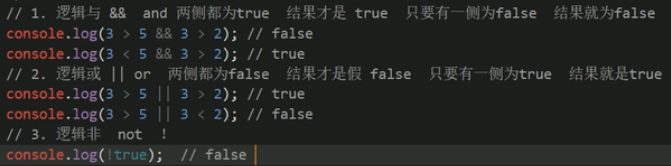
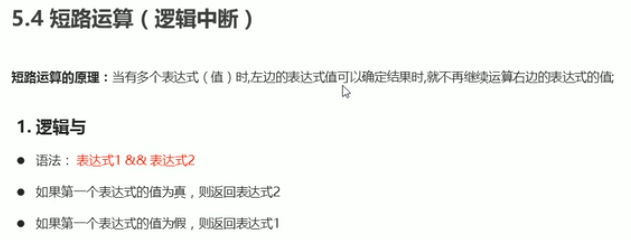
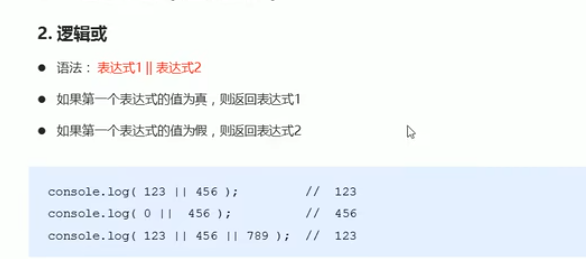


每一个表达式都会返回一个值

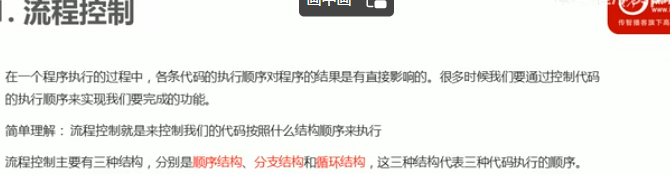


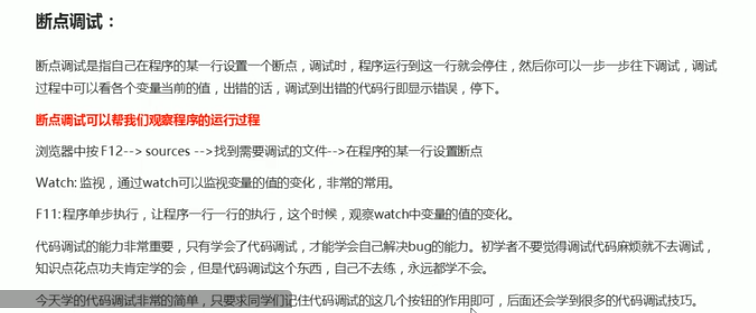












1. 字符串,布尔值 和数字进行比较时,会先隐式的转化为数字类型,之后再比较

console.log("5" > 3); // true

console.log(true == 1); // true

console.log(false == 0); // true

console.log(true == 2); // false

2.字符串 和 字符串比较时,按照从左到右的顺序 比较 字符在码表(ASCII)中的大小

console.log(5 > 12); //false

console.log("5" > 12); //false

console.log("5" > "12"); //true

console.log("15" > "12"); //true

console.log("1" > "12"); //false

3. null 和 undefined 在和数字比较时

3.1 如果遇到 == 不会进行在转化,没有可比性(false)

console.log(null == 0); // false

console.log(null == 1);// false

console.log(null == -1);// false

3.2 如果遇到 > >= < <= 会隐式的转化为数字(null转化为 0 ,undefined 转化为 NaN),可以进行比较

console.log(null >= 0); // true

console.log(null <= 0); // true

console.log(null > -5); // true

因为undefined转化为NaN 所以undefined的判断都是false

console.log(undefined == 0); // false

console.log(undefined >= 0); // false

console.log(undefined <= 0); // false

4. null 和 undefined 在数值上是相等的(官方规定),类型上不相同

var a; // undefined

var b = null;

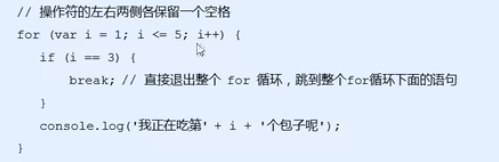
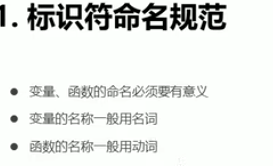
console.log(a == b); //undefiend == null => true

console.log(a === b); //undefiend === null => false

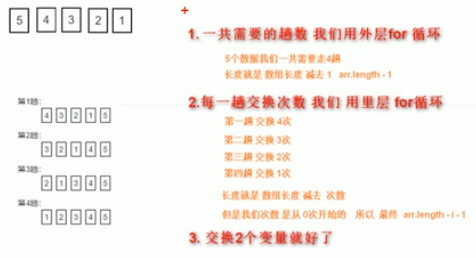
5. NaN 和 任何值 都不相等

console.log(NaN == NaN); //false

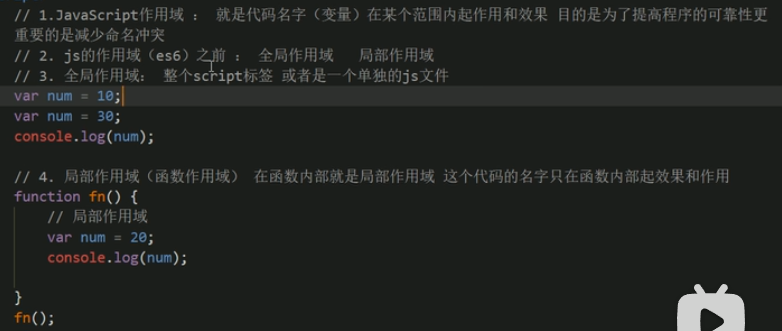
console.log("NaN" == "NaN"); //true

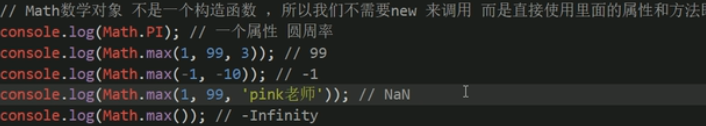
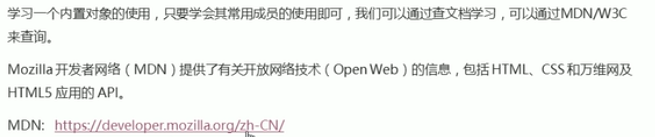
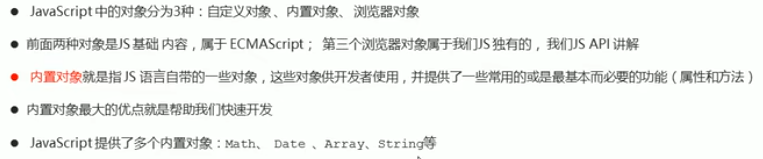
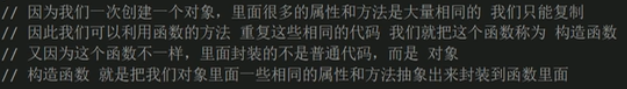
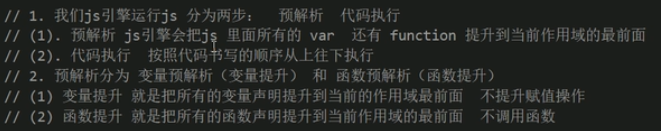
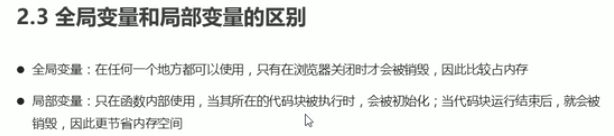
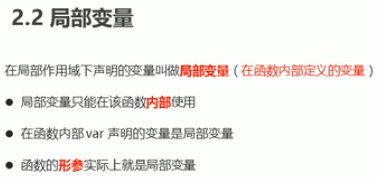
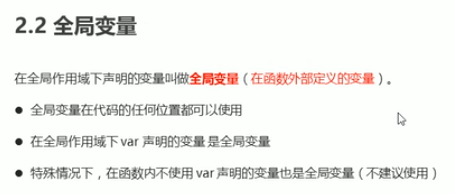


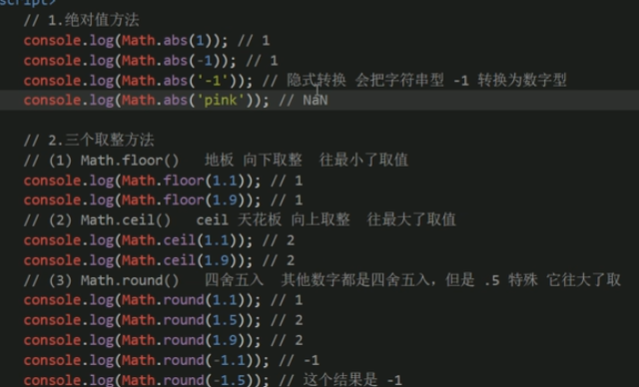




算法就是找出变化，总结规律







　在讲循环的之前，先知道一下循环结构的执行步骤

　　1、声明循环变量；

　　2、判断循环条件;

　　3、执行循环体操作；

　　4、更新循环变量；

　　5、然后循环执行2-4，直到条件不成立，跳出循环。

一、while循环

var num = 1;//1、声明循环变量

while (num<=10){//2、判断循环条件;

document.write(num+"<br />");//3、执行循环体操作；

num++;//4、更新循环变量；

}

Ps： while循环()中的表达式，运算结果可以是各种类型，但是最终都会转为真假，转换规则如下。

　　①Boolean：true为真，false为假；

　　②String：空字符串为假，所有非空字符串为真；

　　③Number：0为假，一切非0数字为真；

　　④null/Undefined/NaN:全为假；

　　⑤Object：全为真。

二、.do-while循环

　　while循环特点：先判断后执行；

　　do-while循环特点：先执行再判断，即使初始条件不成立，do-while循环至少执行一次

var num = 10;

do{

document.write(num+"<br />");//10 9 8 7 6 5 4 3 2 1 0

num--;

}while(num>=0);

document.write(num);//-1

三、for循环

　for循环

　　1、for有三个表达式：①声明循环变量；②判断循环条件；③更新循环变量；

　　　　三个表达式之间，用;分割，for循环三个表达式都可以省略，但是两个“;”缺一不可。

　　2、for循环的执行特点：先判断再执行，与while相同

　　3、for循环三个表达式都可以有多部分组成，第二部分多个判断条件用&& ||连接，第一三部分用逗号分割；

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

document.write(num+" <br />"); //1 2 3 4 5 6 7 8 9 10

}

四、for-in循环

　for-in 循环主要用于遍历对象

　　for()中的格式：for(keys in zhangsan){}

　　keys表示obj对象的每一个键值对的键！！所有循环中，需要使用obj[keys]来取到每一个值！！！

　　for-in 循环，遍历时不仅能读取对象自身上面的成员属性，也能延续原型链遍历出对象的原型属性

　　所以，可以使用hasOwnProperty判断一个属性是不是对象自身上的属性。

　　obj.hasOwnProperty(keys)==true 表示这个属性是对象的成员属性，而不是原先属性

//声明一个Peson类

function Person(){

this.name = "张三";

this.age = 14;

this.func1 = function(){

}

}

//实例化这个类

var zhangsan = new Person();

//使用for-in遍历这个对象

for(keys in zhangsan){

console.log(zhangsan[keys])

}

五、for-of循环

　　ES6 借鉴 C++、Java、C# 和 Python 语言，引入了for...of循环，作为遍历所有数据结构的统一的方法。

　　一个数据结构只要部署了Symbol.iterator属性，就被视为具有iterator接口，就可以用for...of循环遍历它的成员。也就是说，for...of循环内部调用的是数据结构的Symbol.iterator方法。

　　for...of循环可以使用的范围包括数组、Set 和 Map 结构、某些类似数组的对象（比如arguments对象、DOM NodeList 对象）、后文的 Generator 对象，以及字符串。

下面是一些数据结构的遍历

①数组

JavaScript 原有的for...in循环，只能获得对象的键名，不能直接获取键值。ES6 提供for...of循环，允许遍历获得键值。

var arr = ['a', 'b', 'c', 'd'];

for (let a in arr) {

console.log(a); // 0 1 2 3

}

for (let a of arr) {

console.log(a); // a b c d

}

上面代码表明，for...in循环读取键名，for...of循环读取键值。如果要通过for...of循环，获取数组的索引，可以借助数组实例的entries方法和keys方法

②Set 和 Map 结构

var engines = new Set(["Gecko", "Trident", "Webkit", "Webkit"]);

for (var e of engines) {

console.log(e);

}

// Gecko Trident Webkit

var es6 = new Map();

es6.set("edition", 6);

es6.set("committee", "TC39");

es6.set("standard", "ECMA-262");

for (var [name, value] of es6) {

console.log(name + ": " + value);

}

// edition: 6

// committee: TC39

// standard: ECMA-262

上面代码演示了如何遍历 Set 结构和 Map 结构。值得注意的地方有两个，首先，遍历的顺序是按照各个成员被添加进数据结构的顺序。其次，Set 结构遍历时，返回的是一个值，而 Map 结构遍历时，返回的是一个数组，该数组的两个成员分别为当前 Map 成员的键名和键值。

③类似数组的对象

类似数组的对象包括好几类。下面是for...of循环用于字符串、DOM NodeList 对象、arguments对象的例子。

// 字符串

var str = "hello";

for (let s of str) {

console.log(s); // h e l l o

}

// DOM NodeList对象

let paras = document.querySelectorAll("p");

for (let p of paras) {

p.classList.add("test");

}

// arguments对象

function printArgs() {

for (let x of arguments) {

console.log(x);

}

}

printArgs('a', 'b');// 'a' 'b'

　　与其他遍历语法的比较

for...in循环有几个缺点

　　①数组的键名是数字，但是for...in循环是以字符串作为键名“0”、“1”、“2”等等。

　　②for...in循环不仅遍历数字键名，还会遍历手动添加的其他键，甚至包括原型链上的键。

　　③某些情况下，for...in循环会以任意顺序遍历键名。

　　for...in循环主要是为遍历对象而设计的，不适用于遍历数组。

for...of循环

　　有着同for...in一样的简洁语法，但是没有for...in那些缺点。

　　不同于forEach方法，它可以与break、continue和return配合使用。

　　提供了遍历所有数据结构的统一操作接口。

六、循环控制语句

　　1、break：跳出本层循环，继续执行循环后面的语句。

　　　　如果循环有多层，则break只能跳出一层。

　　2、continue：跳过本次循环剩余的代码，继续执行下一次循环。

　　　　①对与for循环，continue之后执行的语句，是循环变量更新语句i++；

　　　　②对于while、do-while循环，continue之后执行的语句，是循环条件判断；

　　　　因此，使用这两个循环时，必须将continue放到i++之后使用，否则，continue将跳过i++进入死循环。

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

if(i == 4){

continue;

}

console.log(n);//1 2 3 5 6 7 8 9

}

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

if(i == 4){

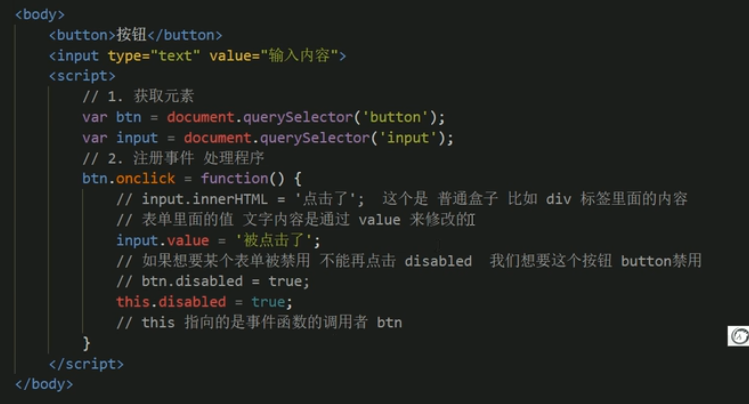
break;

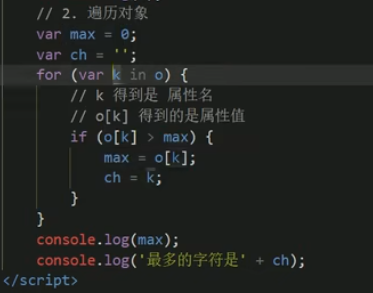
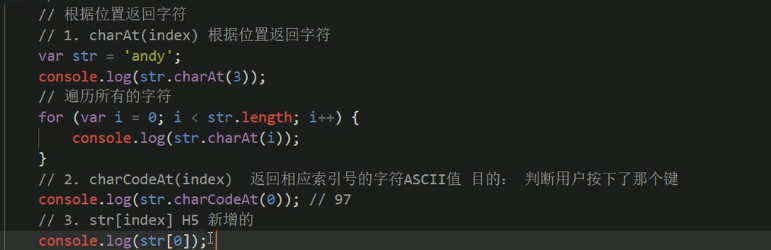
}

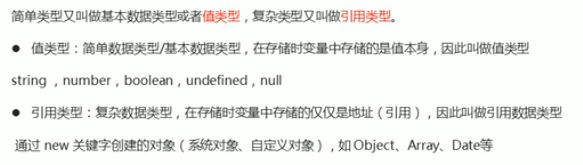
console.log(i);//1 2 3

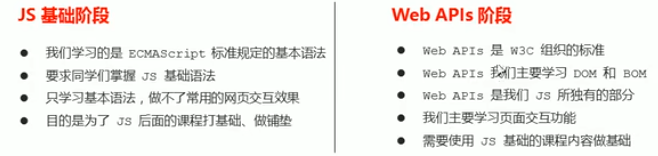
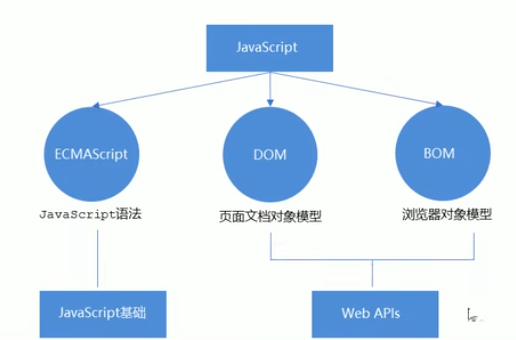
}

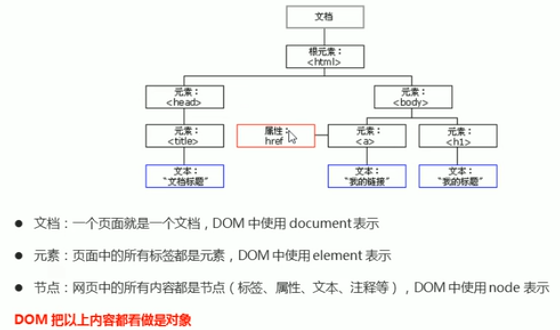
Jquery部分









获取元素 事件基础 操作元素 节点操作

