# 第12天【JavaScript变量和数据类型】

## 主要内容

1. JavaScript简介
2. JavaScript 发展史
3. JavaScript用法
4. JavaScript输出
5. JavaScript语法
6. 变量
7. JavaScript数据类型
8. 注释

## 学习目标

|  |  |  |
| --- | --- | --- |
| 节数 | 知识点 | 要求 |
| 第一节 | JavaScript简介 | 了解 |
| JavaScript发展史 | 了解 |
| JavaScript用法 | 掌握 |
| JavaScript输出 | 掌握 |
| 第二节 | 语法 | 掌握 |
| 变量1 | 掌握 |
| 第三节 | 变量2 | 掌握 |
| 第四节 | JavaScript数据类型 | 掌握 |
| 数据类型转换 | 掌握 |

1. **JavaScript简介**

JavaScript 是互联网上最流行的脚本语言，这门语言可用于 HTML 和 web，更可广泛用于服务器、PC、笔记本电脑、平板电脑和智能手机等设备。

JavaScript用来制作web页面交互效果，提升用户体验。

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在今天，如果你要离开他，寸步难行，网页将无法正常使用。

1. **JavaScript发展史**

### 2.1JavaScript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ECMAScript5。



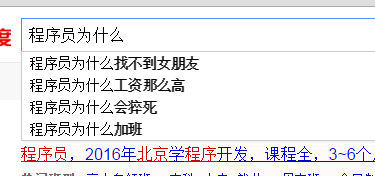
Brendan Eich (JavaScript之父)

出任Mozilla CTO

### 2.2JavaScript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。

### 2.3JavaScript与ECMAScript的关系：

虽然JavaScript和ECMAScript通常被人用来表达相同的意思，但JavaScript的含义去比ECMA-262中规定的多得多。一个完整的JavaScript实现应由三个部分组成：

1.核心（ECMAScript）

2.文档对象模型（DOM）

3.浏览器对象模型（BOM）

ECMA：欧洲计算机制造商协会

1. **JavaScript用法**

JavaSript程序，要写在HTML页中，**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，最最常见的宿主面环境，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。**

注意：

1）HTML 中的脚本必须位于 <script> 与 </script> 标签之间。

2）脚本可被放置在 HTML 页面的 <body> 和 <head> 部分中。

常用的两种方法：

1）在head标签里面引入外部的.js文件

|  |
| --- |
| <script src="myScript.js"></script> |

2）在body结束标签之前引入内部脚本

|  |
| --- |
| <script>  alert("我的第一个 JavaScript");  </script> |

1. **JavaScript输出**

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。

### 4.1输出到控制台

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：

|  |
| --- |
| 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：

|  |
| --- |
| 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：

|  |
| --- |
| console.log("你好啊"); |

养成打开控制台调试程序的习惯。

### 4.2输出到网页

将内容写到 HTML 文档中

|  |
| --- |
| document.write(); |

### 4.3弹出警告框

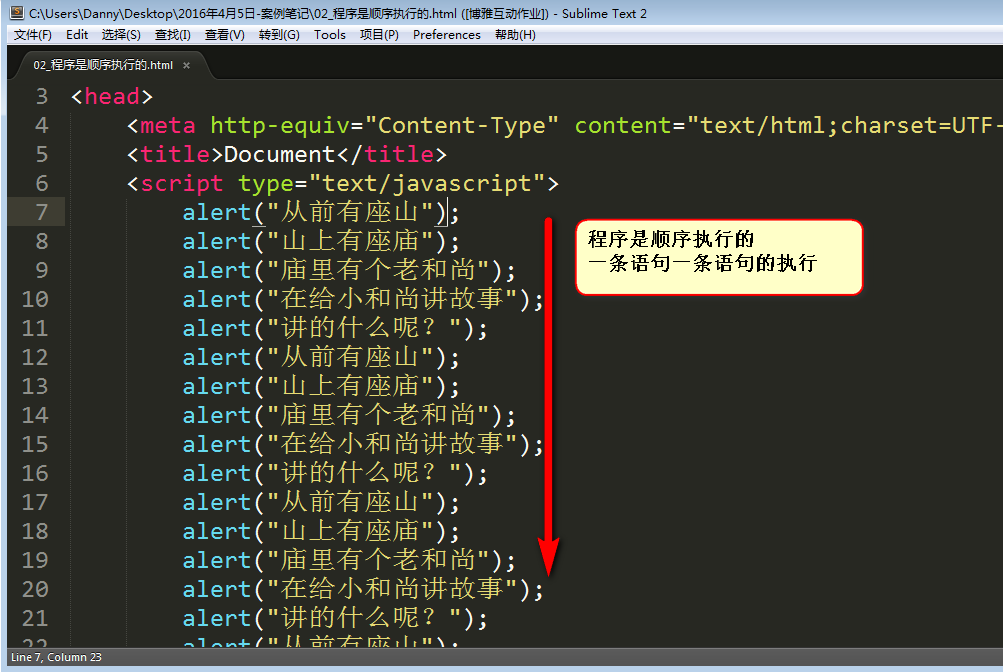
alert就是英语中“警报”的意思，能够让页面弹出警告框。

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。

|  |
| --- |
| 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**。**

1. **JavaScript语法**

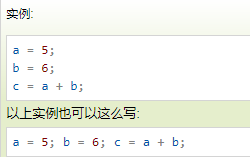
1、JavaScript大小写敏感

2、分号用于分隔 JavaScript 语句。

3、通常我们在每条可执行的语句结尾添加分号。

4、使用分号的另一用处是在一行中编写多条语句。

ps：多个语句可以写在一行内。但是一般情况下一行只写一条语句，并且以分号结尾



1. JavaScript会忽略多余的空格

**JavaScript语句和语句之间的换行、空格、缩进都不敏感。**

|  |
| --- |
| <script type="text/javascript">  alert("你好");alert("帅");  alert("哈哈哈哈");  </script> |

等价于：

|  |
| --- |
| <script type="text/javascript">  alert("你好");  alert("帅");  alert("哈哈哈哈");  </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：

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

正确写法：

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

1. **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。**

变量是用于存储信息的"容器"。

|  |
| --- |
| var x=5;  var y=6;  var z=x+y; |

### 6.1声明变量

**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

|  |
| --- |
| 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**。

|  |
| --- |
| a = 100; |

现在a变量的值就是100。所以我们输出

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

控制台就会输出a变量的值，就是



声明变量的同时给变量赋值，称为变量的初始化。

|  |
| --- |
| var a = 100; |

### 6.2一条语句可以声明多个变量

可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

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

声明也可横跨多行：

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

一条语句中声明的多个不可以赋同一个值:

|  |
| --- |
| var x,y,z=1; |

x,y 为 undefined， z 为 1。

### 6.3声明变量没有赋值

声明变量没有赋值，不会报错，默认值是undefined（未定义）

|  |
| --- |
| var a;  console.log(a);//undefined |

### 6.4变量必须声明，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：

|  |
| --- |
| 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。

### 6.5重新声明变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Ferrari"：

|  |
| --- |
| var carname="Ferrari";  var carname; |

### 6.6变量重新赋值

变量重新赋值后，后面的值会替换前面的值。

|  |
| --- |
| var a = 10;  console.log(a);//10  a = 100;  console.log(a);//100 |

### 6.7变量声明提升

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。

记住，js只能提升变量的声明，而不能提升变量的赋值。

|  |
| --- |
| console.log(a);//undefined  var a =100;  console.log(a);//100 |

等价于：

|  |
| --- |
| var a;  console.log(a);//undefined  a =100;  console.log(a);//100 |

### 6.8不写var的情况

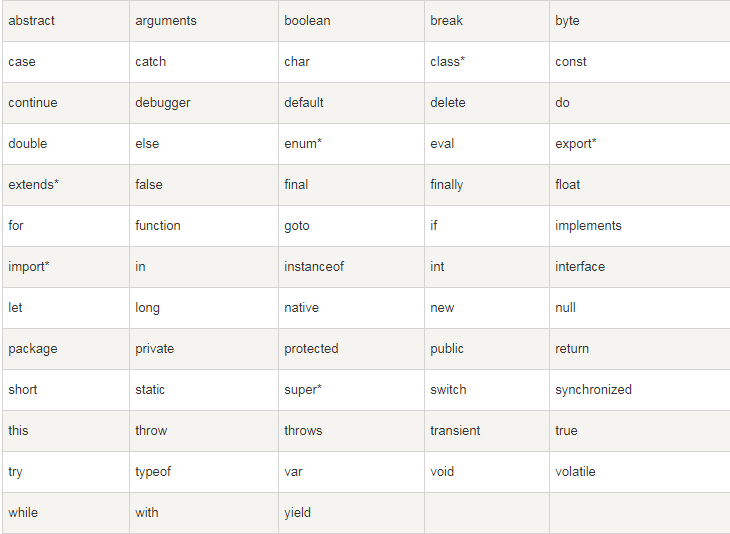
|  |
| --- |
| abc = 123;  console.log(abc);//123 |

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**。

### 6.9变量命名规则

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。多个单词建议驼峰命名法，建议起有意义的名字，尽量不要起a、b、c这些。

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。

合法变量名的举例：

|  |
| --- |
| // 以下都是合法的变量名  var a;  var A;  var b333;  var \_abc;  var $;  var $o0\_0o$;  var \_\_\_\_\_\_\_\_;  var \_; |

非法的：

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

1. **JavaScript数据类型**

Javascript中有六大基本数据类型，这六种数据类型也就是我们声明变量的类型。

基本类型5种

number 数字类型

string 字符串类型

boolean 布尔类型，仅有两个值true 和 false

undefined 未定义类型，变量未定义时的值，这个值自己是一种类型

null 空类型

引用类型

object 对象类型

### 7.1number类型

**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。**

|  |
| --- |
| <script type="text/javascript">  //下面定义的变量都是number类型  var a = 100;  var b = 234243245345;  var c = -345345435435;  var d = 345.3245234;  var e = 5e6;  var f = 5e-6  var g = -0.66;  var h = Infinity;  var i = NaN;    console.log(typeof a);//number  console.log(typeof b);//number  console.log(typeof c);//number  console.log(typeof d);//number  console.log(typeof e);//number  console.log(typeof f);//number  console.log(typeof g);//number  console.log(typeof h);//number  console.log(typeof i);//number  </script> |

### 7.2string类型

字符串是存储字符（比如 “KrisWu"）的变量。字符串可以是引号中的任意文本。可以使用单引号或双引号。

|  |
| --- |
| var m1 = "哈哈";  var m2 = "123";  var m3 = ""; //空字符串，也是字符串  console.log(typeof m1);//string  console.log(typeof m2);//string  console.log(typeof m3);//string |

可以在字符串中使用引号，只要不与包围字符串的引号冲突即可：

|  |
| --- |
| var answer = “He is called 'Kris'”;//外层是双引号，内层是单引号  var answer = 'He is called "Kris"'; //外层单引号，内层双引号 |

### 7.3boolean类型

布尔（逻辑）只能有两个值：true 或 false。布尔常用在条件测试中。

|  |
| --- |
| var x=true;  var y=false;  console.log(typeof x);//boolean  console.log(typeof y);//boolean |

### 7.4undefined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。

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

### 7.5null类型

null：是一个空对象指针，表示准备用来保存对象，还没有真正保存对象的值，如果定义的变量准备在将来用于保存对象，应该将该变量初始化为null。所以null值用typeof检测出来是"object"。

|  |
| --- |
| var a = null;  console.log(a);//null  console.log(typeof a);//object |

### 7.6object类型

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

|  |
| --- |
| var person={firstname:"John", lastname:"Doe", id:5566}; |

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

|  |
| --- |
| var person={  firstname:"John",  lastname:"Doe",  id:5566  }; |

### 7.7JavaScript拥有动态的数据类型

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。**

|  |
| --- |
| var a;  console.log(typeof a);//undefined  a = 100;  console.log(typeof a);//number  a = "hello";  console.log(typeof a);//string |

### 7.8typeof

这个关键字，用来检测一个变量的类型。

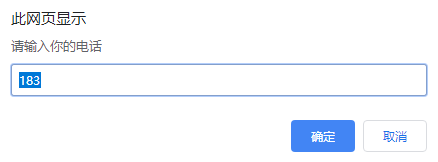
|  |
| --- |
| var a = 100;  console.log(**typeof** a);//number  var b =”100”;  console.log(**typeof** b);//string |

1. **变量类型的转换**

### 8.1string → 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：

|  |
| --- |
| prompt("请输入你的电话","183"); |



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。

|  |
| --- |
| 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：

|  |
| --- |
| var a = prompt("请输入你的电话","139");  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。

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

下面的实例结果都是123，最后一个是-123：

|  |
| --- |
| parseInt("123") //123  parseInt("123.6") //123  parseInt("123年都会很爱你") //123  parseInt("123年11月") //123  parseInt("123px") //123  parseInt("-123.99999999") //-123 |

parseInt如果不能转，那么就返回NaN

|  |
| --- |
| console.log(parseInt("Hello")); //NaN  console.log(parseInt("s546")); //NaN  console.log(parseInt("三百六十五"));//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。

|  |
| --- |
| <script type="text/javascript">  var a = "123.67.88";  var b = parseFloat(a);  console.log(b);//123.67  </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。

### 8.2number → 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。

|  |
| --- |
| var a = 123;  var b = a + "";  console.log(b);//123  console.log(typeof b);//string |

1. **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：

|  |
| --- |
| <!--我是注释--> |

CSS注释：

|  |
| --- |
| /\*我是注释\*/ |

JavaScript里面的注释：

单行注释：

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

或者多行注释：

|  |
| --- |
| **/\***  我是一行注释  再来一行  哈哈，我又写了一行  **\*/** |

注释的符号，不能嵌套，错误的示范：

|  |
| --- |
| /\*  我是一行注释  **/\***  **\*/**  再来一行  哈哈，我又写了一行  \*/ |

Webstorm中注释的快捷键，单行注释为ctrl+/，多行注释为ctrl+shift+/