JavaScript支持函数式编程和面向对象编程。

程序是一系列语句的集合。表达式产生值。表达式通常是函数的参数，或者赋值的右部分。

JS中分号可以省略，用于结束语句块。特例：函数表达式作为一个表达式。这样的表达式出现在语句最后，它的后面会跟上一个分号。例：

var f=function(){};

单行注释//，多行注释/\* \*/

复合赋值运算符：例如：x+=1;等价于x=x+1;

标识符：变量的名字就是一个标识符。标识符区分大小写。标识符的第一个字符可以是任意的字符、美元符、下划线。

保留字不能作为变量名字使用，Infinity、NAN、undefined,标准的全局变量名都不能作为局部变量名

JS所有的值都有属性。每一个属性都有一个key(或name)和一个value。通过.可以读取和赋值属性。.还可以调用方法。

JS原始值（数据类型）：Boolean、Number、String、Null、Undefined

原始值特点：1>按值进行比较2>属性不能改变。

对象的特点：1>按引用进行比较2>属性可变

undefined：未被初始化（赋值）的变量，丢失的参数，访问不存在的属性例如：function f(x){return x} f() undefined

null:没有对象。在用到对象的时候表示空值。例如：参数、对象链中的最后一个元素等。

undefined和null没有属性，toString()方法也没有。

通常，函数允许透过undefined或null来表示缺失的值。

例：if（x===undefined || x===null){

}//undefined和null也可被视为false if(!x){}

判断数据类型：typeof主要用于原始值，instanceof用于对象值。

typeof一个函数，结果为function，typeof null 返回结果为null

instanceof用法：value instanceof Constr。如果value是一个通过Constr构造器创建的对象，则返回true。

var b=new Bar();b instanceof Bar; true

布尔值为false:undefined null -0 NaN ""

&&第一个值为假值，返回它，否则返回第二个运算数。 123 &&"abc" "abc"

||第一个值为真值，返回它，否则返回第二个运算数。"" || 123 123

JS中所有的数字是浮点数。Infinity多数情况下是一个错误的值。Infinity比任何一个数都大（NaN除外），-Infinity比任何一个数都小（NaN除外）。

字符串可以直接通过字符串字面量来创建。\用于转义字符及产生一些控制字符。

for循环：初始化会在循环开始之前执行。条件会在每次循环迭代之气做检查。后迭代会在每次循环迭代后执行。

do-while迭代体至少执行一次。break跳出本次循环，continue终止循环。

JS中，函数的所有参数可以被自由调用。arguments可使所有参数可用。看起来是数组，但不具备数组方法。

函数传参时，多余的参数被忽略，缺失的参数值为undefined。

将arguments转换为数组

function toArray(arrayLikeObject){

return Array.prototype.slice.call(arrayLikeObject);

}

激活严格模式（使JS更加干净）：

1><script> 'use strict'</script>

2>每个函数中激活严格模式

function functionInStrictMode(){

'use strict';

}

IIFE：立即调用函数表达式。造成无意共享。

若方法中有嵌套函数，嵌套函数内部不能访问方法中的this变量。

遍历数组：ForEach和map。

['a','b','c'].forEach(

function(elem,index){

console.log(index+'.'+elem);

0.a

1.b

2.c

正则表达式：1>使用斜线分割。例如：/^abc$/

2>test（）方法匹配吗： >/^a+b+$/.test('aaab') true >/^a+b+$/.test('aaa') false

3>exec()方法：匹配以及捕获分组：>/a(b+)a/.exec('\_abbba\_aba\_') ['abbba','bbb']

4>replace（）方法：搜索和替换 >'<a> <bbb>'.replace(/<(.\*?)>/g,'[$1]') '[a] [bbb]'