# JavaScript

<https://www.jianshu.com/p/92fa46c5841a>

## 1.1JavaScript简介



概述：JavaScript在前端中简称【JS】，JavaScript是开发web脚本语言。但是它也被用到了很多非浏览器环境中【比如：node平台】，JavaScript在1995年由[Netscape](https://baike.baidu.com/item/Netscape)公司的[Brendan Eich](https://baike.baidu.com/item/Brendan%20Eich)（创始人大概用了十天的时间将这门语言创建出来），JS可以嵌套在静态页面中可以给静态页面添加一些动态效果（脚本语言），

不同浏览器厂商（在浏览器中都有内置解析器解析JS语法）；

各大浏览器厂商：chorme、IE、火狐、sofai、欧朋（都有属于自己的解析器）

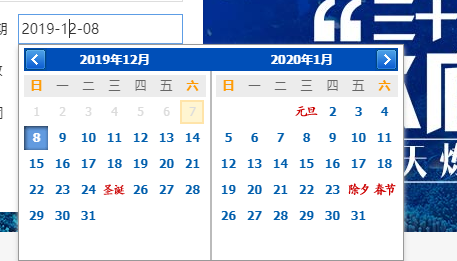
谷歌浏览器：V8解析器（可以解析JS语法：让谷歌浏览器知道你在写什么）

布兰登，艾奇



脚本语言：【PHP、JavaScript】

有一个很大特征：可以嵌套静态页面中添加一些动态效果。



## 1.2JavaScript组成

问题：前端三层？

前端分为三层：

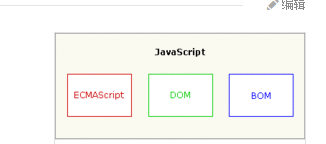
结构层 前面课程当中学习标签

样式层 前面课程我们学习样式

行为层 即将学习javaScript

问题2：JavaScript组成？

答：分为三部分



ECMAScript:欧洲计算机协会大概每年6月中旬定制语法规范；

比如：变量为什么用var、函数关键字为什么function、循环语句为什么是for

注意：咱们这段事件学习的是2014年规范简称ES5

DOM【document object model】文档对象模型（经常用来操作标签：在JS当中）

BOM【browser object model】:浏览器对象模型（模拟浏览器）

|  |  |
| --- | --- |
| ECMAScript | 欧洲计算机协会每年6月份定时语法规范 |
| DOM | Document Object Model |
| BOM | Browser Object Model |

## 1.3JS书写格式

概述：JavaScript属于脚本语言，JS语法务必、必须、一定嵌套在静态页面中，JS才可以运行；

我们可以利用双闭合标签<script></scipt>在静态页面中书写JS语法，双闭合标签scipt可以防止静态页面中任意地方，script当然也可以有多个；



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //这里面是要书写JS语法的地方

    alert("Hello word");

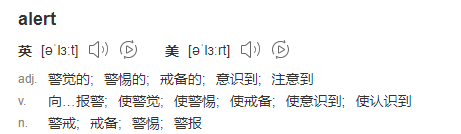
</script>

* JS语法必须放置在双闭合标签script里面
* 双闭合标签script可以放置静态页面中任意地方（一般放置静态页面底部）
* 程序当中当然可以有多个script标签

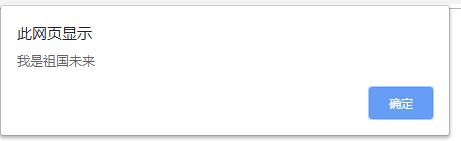
# 内置功能函数使用

内置：浏览器厂商天生本身天生就有的，我们程序员可以直接使用

## 2.1警告框-alert



概述：警告框功能是在浏览器正上方（中间）弹出一个警告框。



用法：

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //单行注释

    /\*

       多行注释：可以写多行注释给同事们看

       工作当中尽量养成这个习惯

    \*/

    //JS字符串：人说的话，务必加上双引号、单引号

    alert("我是祖国未来");

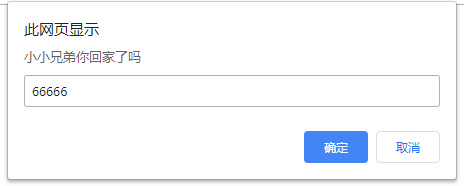
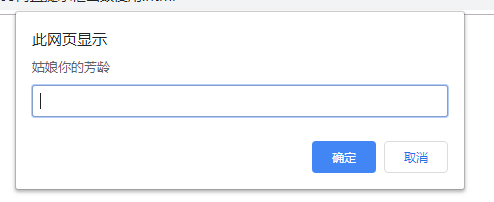
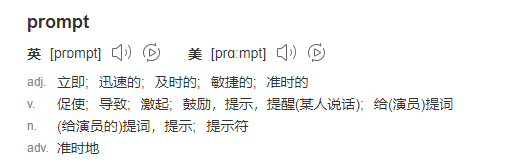
    alert("我们希望我们将来很好");

    alert('希望听贾老师的课程你们有收获');

</script>

* 警告框功能可以多次使用
* JS当中字符串，外层需要加上双引号、单引号（别一双一单）
* 每行代码最后面加上一个分号：代表的是这行语句结束

## 2.2提示框-prompt



概述：在JS当中内置函数prompt，可以在浏览器正上方弹出一个提示框；

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //提示框

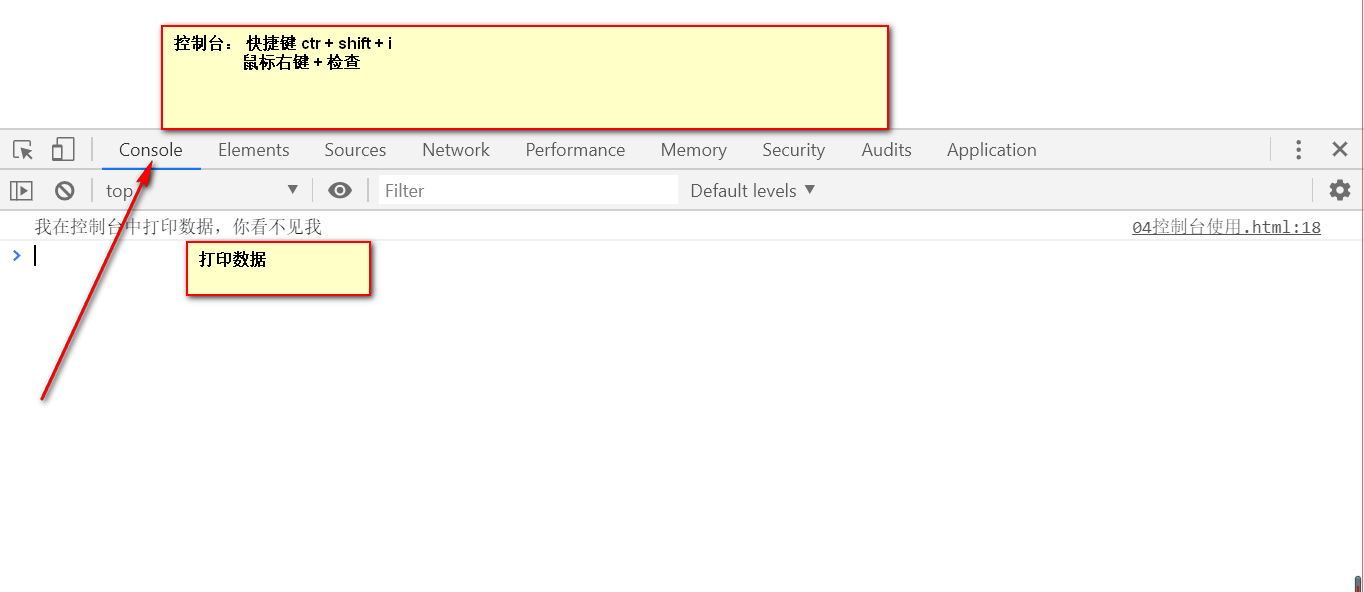
    prompt("姑娘你的芳龄");

    prompt("小小兄弟你回家了吗", 66666);

</script>

## 2.3控制台使用

概述：console.log（打印的数据）



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //控制台打印数据

    console.log("我在控制台中打印数据，你看不见我");

    console.log(12306);

</script>

# 数据类型

概述：在JS当中也有数据类型也分：基本数据类型（5） 、引用类型【复杂数据类型】（1）

//基本数据类型

|  |  |
| --- | --- |
| 数据类型 | 数值 |
| String：字符串（人说的话） | “我爱你祖国” |
| Number：数字类型 | 100，3.14，-666 |
| Boolean:布尔类型 | true、false |
| Undefined：未定义 | undefined |
| Null：空对象类型 | null |

引用类型数据：

|  |  |
| --- | --- |
| 数据类型 | 数值 |
| Object:引用类型 | 函数、数组、正则、DOM、BOM |



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //字符串（String）:字符串即为人说的话【务必加上双引号、单引号】

    console.log("我爱你JS");

    console.log("最近快过年了，回家要胖五进");

    //数字类型(Number):咱们小学学习数字

    console.log(100);

    console.log(-4.14);

    console.log(123456789);

    //布尔类型数值：true、false

    console.log(true);

    console.log(false);

    //未定类型数据

    console.log(undefined);

    //空对象类型

console.log(null);

    //JS当中有一个关键字typeof可以检测数据类型

    console.log(typeof 123);

    console.log(typeof true);

    console.log(typeof undefined);

</script>

* 在JS当中有五个基本数据类型是：字符串、数字、布尔值、未定义、空对象
* typeof是JS当中一个关键字可以检查数据类型

# 变量\*\*\*\*\*\*

概述：变量（variable）来源于数学,变量相当于是一个容器，变量可以存储数据、存储计算完结果；

通过访问变量名字获取到存储数据。

JavaScript当中变量使用：

第一步：用关键字var 声明变量 var 变量名字

第二步：给变量进行赋值 变量名字 = 123；

第三步：使用变量 console.log(变量名字)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

</head>

<body>

</body>

</html>

<script>

    //第一步：用关键字var进行声明变量

    var num;

    //第二步：可以给变量进行赋值 （将右侧数据||计算完结果）赋值给左侧变量

    num = 123;

    //第三步：使用

    //通过访问变量名字、获取到对应存储数据

    console.log(num);

</script>

* JS当中变量只是用关键字var声明但未赋值，默认初始undefined

  var abc;

    //给变量进行赋值（多次赋值）

    abc = 123;

    abc = "我爱你祖国";

    abc = "我喜欢貂蝉、吕布、鲁班";

    console.log(abc);

* JS当中变量 可以声明一次、但是可以多次赋值
* 多次赋值以后在使用：使用的是最后一次赋值数据

<script>

    var a = 100;

    var b = 200;

    var c = 300;

    console.log(a, b, c);

</script>

这种写法才是常用声明变量方式：就是将第一步和第二步结合在一起；

<https://baike.baidu.com/item/IEEE%20754/3869922?fr=aladdin>

0.1 + 0.2 ==0.3 错误的 由于遵守iEEE754浮点数标准（后面保留17位小数）

# 运算符

概述：在JS当中拥有数学运算符、比较运算符、逻辑运算符、赋值运算符等等。

运算符又称之为操作符。

## 5.1数学运算符（5）

概述：数学运算符即为： 加、减、乘、除、取余数。在JS当中任意类型数据都可以参与数学运算符。

除了数字以为其他类型数据需要隐式转换为数字参与数学运算符。

### 5.1.1数字参与

概述：如果是数字参与数学运算符：运算符口诀和小学学习是一模一样的

<script>

    //数学运算符------数字参与

    //如果是数字和数字进行数学运算符：

    //先算乘除、后算加减、如果有小括号先算小括号里面的

    console.log(66 + 22);

    console.log(66 - 33);

    console.log(22 \* 22);

    console.log(6 / 3);

    console.log(99 % 3); // 0

</script>

### 5.1.2布尔参与

概述：在JS当中布尔值类型数据也可以参与数学运算符，布尔值参与数学运算符，计算机底层自动隐式转换为数字参与。（底层完成隐式转换为起始通过系统内置函数Number完成）

布尔类型数据参与数学运算符：隐式转换为数字 true=>1 false=>0

    console.log(Number(true));//1

    console.log(Number(false));//0

    console.log(true + 99);//100

    console.log(false \* 100);//0

### 5.1.3未定义类型参数

概述：未定义类型是数值也可以参与数学运算符，需要隐式转换为数字NaN【not a number】

NaN:是JS语言当中一个数字类型特殊值，一般进行数学运算符的时候，计算不出结果，返回数字NaN；

    //在JS当中未定义类型数据undefined也可以参数学运算符：

    //但是要隐式转换为数字参与

    //undefined未定义类型数据参与数学运算符的时候会隐式转换为数字NaN

    /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

    //NaN起始也是数字类型数值：值不够这个数值比较特殊，一般在计算数学运算符的时候算不出结果的时候

    //返回数字NaN

    console.log(0 / 0);

    console.log(typeof NaN);

    console.log(NaN + 33);

    console.log(NaN \* 33);

    console.log(NaN + NaN);

    console.log(undefined + 33);

    console.log(undefined / 33);

* 总结：未定义类型数据也可以参数数学运算符，但是需要隐式转换为数字NaN参与
* NaN是数字类型一个特殊值：这个数字进行数学运算的时候都是计算不出结果（NaN）

### 5.1.4空对象类参与

概述：空对象类型数据null，也可以参与数学运算符，但是需要隐式转换为数字参与数学运算符；

    //Null类型数值参与数学运算符

    console.log(Number(null));

    console.log(null \* 99);

    console.log(0 + NaN);

    console.log(null + NaN);

### 5.1.5小总结

概述：任意类型数据都可以参与数学运算符：如果不是数字类型的，隐式转换（浏览器自动）为数字参与；

|  |  |
| --- | --- |
| 布尔值 | true=>1 false=>0 |
| 未定义类型 | undefined=>NaN |
| 空对象类型 | null=>0 |

### 5.1.6字符串参与\*\*\*\*\*\*\*

概述:字符串也可以参与数学运算符，但是它有一些特殊：

特殊：字符串参与数学运算符的时候，加号比较特殊；

   console.log("12" + 44);

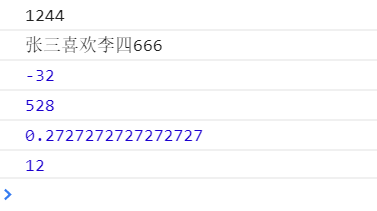
    console.log("张三喜欢李四" + 666);

    console.log("12" - 44);

    console.log("12" \* 44);

    console.log("12" / 44);

console.log("12" % 44);



注意1：

字符串也可以参数数学运算符：但加号比较特殊，

如果加号遇见字符串称之为连字符：将数据从左到右拼接为字符串

如果减、乘、除、取余数字符串参与：都隐式转换为数字参与。

<script>

    //字符串参与

    console.log("张三" + 66);

    console.log("张三" + "李四");

    //如果是字符串参与数学运算符：减、乘、除、取余数

    console.log("小明" / 66);

    console.log("小红红" \* "小兰兰");

</script>

## 5.2比较运算符（8）

概述：任意类型数据都可以参与比较运算符，比较最终结果：要么为布尔值true、要么为布尔值false

除了数字以外其他类型数据也可以参与比较运算符：但是需要隐式转换为数字参与（字符串特殊）

|  |  |
| --- | --- |
| > | 大于 |
| >= | 大于等于 |
| < | 小于 |
| <= | 小于等于 |
| == | 相等判断（就看数值是否一样） |
| != | 不等 |
| === | 全等（看数值+看数据类型） |
| !== | 不全等 |

### 5.2.1数字比较

    //数字参与比较运算符

    console.log(3 > 6); //false

    console.log(3 >= 6); //false

    console.log(3 < 6); //true

    console.log(3 <= 6); //true

    console.log(3 == 3); //true

    console.log(3 != 3); //false

    console.log(3 === "3"); //false

    console.log(3 !== 3); //false

注意1：

一个等号：赋值运算符 不是进行比较运算符（常用语给变量赋值）

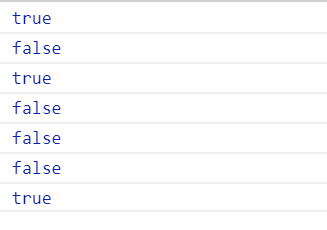
两个等号：相等比较运算符 (只看数值是否一样)

三个等号：全等比较 （即看数值又看类型比较）

### 5.2.2其他类型数据比较（字符串除外）

概述：其他类型数据也可以参与比较（数字、字符串除去），其他类型数据也可以参与比较运算符，

但是需要隐式转换为数字参与。



  console.log(100 > true);

    console.log(true === 1);

    console.log(true == 1);

    console.log(true < false);

    console.log(NaN > 100);

    console.log(undefined > 100);

    console.log(6666 > null);

### 5.2.3字符串比较（奇葩）

概述：字符串参与比较时候分为两种情况：

第一种：字符串和数字

第二种：字符串和字符串比较

#### 5.2.3.1字符串和数字比较

概述：如果是字符窜和数字比较：字符串隐式转换为数字参与比较；

<script>

    //字符串和数字比较

    console.log("66" > 99); //false

    console.log("33" > 12); //true

    console.log("33" == 33); //true

    console.log("33" === 33); //false

</script>

#### 5.2.3.2字符串和字符串进行比较

<https://baike.baidu.com/item/ASCII/309296?fr=aladdin> ASCII

概述：如果是字符串和字符串进行比较：通过的是ASCII码数值进行比较：

两个字符串都是从左到右有一个一个字符进行比较：知道某一个字符比较出结果，比较结束了；

0-9<A-Z<a-z:从左到右依次变大

    //字符串和字符串比较

    console.log("a2" > "A2"); //true

    console.log("b2a" > "ABC"); //true

    console.log("李四" == "李四");//true