第30讲：对函数的扩展

一：新增的箭头函数=>

语法：参数 => 函数体 或者 （参数） => { 函数体 }

优点：减少冗余的代码，避免this指向错误

var f = function (v) {

return v + 1;

};

f(2);//返回3

//使用箭头函数，上述代码等效如下,只有一个参数和一条语句

var f = v => v + 1; //单参数可以不用（），单语句可以不用return关键字

//var f = (v) => {return v + 1;};

f(2);//返回3

二：对函数参数默认值的扩展

var sum = function(a,b,c){

b = b||4;

c = c||5;

return a+b+c;

};

console.log(sum(1,2,3));//1+2+3

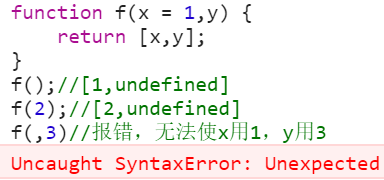
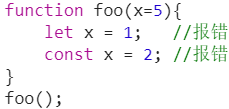
console.log(sum(1,2));//1+2+5

console.log(sum(1));//1+4+5

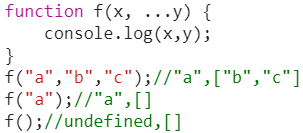
console.log(sum(1,0,0));

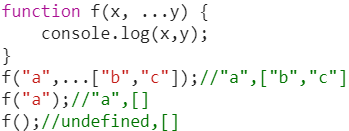
注意事项

1：函数参数默认值是有默认声明的，所以函数体内不能再用let或者是const 重复声明

2：参数一般有顺序，有默认值的参数应该是尾参数。  

三：rest与spread操作符





第31讲：新增的数据类型和数据结构

一：新增的数据类型（symbol）

symbol变量属于基本数据类型（不是对象），前边儿不能使用new命令

let s = Symbol();//不能用new

typeof s;// "symbol"

//Symbol函数可以接受一个字符串作为参数，表示对Symbol实例的描述，主要是为 了在控制台显示，或者转为字符串时，比较容易区分。

var s1 = Symbol('foo');

var s2 = Symbol('bar');

console.log(s1); // Symbol(foo)

console.log(s2); // Symbol(bar)

console.log(s1.toString()); // "Symbol(foo)"

console.log(s2.toString()); // "Symbol(bar)"

//注意，Symbol函数的参数只是表示对当前Symbol值的描述，因此相同参数的 Symbol函数的返回值是不相等的。

// 没有参数的情况

var s1 = Symbol();

var s2 = Symbol();

s1 === s2 // false

// 有参数的情况

var s1 = Symbol("foo");

var s2 = Symbol("foo");

s1 === s2 // false

//Symbol值不能与其他类型的值进行运算，会报错。

var sym = Symbol('My symbol');

//"your symbol is " + sym;//报错

//但是，Symbol值可以显式转为字符串。

var sym = Symbol('My symbol');

String(sym); // 'Symbol(My symbol)'

sym.toString(); // 'Symbol(My symbol)'

作为属性名的symbol

var mySymbol = Symbol();

// 第一种写法

var a = {};

a[mySymbol] = 'Hello!';//注意中括号内不要加引号，后面介绍加引号和不加引 号的区别

// 第二种写法

var a = {

[mySymbol]: 'Hello!'

};

// 第三种写法

var a = {};

Object.defineProperty(a, mySymbol, { value: 'Hello!' });

// 以上写法都得到同样结果

a[mySymbol] // "Hello!"

//注意，Symbol值作为对象属性名时，不能用点运算符，使用中括号是注意使用引 号和不用引号的区别

var mySymbol = Symbol();

var a = {};

a.mySymbol = 'Hello!';

a[mySymbol] // undefined

a['mySymbol'] // "Hello!"

//同理，在对象的内部，使用Symbol值定义属性时，Symbol值必须放在方括号之中

// 如果不用[]的话相当于使用s对应的字符串定义属性

let s = Symbol();

let obj = {

[s]: function (arg) {}

};

obj[s](123);

// 上面代码中，如果s不放在方括号中，该属性的键名就是字符串s，而不是s所代 表的那个Symbol值。

// 采用增强的对象写法，上面代码的obj对象可以写得更简洁一些

let obj = {

[s](arg) { }

};

作为属性名的symbol，区分使用点操作符和中括号操作符时，访问对象的属 性不同，symbol需使用 [ ] ，而不是点

使用symbol值定义属性时，symbol值必须放在方括号之中

symbol作为属性名，该属性不会出想在循环中

二：新增的数据结构set

输出不重复的数

var s1 = new Set([1,2,3,4,5,5,6,2,2]);

console.log(s1);

var s2 = new Set();

[2, 3, 5, 4, 5, 2, 2].map(x => s2.add(x));

for (var i of s2) {

console.log(i);

}// 2 3 5 4

var set = new Set([1, 2, 3, 4, 4]);

console.log([...set]);

// [1, 2, 3, 4]

var items = new Set([1, 2, 3, 4, 5, 5, 5, 5]);

console.log(items.size); // 5

// 去除数组的重复成员

[...new Set([1,2,3,3])];

s.add(1).add(2).add(2);

// 注意2被加入了两次

s.size // 2

s.has(1); // true

s.has(2); // true

s.has(3); // false

s.delete(2);

s.has(2); // false

三：新增数据结构map

