

简单的直接量有2种：数字、字符串。

数值的直接量的表达非常简单，就是写上去就行了，不需要任何的符号：

|  |
| --- |
| 1. alert(886); //**886是数字，所以不需要加引号**。 |

**字符串，就是人说的话，比如单词、句子，它们不是数字。一定要加上引号。**

|  |
| --- |
| 1. alert(今天天气很好); **//错误的语句，因为没有加上引号** |

控制台报出错误：



正确的：

|  |
| --- |
| 1. alert(**"**今天天气很好**"**); //今天天气很好，是字符串，必须加上引号。 |

小练习，下面的语句是否正确：

|  |
| --- |
| 1. alert(八八六); //错误的！因为八八六是汉字，必须加上引号 2. alert(5.67); //正确的！因为5.67是数字，不用加上引号 3. alert("100"); //**正确的！虽然100是数字，但是可以是人说的啊，所以加上引号也正确。** |

下午将学习 ”100” 和100不是一个东西！

★★★★★阶段性小练习★★★★★

|  |
| --- |
| 1. alert(1000); √ 2. alert("1000"); √ 3. alert(我爱你) **×** 4. alert("我爱你"); √ 5. alert(五百万); **×** 6. alert("五百万"); √ 7. alert("50000000"); √ |

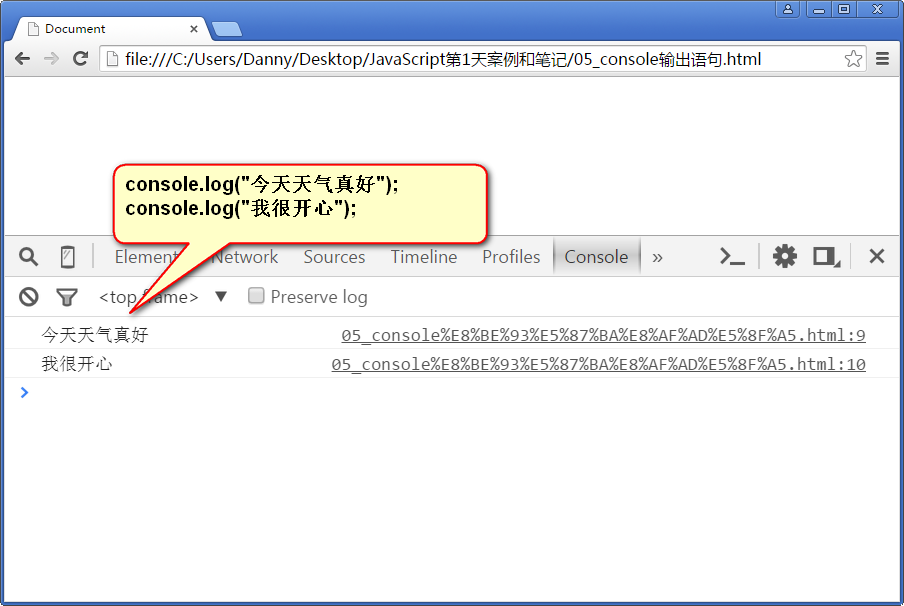
**！！下午课程**

我们上午学习了一个语句，叫做alert弹出警告框。我们现在再学习一条语句：

|  |
| --- |
| 1. console.log("今天天气真好"); |

console表示“控制台”，log就是“输出”

表示在控制台中输出，控制台在Chrome浏览器的F12中。控制台是工程师、程序员调试程序的地方。我们经常使用这条语句输出一些东西，来测试程序是否正确。很像电工用的“电笔”。老百姓不会在意这里的内容。



两种语句，你只需要知道，照着写，就能实现效果：

|  |
| --- |
| 1. alert("哈哈"); 2. console.log("哈哈"); |

# 五、变量

## 5.1 整体感知

初中的时候，学习了一个学科“代数”，x、y、z、a、b、c。

**计算机的程序中，也是有这样的量，就是用字母来表示数字、字符串等其他东西的，称为“变量”**。

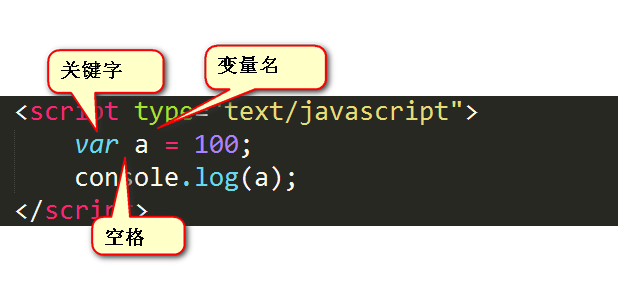
|  |
| --- |
| 1. var a = 100; 2. console.log(a); |



var a = 100;

这就是一个固定的写法，就是语法。也就是说，我们使用var来定义一个变量。

var就是英语“**var**iant”变量的缩写。后面要加一个空格，空格后面的东西就是“变量名”，我们可以给变量任意的取名字。



var就是一个关键字，所谓关键字，就是有特殊功能的小词语。关键字后面一定要有空格隔开。

**等号表示赋值，会将等号右边的值，赋给左边的变量。**

## 5.2 变量的命名规范

变量名有命名规范：

**只能由英语字母、数字、下划线、美元符号$构成，且不能以数字开头，并且不能是JavaScript保留字**。

下列都是非常正确的变量命名：

|  |
| --- |
| 1. var haha = 250; 2. var xixi = 300; 3. var a1 = 400; 4. var a2 = 400; 5. var abc\_123 = 400; 6. var $abc = 999; 7. var $o0\_0o$ = 888; 8. var $ = 1000; 9. var \_ = 2000; 10. var \_\_\_\_\_\_\_\_ = 3000; |

下列都是错误的命名：

|  |
| --- |
| 1. //var a-1 = 1000; //不能有怪异符号 2. //var a@ = 2000; //不能有怪异符号 3. //var 2year = 3000; //不能以数字开头 4. //var a￥ = 4000; //不能有怪异符号 5. //var a\*#$#$@ = 5000; //不能有怪异符号 6. //var a b =300; //不能有空格 |

下列的单词，叫做保留字，就是说不允许当做变量名，不用记：

abstract、boolean、byte、char、class、const、debugger、double、enum、export、extends、final、float、goto

implements、import、int、interface、long、native、package、private、protected、public、short、static、super、synchronized、throws、transient、volatile

大写字母是可以使用的，并且大小写敏感。也就是说A和a是两个变量。

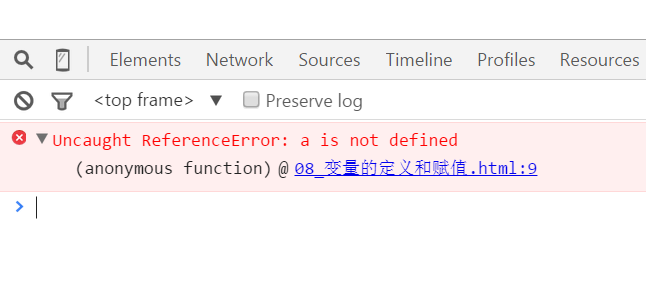
|  |
| --- |
| 1. var A = 250; 2. var a = 888; |

## 5.3 变量的定义和赋值

变量用var来定义。只有定义之后，这个变量才能够使用。

比如，我们不设置变量，直接输出：

|  |
| --- |
| 1. <script type="text/javascript"> 2. console.log(a); 3. </script> |



正确：

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

有经验的程序员，会把定义和赋值写在一起：

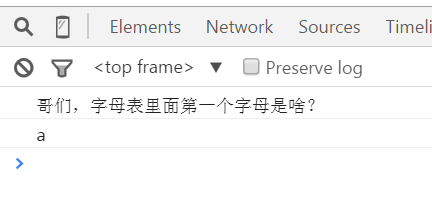
|  |
| --- |
| 1. var a = 100; //定义，并且赋值100 2. console.log(a); //输出100 |

在JavaScript中，永远都是用var来定义变量，这点C、Java等等既然不同。

赋值用等于号，表示等号右边的值，赋给左边的变量。

## 5.4 区分变量和字符串

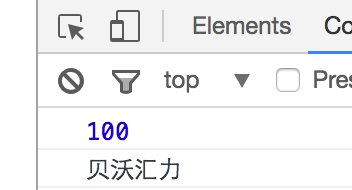
|  |
| --- |
| 1. var a = 100; 2. console.log("哥们，字母表里面第一个字母是啥？"); 3. console.log(**"**a**"**); //输出字母a |



这个a在引号里面，所以就是一个字符串“a”了，而不是一个变量。换句话说，一个变量如果想输出保存的内容，那么就不能加引号。

# 六、变量的类型

|  |
| --- |
| 1. var a = 100; //存放了一个数字 2. var b = **"**贝沃汇力**"**; //存放了一个字符串 3. console.log(a); //输出变量a 4. console.log(b); //输出变量b |



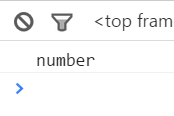
变量里面能够存储数字、字符串。变量会自动的根据存储的东西类型不同，来决定自己的类型。

也就是说变量有类型。

## 6.1 数值型

如果一个变量中，存放了数字，那么这个变量就是数值型的。

|  |
| --- |
| 1. var a = 100; //定义了一个变量a，并且赋值100 2. console.log(typeof a); //输出a变量的类型 |



typeof表示“某某的类型”

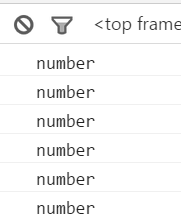
语法：

|  |
| --- |
| 1. typeof 变量 |

JavaScript种，只要是个数，那么就是数值型的，**无论整浮、无论大小、无论正负，都是number类型的**。

“浮”就是浮点数，就是“小数”，术语叫做“浮点数”。

|  |
| --- |
| 1. var a = 100; //定义了一个变量a，并且赋值100 2. var b = 234234234324324324; 3. var c = 3.234234234324324; 4. var d = -23423432432432432; 5. var e = -34.3423423432; 6. var f = 0.324234234; 7. console.log(typeof a); 8. console.log(typeof b); 9. console.log(typeof c); 10. console.log(typeof d); 11. console.log(typeof e); 12. console.log(typeof f); |

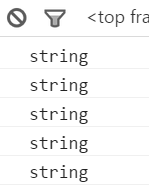


就业班中，老师还将给大家拓展数值型的东西，今天来说先到这里。

## 6.2 字符串型

string就是字符串型的意思。

|  |
| --- |
| 1. var a = **"**abcde**"**; 2. var b = **"**贝沃汇力**"**; 3. var c = **"**123123**"**; 4. var d = **"**哈哈哈哈哈**"**; 5. var e = **""**; **//空字符串** 6. console.log(typeof a); 7. console.log(typeof b); 8. console.log(typeof c); 9. console.log(typeof d); 10. console.log(typeof e); |



## 6.3 连字符和加号

键盘上+在JS中有两层含义：

1） 连字符

2） 加

|  |
| --- |
| 1. console.log(**"**我**"** + **"**爱**"** + **"**你**"**); //连字符，把三个独立的汉字，连接在一起了 2. console.log(**"**我+爱+你**"**); //原样输出 3. console.log(1+2+3); //输出6 |



同样是加号，有时候表示的是连字符，有时候表示的是加号。什么时候是连字符？什么时候是加呢？

**如果加号两边都是数值，此时是加。否则，就是连字符。**

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



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



|  |
| --- |
| 1. var a = 234234; 2. var b = 234323112; 3. console.log("a+b"); //由于加上了引号，所以就表示原样输出 |



|  |
| --- |
| 1. var a = 1; 2. var b = 2; 3. console.log("a" + b); //"a"就不是变量了！所以就是"a"+2 输出a2 |



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



# 七、变量值的传递

语句：

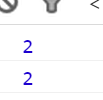
|  |
| --- |
| 1. a = b; |

**将等号右边的值，赋给左边的变量；等号右边的变量，值不变。**

**把b的值赋给a，b不变。**

案例：

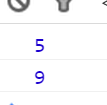
|  |
| --- |
| 1. var a = 1; //定义a，并且赋值1 2. var b = 2; //定义b，并且赋值2 3. a = b; //就是将b的值给a，b的值不变。所以a就是2了，b是2不变。 4. console.log(a); //2 5. console.log(b); //2 |



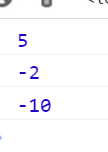
|  |
| --- |
| 1. var a = 1; 2. var b = 2; 3. b = a; //将a的值给b，a的值不变，所以b就是1了，a还是1不变 4. console.log(a); 5. console.log(b); |



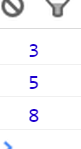
|  |
| --- |
| 1. var a = 1; //定义a变量，值是1 2. var b = 2; //定义b变量，值是2 3. a = b + 3; //将5赋值给a，b的值还是2。 4. b = a + 4; //将9赋值给b，a的值还是5 5. console.log(a); //5 6. console.log(b); //9 |



|  |
| --- |
| 1. //a b c 2. var a = 1; //1 3. var b = 2; //1 2 4. var c = 3; //1 2 3 5. a = b + c; //5 2 3 6. b = c - a; //5 -2 3 7. c = a \* b; //5 -2 -10 8. console.log(a); 9. console.log(b); 10. console.log(c); |



|  |
| --- |
| 1. //a b c 2. var a = 1; 3. var b = 2; 4. var c = 3; //1 2 3 5. a = a + b; //3 2 3 6. b = b + a; //3 5 3 7. c = c + b; //3 5 8 8. console.log(a); //3 9. console.log(b); //5 10. console.log(c); //8 |



|  |
| --- |
| 1. //a b 2. var a = "1"; 3. var b = 2; //"1" 2 4. a = a + b; //"12" 2 5. b = b + a; //"12" "212" 6. console.log(a); //输出12 7. console.log(b); //输出212 |



|  |
| --- |
| 1. //a b 2. var a = "1"; 3. var b = 2; 4. a = b + a; //"21" 2 5. b = b + a; //"21" "221" 6. console.log(a); //21 7. console.log(b) //221 |



# 八、运算符和表达式

## 8.1 数学运算符

整体感知：

我们比如要计算这个：



JavaScript中正确的表达式就是：

|  |
| --- |
| 1. (3 + 4 \* 5) / (6 + 3) |

+、\*、/、( 都是**运算符**

这个式子叫做**表达式**。

运算符有很多分类：数学运算符、逻辑运算符、自增运算符等等。我们今天只学习**数学运算符**。

|  |  |
| --- | --- |
| + | 加 |
| - | 减 |
| \* | 乘 |
| / | 除（问号杠） |
| % | 取余数 |
| ( ) | 括号 |

先算乘除、后算加减：

|  |
| --- |
| 1. var a = 1 + 2 \* 3; 2. console.log(a); |



|  |
| --- |
| 1. var a = 12 / 2 \* 3 + 2; 2. console.log(a); |



小括号，能够影响计算顺序。没有中括号、没有大括号，只有小括号。小括号能嵌套

|  |
| --- |
| 1. var a = (1 + 2) \* 3 + 3 \* 5 2. console.log(a); |



解：原式 = 3\*3 + 15

= 9 + 15

= 24

|  |
| --- |
| 1. var a = ((1 + 2) \* 3 + 3) \* 5 2. console.log(a); |

解：原式 = (3\*3+3)\*5

= (9+3)\*5

= 12\*5

= 60

还能多个嵌套

|  |
| --- |
| 1. var a = (6 + ((1 + 2) \* 3 + 3)) \* 5 2. console.log(a); |



百分号是取余数的意思

|  |
| --- |
| 1. var a = 13 % 5; 2. console.log(a); //输出3 |



因为13 ÷ 5 = 2 …… **3**， 所以结果是3。 得几不重要，我们关心的就是余数。

|  |
| --- |
| 1. console.log(10 % 5); |



因为10 ÷ 5 = 2 …… **0 ，** 所以结果是0。得几不重要，我们关心的就是余数。

|  |
| --- |
| 1. console.log(3 % 5); |



因为3 ÷ 5 = 0 …… **3 ，** 所以结果是3。得几不重要，我们关心的就是余数。

乘、除、取余数的运算优先级相同，谁写在前面，先算谁。

|  |
| --- |
| 1. var a = 1 + 2 \* 3 % 4 / 3; 2. console.log(a); |



解：原式 = 1 + 6 % 4 / 3

= 1 + 2 / 3

= 1 + 0.6666666666

= 1.66666666666666

|  |
| --- |
| 1. var a = (1 + 2) % 4 \* 3 + 5 2. console.log(a); |



解：原式 = 3 % 4 \* 3 + 5

= 3 \* 3 + 5

= 14

## 8.2 乘方和开根号

比如：

 = 3 \* 3 \* 3 \* 3

|  |
| --- |
| 1. var a = **Math.pow(**3,4**)**; 2. console.log(a); |



语法，如果想计算ab

|  |
| --- |
| 1. Math.pow(a,b); |

还是那句话，你不懂为啥突然乱入了一个Math，但是你要知道只要这么写了，就有效果。

Math是英语“数学”，pow就是“power”乘方的意思。



|  |
| --- |
| 1. var a = Math.pow(3,4\*5); 2. console.log(a); |



|  |
| --- |
| 1. var a = Math.pow(3,Math.pow(2,2)); 2. console.log(a); |



|  |
| --- |
| 1. var a = Math.pow(Math.pow(3,2),4); 2. console.log(a); |

开根号：



|  |
| --- |
| 1. var a = **Math.sqrt(**81**)**; 2. console.log(a); |

sqrt就是英语“开根号”的意思。

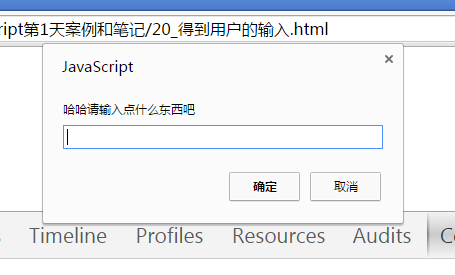
自己在手册中找寻，如何计算sin、cos。

# 九、变量格式转换

## 9.1 用户的输入

|  |
| --- |
| 1. <script type="text/javascript"> 2. var a = **prompt(**"哈哈请输入点什么东西吧"**);** 3. console.log(a); 4. </script> |

prompt就是专门用来弹出能够让用户输入的对话框：



和alert很像，又不像：

|  |
| --- |
| 1. alert("从前有座山"); //直接使用，不需要变量 2. var a = **prompt(**"请输入一个数字"**);** // 必须用一个变量，来接收用户输入的值 |

用户不管输入什么，都是字符串！

## 9.2 字符串→数字

方法就是：

|  |
| --- |
| 1. parseInt(“5”); |

结果就是数字5。

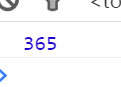
parse是英语里面的转换的意思，Int表示整数。注意拼写：

parseInt

● parseInt带有自动净化的功能：

|  |
| --- |
| 1. console.log(**parseInt("365天每天都爱你10000次")**); |

后面的中文自动消失，只保留最开头的数字。



● 自动带有截断小数功能（取整，不四舍五入）

|  |
| --- |
| 1. console.log(**parseInt(5.8)**); |



总结一下，parseInt是一个非常多功能的东西，可以将字符串转为数字，也可以将数字取整。

|  |
| --- |
| 1. var a = parseInt(5.8) + parseInt(4.7); 2. console.log(a); |



|  |
| --- |
| 1. var a = parseInt(5.8 + 4.7); 2. console.log(a); |



|  |
| --- |
|  |

|  |
| --- |
|  |