# JavaScript基础语法

## JS语句

对于浏览器而言，语句就是命令，它告诉浏览器要做什么

**如：document.write("<p>Hello!</p>");**

语句通常以分号结束

**建议使用分号；**

一系列能被浏览器执行的语句构成JS程序

·语句语法

（1）每一条语句都要以;结束

（2）一条语句必须在同一行内，不能换行

（3）每一条语句都要独占一行（编码规范）

（4）JS语句是JavaScript程序的基本组成单位

## JS语句块

（1）一组相关的代码的集合

（2）用{ }括起来的一些语句组成语句块

1. 用在流程控制、函数或对象中

（4）JavaScript代码的执行次序与书写次序相同

<script type="text/javascript">

function test() {

alert('123');

alert('JS');

}

## JS注释

程序中不会被认为代码执行的内容，通常用于对代码进行解释提高代码可读性

·单行注释 //

·多行注释 /\* \*/

**注意 /\*alert('123');/\*123\*/alert('Js');\*/不能讲两个alert进行注释掉。**

# JavaScript变量及原始数据类型

## 变量的创建和说明

**什么是变量，变量的作用？**

·程序：程序是为解决一个信息处理任务而预先编制的工作执行方案。

·变量：指向内存存储区域的标识符，是存储信息的容器

（1）内存中存储运算数据

（2）变量指向内存中的存储空间

（3）程序中通过变量对数据进行操作

.JS变量的用法

**/\* 声明（创建）变量 \*/**

·使用var关键词 var 变量的名字 = '初识值'

通常变量声明的时候同时赋予初值。

**/\* 变量声明 \*/**

·变量名称中可以包含：数字、字母、\_(下划线)和$

·变量可以以字母、\_(下划线)和$开头（一般以字母开头，也可以用\_（下划线）和$开头但是w3c并不建议这么做）

·变量名称对大小敏感，区分大小写

## 原始数据类型

**什么是数据类型**

（1）程序处理的数据有多种类型

（2）不同的数据类型在内存中存储的大小、区域不同

（3）不同类型的数据具有不同的处理方式

**判断数据类型**

·**使用typeof运算符**

**JS中的数据类型**

·**Number（数值）类型**：表示整数和浮点数类型

·**String（字符串）类型**：表示0个或多个字符组成的字符序列

字符串直接量使用一对单引号或者一对双引号括起来（其中单引号和双引号没有区别）

·**Boolean（布尔）类型**：表示逻辑真假，有true和false两个值

<script type="text/javascript">

var isLogin = false;

alert(typeof isLogin);

</script>

·**Object（对象）类型**：是一种复合数据类型，一些基本数据类型的集合

<script type="text/javascript">

var person = {

name : '张三',

age : 18

}

alert(typeof person);

</script>

·**Undefined**：变量未定义或未被初始化值时，该变量为undefined类型，该类型只有一个取值，即特殊的undefined。

<script type="text/javascript">

var message;

alert(message);//取值为undefined

alert(typeof message);//undefined数据类型

</script>

**·null类型**：表示空对象，该类型只有一个取值，即特殊的null。

<script type="text/javascript">

**通常删除变量的方法是将变量值设置成null**

var obj = null;

alert(obj);//值为null

alert(typeof obj);//这是object类型

</script>

**JS数据类型的特征**

（1）变量声明的时候不指定数据类型

（2）变量在后续程序中可以改变内部存储的数据类型

# 运算符

## 算术运算符

·**加法运算符:+(一方名可以用于数值的相加，另一方面可以用于字符串的连接)**

var box = 1 + 2; //等于 3

var box = 1 + NaN; //NaN，只要有一个 NaN 就为 NaN

var box = 100 + '100'; //100100，字符串连接符，有字符串就不是加法

var box = '您的年龄是：' + 10 + 20; //您的年龄是：1020，被转换成字符串

var box = 10 + 20 + '是您的年龄'; //30 是您的年龄，没有被转成字符串（原因是因为加法运算是从左至右进行运算的）

var box = '您的年龄是：' + (10 + 20); //您的年龄是：30，没有被转成字符串（优先级）

·**减法运算符:-（整数相除可以得到小数）**

var box = 100 - 70; //等于 30

var box = 1 - NaN; //NaN，只要有一个 NaN 就为 NaN

var box = 100 - true; //99，true 转成数值为 1，false会转换为数值0

var box = 100 - ''; //100，''转成了 0

var box = 100 - '70'; //30，'70'转成了数值 70

var box = 100 - null; //100，null 转成了 0

var box = 100 - 'Lee'; //NaN，Lee 转成了 NaN

**·乘法运算符:\*（规则和减法运算符相同）**

**·除法运算符:/ （整数相除可以得到小数）**

**·余数运算符:%**

var box = 10 % 3; //1，余数为 1

var box = 100 % NaN; //NaN

var box = 100 % true; //0

var box = 100 % ''; //NaN

var box = 100 % null; //NaN

var box = 100 % 'Lee'; //NaN

**·自增运算符:++（用法和C语言的相同，注意前置和后置的区别）**

**·自减运算符:--**

其他类型应用一元运算符的规则

var box = '89'; box++; //90，数值字符串自动转换成数值

var box = 'ab'; box++; //NaN，字符串包含非数值转成 NaN

var box = false; box++; //1，false 转成数值是 0，累加就是 1

var box = 2.3; box++; //3.3，直接加 1

**·求负运算符:-**

减运算规则如下：

var box = 100; -box; //-100，对于数值，直接变负

var box = '89'; -box; //-89，数值字符串转换成数值

var box = 'ab'; -box; //NaN，字符串包含非数值转成 NaN

var box = false; -box; //0，布尔值转换成相应数值

var box = 2.3; -box; //-2.3，没有变化

## 关系运算符

比较两个值，然后返回**一个布尔值**，表示是否满足比较条件

·==相等

·===全等

·!=不相等

·!==全不相等

和其他运算符一样，当**关系运算符操作非数值时**要遵循一下规则：

**1.两个操作数都是数值，则数值比较；**

**2.两个操作数都是字符串，则比较两个字符串对应的字符编码值；**

**3.两个操作数有一个是数值，则将另一个转换为数值，再进行数值比较；**

var box = 3 > 2; //true

var box = 3 > 22; //false

var box = '3' > 22; //false

var box = '3' > '22'; //true （按照字符编码值从左至右进行比较）

var box = 'a' > 'b'; //false a=97,b=98

var box = 'a' > 'B'; //true B=66

·<小于

·<=小于等于

·>大于

·>=大于等于

**相等和全等运算符区别：**

**（1）相等运算符：先进行数据类型的转换，之后再进行值比较**

**（2）全等运算符：不进行类型转换，类型不同，则直接返回false。**

****

## 逻辑运算符

·非运算符：！

逻辑非运算符可以用于任何值。无论这个值是什么数据类型，**这个运算符都会返回一个布尔值。**它的流程是：先将这个值转换成布尔值，然后取反，规则如下：

var box = !(5 > 4); //false

var box = !''; //true

var box = !'Lee'; //false

var box = !0; //true

var box = !8; //false

var box = !null; //true

var box = !NaN; //true

var box = !undefined; //true

使用一次逻辑非运算符，流程是将值转成布尔值然后取反。

而使用两次逻辑非运算符就是将值转成成布尔值取反再取反，**相当于对值进行 Boolean()转型函数处理**。

var box = !!0; //false

var box = !!NaN; //false

·或运算符：||

如果两边的操作数有一个操作数不是布尔值的情况下，逻辑与运算就**不一定返回布尔值**，此时，遵循已下规则：

**若两个值都为真，或者第一个为真的时候，返回结果就是第一个操作数**

和逻辑与运算符相似，逻辑或运算符也是短路操作。当第一操作数的求值结果为 true ，

就不会对第二个操作数求值了。

var box = true || age; //true

var box = false || age; //出错，age 未定义

·与运算符：&&

如果两边的操作数有一个操作数不是布尔值的情况下，**与运算就不一定返回布尔值**，此时，遵循已下规则：

**若两个值都为真，返回结果就是第二个操作数**

逻辑与运算符属于短路操作，顾名思义，如果第一个操作数返回是 false，第二个数不管是 true 还是 false 都返回的 false。

var box = true && age; //出错，age 未定义

var box = false && age; //false，不执行 age 了

## 赋值运算符

（用于给变量赋值，使用 = ）

<script type="text/javascript">

var x = 10;

y = x;

alert(y);//此时不用声明变量吗？

x += y;//等同于x = x + y

x -= y;//等同于x = x - y

x \*= y;//等同于x = x \* y

x /= y;//等同于x = x / y

x %= y;//等同于x = x % y

x >>= y;//等同于x = x >> y

x <<= y;//等同于x = x << y

x >>>= y;//等同于x = x >>> y

x &= y;//等同于x = x & y

x |= y;//等同于x = x | y

x ^= y;//等同于x = x ^ y

</script>

## 三元运算符

三元运算符：?:（语法和C语言的相同）

# 数据类型转换

## 隐式转换

**转换成 String 类型：用 + 连接**

如：var sum =‘img’+ 3 + ’.jpg’;

**转换成 Boolean 类型：变量之前加 !!**

**+左右出现字符串时，作为字符串连接运算符使用**

**- 、\*、/、%左右出现字符串（布尔）时，将字符串（布尔）转换为数值类型**

**比较运算符左右出现数值和字符串时，会将字符串转换为数值，出现布尔类型时，会将布尔类型转换为数值类型**

## 显式转换

**转换为数值类型**

**转换方法（使用number函数、使用parseInt函数、使用parseFloat函数）**

从位置 0 开始查看每个字符，直到找到第一个非有效的字符为止，然后把该字符之前的字符串转换成整数。

但对**parseFloat()**来说，第一个出现的小数点是有效字符。如果有两个小数点，第二个小数点将被看作无效的。

<script type="text/javascript">

Number("324");//324,转换为number类型

Number("324 mins");//NaN(Not a Number)

Number("");//0

Number(false);//0

Number(true);//1

Number(undefined);//NaN

Number(null);//0

parseInt("42.cats");//42以数值开始会转换成整型数值

parseInt("some 42 cats");//NaN

parseFloat("43.12cats");//43.21

</script>

**转换成字符串类型**

转换方法（使用String函数）

<script type="text/javascript">

String(123);//"123"

String("123");//"123"

String(true);//"true"

String(false);//"false"

String(undefined);//"undefined"

String(null);//"null"

</script>

**转换成布尔类型**

转换方法（使用Boolean函数）

<script type="text/javascript">

Boolean(undefined);//false

Boolean(null);//false

Boolean(0);//false

Boolean(NaN);//false

Boolean("");//false

Boolean(" ");//空格所以结果为true

</script>

转换规则：以下六个值转换为false、undefined、null、false、0、NaN、空字符串，其余的转换为true。

# JavaScript流程控制结构

## 选择和分支语句

*if 语句* - 只有当指定条件为 true 时，使用该语句来执行代码

*if...else 语句* - 当条件为 true 时执行代码，当条件为 false 时执行其他代码

*if...else if....else 语句* - 使用该语句来选择多个代码块之一来执行 （一个执行，其他的就不再进行执行了）

*switch 语句* - 使用该语句来选择多个代码块之一来执行（用法同C语言）

**对于 if 语句括号里的表达式，ECMAScript 会自动调用 Boolean()转型函数将这个表达式的结果转换成一个布尔值。如果值为 true，执行后面的一条语句，否则不执行。**

PS：if 语句括号里的表达式如果为 true，只会执行后面一条语句，如果有多条语句，那么就必须使用复合语句把多条语句包含在内。

PS2：推荐使用第一种或者第三种格式，一行的 if 语句，或者多行的 if 复合语句。这样就不会因为多条语句而造成混乱。

PS3：复合语句我们一般喜欢称作为：代码块。

## 循环语句

**do...while 语句**

do...while 语句是一种先运行，后判断的循环语句。也就是说，不管条件是否满足，**至少先运行一次循环体。**

var box = 1; //如果是 1，执行五次，如果是 10，执行 1 次

do {

alert(box);

box++;

} while (box <= 5); //先运行一次，再判断

**while 语句**

while 语句是一种先判断，后运行的循环语句。也就是说，必须满足条件了之后，方可运行循环体。

var box = 1; //如果是 1，执行五次，如果是 10，不执行

while (box <= 5) { //先判断，再执行

alert(box);

box++;

}

**for 语句**

for 语句也是一种先判断，后运行的循环语句。但它具有在执行循环之前初始变量和定义循环后要执行代码的能力。

for (var box = 1; box <= 5 ; box++) { //第一步，声明变量 var box = 1;

alert(box); //第二步，判断 box <=5

} //第三步，alert(box)

//第四步，box++

//第五步，从第二步再来，直到判断为 false

## break和continue语句

**break 和 continue 语句用于在循环中精确地控制代码的执行。其中，break 语句会立即退出循环，强制继续执行循环体后面的语句。而 continue 语句退出当前循环，继续后面的循环。**

for (var box = 1; box <= 10; box++) {

if (box == 5) break; //如果 box 是 5，就退出循环

document.write(box);

document.write('<br />');

}

for (var box = 1; box <= 10; box++) {

if (box == 5) continue; //如果 box 是 5，就退出当前循环

document.write(box);

document.write('<br />');

}

# JavaScript的基本代码规范

## 代码规范的重要性：

（1）方便代码的交流和维护。

（2）不影响编码的效率，不与大众习惯冲突。

（3）使代码更美观、阅读更方便。

（4）使代码的逻辑更清晰、更易于理解。

## JavaScript基本规范

* 变量定义
  + 尽量使用 var 关键字定义（否则会被当成全局变量）
  + 尽量减少全局变量的使用
  + 变量名定义要有意义
* 单行程序，以分号结束
* 缩进和注释

# 调试工具的使用

1.代码编辑工具

·文本编辑器：用于编辑纯文本内容的工具 例如：记事本，sublime text 等

·IDE：集成开发环境，整体提供：编辑器、编译器、调试器、图形界面工具 例如：VS2012

2.代码调试运行工具

·运行工具：浏览器（使用浏览器打开页面文件即可运行程序）

·调试工具：浏览器开发者工具

（1）谷歌：谷歌开发者工具

（2）火狐：火狐开发者工具、firebug插件

·使用

（1）打开：使用F12

（2）查看页面源代码

（3）使用控制台 查看错误 控制台输出 console.log()

<script type="text/javascript">

var a='123';

console.log(a);

</script>