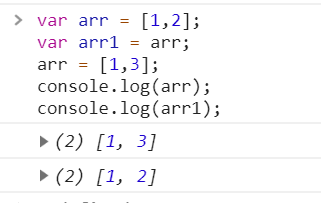
1. background是复合属性，必须写在相关的背景属性的最前面，否则没有效果。例如：有一个background样式，还有一个background-size样式，则background必须写在background-size的前面。
2. getElementByClassName获得的是一个数组，视情况而定在后面加[0]。
3. 三列布局的时候，中间的布局和样式必须写在右边样式和布局之后，否则会导致右边的元素挤到下一行。例如：left、middle、right定义三个div的class，则写布局和class样式的时候，顺序为：.left .right .middle。middle在布局和样式的时候必须放在最后。



数组是引用值，arr指向一个新的堆队列，所以arr输出1，3.但是arr1还是之前的arr赋值得到的是1，2.

1. var a = 10;

var b = ++a – 1 + a ++;

console.log(b);

console.log(a);

输出的结果b = 21;

输出结果a = 12；

b的计算过程中a第一次已经变成了11，所以a++的时候a已经是11，之后输出的a为12，a++是先输出再加，所以得到的b的值为11 – 1 + 11 = 21。

1. a = 1;

b = 3;

b %= a + 3;

输出a的值为1

输出b的值为3，先计算a+3的结果，再计算b%=4的值。赋值符号的运算级别是最低的。所以先计算+运算符得到的值，再做取模运算。运算是自左向右。

a = 1 + 1 + ‘a’ + 1 ,得到的是2a1

a = 1 + ‘a’+ 1 + 1,得到的是1a11

1. a = ‘10’ > ‘8’输出的结果得到的是false

因为这是字符串比较，会根据asc码来进行比较，并且从第一位开始比较，如果第一位就比较出结果，则就是第一次比较出的结果，案例中1的asc码小于8的asc码，所以是false。

1. 计算机中infinity是固定值，所以infinity == infinity返回的boolean是true，然而NaN == NaN ,返回的布尔值是false，因为NaN不等于任何值，包括它本身。
2. for循环的执行顺序：

for（var i = 一个值； i < n; i ++）{

//循环体；

}

第一步：先定义一个变量i的值。

第二步：判断条件是否满足，如果条件满足，则执行循环体的语句。如果条件不满足，报错或者跳出循环。

第三步：如果循环的条件满足，即i < n，则执行循环体的语句之后，再执行 i ++；i ++是for循环内最后执行的一个语句。

1. js预编译知识：

function bar() {

return foo;

foo = 10;

function foo(){

}

var foo = 11;

}

console.log(bar());

打印输出的结果是foo（）{}函数

return 函数表示结束函数输出值，放在函数的最前面就直接输出函数。

1. js预编译知识

console.log(bar());

function bar() {

foo = 10;

function foo(){

}

var foo = 11;

return foo;

}

此处输出的结果为11，AO中的foo由undefined变为function，但是由于执行顺序的原因，foo又被定义为变量等于11，所以输出的foo等于11.

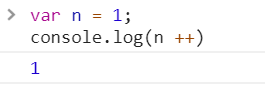
1. 注意：var b = function(){};在程序中并不代表定义一个函数：如果

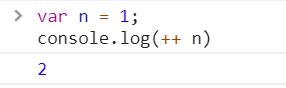
console.log(b);

var b = fuction (){};

此时输出的console结果是：undefined，因为此时看的是变量b，要找变量和实参，所以只有一个变量b，但是没有定义的值。

1. n ++ 和 ++ n

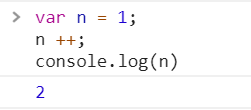




第一次使用n++是先输出变量的值，再进行加法运算。

第一次使用++n是先对变量进行加法运算，再输出。

但是，注意的点：



此时console的n是已经执行过一次n++，即已经输出过一次n = 1，再执行加之后，此时的n = n + 1；