<script>

// 第一部分：综合的

// 字符串转数组

var str1="abcdefg";

var arr1a=str1.split("",3);

console.log(arr1a); //["a", "b", "c"]

var arr1b=str1.split(""); //不加第二个参数就是转换全部

console.log(arr1b); //["a", "b", "c", "d", "e", "f", "g"]

// 数组转字符串

var arr2=["aa", "bb", "cc", "dd", "ee", "ff"];

var str2=arr2.join("@");

console.log(str2); //aa@bb@cc@dd@ee@ff

// 第二部分：字符串和数组都有的方法

// 相加.concat()

// 数组的concat

var a1=['a', 'b', 'c'];

var a2=['d', 'e', 'f'];

var a3=a1.concat(a2);

console.log(a3); //["a", "b", "c", "d", "e", "f"]

var a4=a1.concat(a2, 'g'); //还可以带第二个参数，在尾部追加

console.log(a4); //["a", "b", "c", "d", "e", "f", "g"]

// 字符串的concat（很少被使用，通常使用加号"+"来连接字符串）

var s1= 'C'.concat('a', 't');

console.log(s1); //Cat

// 截取.slice()

// 数组方法 array.slice(start, end)

var arr3=['a', 'b', 'c', 'd', 'e'];

var arr3a=arr3.slice(2,4); //从arr3的第2位复制到arr3的第4位之前

console.log(arr3a); //["c", "d"]

var arr3b=arr3.slice(0); //如果不加第二个参数，则复制到arr3的末尾

console.log(arr3b); //["a", "b", "c", "d", "e"]

// 字符串方法 string.slice(start, end)

var str3='and in it he says "Any damn fool could';

var str3a=str3.slice(18);

console.log(str3a); //"Any damn fool could

var str3b=str3.slice(0,3);

console.log(str3b); //and

var str3c=str3.slice(-5);

console.log(str3c); //could

// 第三部分：数组的方法

// 数组的追加和删除 push pop unshift shift

// 尾部追加元素 array.push(item...) 修改原数组，并返回数组修改后的长度

var arr4=['a', 'b', 'c'];

var arr4a=['d', 'e'];

var arr4b=arr4.push(arr4a, 'f');

console.log(arr4); //原数组被改变 ['a', 'b', 'c', ['d', 'e'], 'f']

console.log(arr4b); //返回原数组改变后的长度

// 尾部删除数据 array.pop() 删除原数组最后一个元素并返回该元素，如果数组是空的，则返回undefined

var arr5=[1,2,3];

var arr5a=arr5.pop();

console.log(arr5); //原数组被改变 [1, 2]

console.log(arr5a); //返回被删除的元素 3

// 从头部追加元素 array.unshift(item...) 原数组被修改，并返回数组修改后的长度

var arr6=[1,2,3,4,5];

var arr6a=arr6.unshift(0);

console.log(arr6); //原数组被修改 [0, 1, 2, 3, 4, 5]

console.log(arr6a); //返回被修改数组的长度 6

// 从头部删除元素 array.shift() 删除原数组的第1个元素，并返回该元素。如果数组是空的，则返回undefined

var arr7=[1,2,3];

var arr7a=arr7.shift();

console.log(arr7); //原数组被修改 [2, 3]

console.log(arr7a); //返回被删除的元素 1

//array.splice(start, deleteCount, item...)

// 从数组中删除一个或多个元素，并用新的item代替被删除的元素。参数deleteCount是要被删除的个数（从start开始）

// 该方法返回一个包含被删除元素的数组

var arr8=['a', 'b', 'c', 'd', 'e'];

var arr8a=arr8.splice(1,2,'w');

console.log(arr8); //原数组被修改，从1位开始的2个元素被删除，用'w'替换了那两个位置 ["a", "w", "d", "e"]

console.log(arr8a); //返回一个包含被删除元素的数组 ["b", "c"]

var arr8b=arr8.splice(1,1);

console.log(arr8); //可以不带第三个参数，就是只删除，不替换

console.log(arr8b);

// 反转 array.reverse() 反转原数组里的元素顺序，返回反转后的数组

var arr9=[1,2,3];

var arr9a=arr9.reverse();

console.log(arr9); //原数组被改变 [3, 2, 1]

console.log(arr9a); //返回反转后的数组的引用，注意是指针，再操作arr9a时，arr9也会随之改变 [3, 2, 1]

arr9a.reverse();

console.log(arr9a); //又对arr9a进行反转操作 [1, 2, 3]

console.log(arr9); //arr9也被改变了 [1, 2, 3]

// 排序 array.sort(comparefn)

// 数字排序

var arr10=[4, 42, 10, 8, 16, 23];

arr10.sort(function(a,b){

return a-b;

});

console.log(arr10); //[4, 8, 10, 16, 23, 42]

// 数字和字符串混排

var arr11=[4, 'bb', 42, 10, 'a', 8, 16, 'xyz', 23];

arr11.sort(function(a, b){

if(a===b){

return 0;

}

if(typeof a === typeof b){

return a < b ? -1 : 1;

}

return typeof a < typeof b ? -1 :1;

});

console.log(arr11); //[4, 8, 10, 16, 23, 42, "a", "bb", "xyz"]

// 如果需要倒序，可以再使用.reverse方法

// 第四部分：字符串的方法

// 字符串的基本查找方法 charAt charCodeAt indexOf lastIndexOf

// string.charAt(pos) 返回pos位置处的字符。如果pos小于0或者大于等于字符串长度，则返回空字符串

var str4='abcdef';

var str4a=str4.charAt(3);

console.log(str4a); //d

//string.charCodeAt(pos) 返回pos位置处的字符码位。如果pos小于0或者大于等于字符串长度，则返回NaN

var str5='abcdef';

var str5a=str5.charCodeAt(3);

console.log(str5a); //100

//string.indexOf(searchString, position)

// 查找字符串searchString，如果找到，则返回第一个匹配字符的位置，否则返回-1

// position是可选参数，可设置从某个位置开始查找

var str6='Mississippi';

var str6a=str6.indexOf('ss');

var str6b=str6.indexOf('ss', 3);

var str6c=str6.indexOf('ss',6);

console.log(str6a); //2

console.log(str6b); //5

console.log(str6c); //-1

//string.lastIndexOf(searchString, position)

// 和indexOf方法类似，但是从末尾开始查找

var str7='Mississippi';

var str7a=str7.lastIndexOf('ss');

var str7b=str7.lastIndexOf('ss', 3);

var str7c=str7.lastIndexOf('ss',6);

console.log(str7a); //5

console.log(str7b); //2

console.log(str7c); //5

// 比较字符串 string.localeCompare(that) 如果string比that小则结果为负数。如果相等为0

// 可以当做sort的比较函数使用

var str8=['AAA', 'A', 'aa', 'a', 'Aa', 'aaa'];

str8.sort(function(a, b){

return a.localeCompare(b);

});

console.log(str8); //["a", "A", "aa", "Aa", "aaa", "AAA"]

// 转换为小写 string.toLowerCase()

var str9='I am a Chinese';

var str9a=str9.toLowerCase(); //不能改变原字符串

console.log(str9a); //i am a chinese

// 转换为大写 string.toUpperCase()

var str10='I am a Chinese';

var str10a=str10.toUpperCase(); //不能改变原字符串

console.log(str10a); //I AM A CHINESE

//根据一串数字编码返回一个字符串 String.fromCharCode(char...)

var str11=String.fromCharCode(67, 97, 116);

console.log(str11); //Cat

//删除前后的空格 trim() 返回原字符串的复制，不改变原字符串

var str12=' wo shi ren ';

var str12a=str12.trim();

console.log(str12a); // 'wo shi ren'

console.log(str12); //原字符串不会被改变 ' wo shi ren '

// 字符串的匹配方法 match() search() replace()

var str13='cat, bat, sat, fat';

var str13a= /.sat/;

var str13b=str13.match(str13a);

console.log(str13b.index); //9

console.log(str13b[0]); //sat

console.log(str13a.lastIndex); //0

var str13c=str13.search(str13a);

console.log(str13c); //search返回第一个匹配元素的位置，如果没找到就返回-1

var str13d=str13.replace('at', 'ond'); //使用字符串'at'只能替换第一个匹配的位置

console.log(str13d); //cond, bat, sat, fat

var str13e=str13.replace(/at/g, 'xxx'); //将第一个参数修改为带有全局标志的正则表达式，就将全部"at"都替换成了"ond"。

console.log(str13e); //cxxx, bxxx, sxxx, fxxx

</script>