ES6

## 变量的声明

### Let

* Let声明的变量只在它自己的代码块（一对花括号包含的区域称之为一个代码块）起作用
* Let声明的变量，必须先声明，再使用

### Const

* Const 用来声明一个不可赋值的变量
* 变量的值只能在声明的时候赋予

Const a=1；

a=2 //错误

* 下面的不是赋值，是操作，所以是可以的

const arr=[1 , 2];

arr.push( 3 );

//[1 ,2 ,3]

* Const声明的变量不能被删除
* Const的作用域和let一样

## Set

1. **定义**：在其他语言里面称为集合，是一种和Array相似的数据结构，不同之处在于，set中的元素都是不重复的。
2. **用法**：

var s= new Set();

s.add(1);//添加元素

console.log(s);

console.log(s.has(1));// 输出true has方法检查元素是否在set中

console.log(s.size);//输出set的长度，相当于length

s.delete(1);//delete用来删除一个元素

console.log(s);

1. 作用

* Set的作用主要是用来去重
* 下面是使用Set去重并进行排序

function 高级写法(){

let arr = [3,5,2,1,3,2,4];

let newArr = [ ...new Set( arr ) ].sort();

console.log( newArr );

};

高级写法();

## Map

1. 定义：

## …扩展符

1. 作用：是把数组解开成单独的元素
2. 用法

var a1=[1,2,3];

var a2=[...a1,4,5];

console.log(a2);

## 解包

1. 定义

解包是其他语言（python）中学过来的特性，见下方例子：

var [a,b]=[1,2];

console.log(a);//1

console.log(b);//2

// 相当于一下老代码

// var arr=[1,2];

// var a=arr[0];

// var b=arr[1];

//一行代码交换变量

[a,b]=[b,a];

console.log(a);//2

console.log(b);//1

## 函数默认参数

var foo=function(a, name="guo"){

console.log(a,name);

}

foo();//undefined,guo

## 可变参数

* 用. . .语法可以实现可变长度的参数，多余的参数会被放进args数组（args 是任意的变量名）

var foo=function(a,...args){

console.log(a,args.length);

}

foo(1,2,3,4)//1,3

## 箭头函数

* 箭头函数就是匿名函数的简化版，可以使代码更简洁
* 箭头函数的this是绑定了的
* 箭头函数没有arguments对象
* 如果要多参数，必须用. . .
* 语法如下

（参数1，参数2）=>{语句}

（参数1，参数2）=>语句

//上面两行相当于下面这个函数

function（）{

return 语句

}

//如果只有一个参数，圆括号可省略

（参数1）=>{语句}

参数1=>{语句}

//但是如果没有参数，必须需要使用圆括号

（）=>{语句}

var a1=[1,2,3];

var a2=a1.map(function(n){

return n\*n;

})

//以上代码相当于

var a3=a1.map(n=>n\*n);

console.log(a2);

console.log(a3);

a4=function(a){

return function(b){

return a+b;

}

}

console.log( a4(3)(3) )

//以上代码相当于

console.log( (a=>b=>a+b)(3)(3) )

## 新增函数

### Includes

* 在ES5，Array已经提供了indexOf用来查找某个元素的位置，如果不存在就返回-1，但是这个函数在判断数组是否包含某个元素时有两个小不足，第一个是它会返回-1和元素的位置来表示是否包含，在定位方面是没问题，就是不够语义化。另一个问题是不能判断是否有NaN的元素。

const arr0 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', NaN]

console.log('%s', arr0.indexOf("g"))//6

console.log('%s', arr0.indexOf(NaN))//-1

* ES6提供了Array.includes()函数判断是否包含某一元素，**除了不能定位外**，解决了indexOf的上述的两个问题。它直接返回true或者false表示是否包含元素，对NaN一样能有有效。

const arr1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', NaN]

console.log('%s', arr1.includes('c'))//true

console.log('%s', arr1.includes('z'，3))//false 第二个参数表示判断的起始位置，第二个参数也可以是负数，表示从右数过来第几个，但是不改变判断搜索的方向，搜索方向还是从左到右

console.log('%s', arr1.includes(NaN))//true