1.数组

- concat() 连接两个或更多的数组，并返回结果。
  join() 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。
  pop() 删除并返回数组的最后一个元素
  push() 向数组的末尾添加一个或更多元素，并返回新的长度。
  shift() 删除并返回数组的第一个元素。
  unshift() 向数组的开头添加一个或更多元素，并返回新的长度。
  reverse() 颠倒数组中元素的顺序。
  slice() 从某个已有的数组返回选定的元素
  sort() 对数组的元素进行排序
  splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。
  toSource() 返回该对象的源代码。只有Gecko浏览器支持（firefox）
  toString() 把数组转换为字符串，并返回结果。
  toLocaleString() 把数组转换为本地数组，并返回结果。
  valueOf() 返回数组对象的原始值
- indexOf()
- lastIndexOf()
- forEach()
- map()
- filter()
- every()
- some()
- reduce()
- reduceRight()

针对ES5新增的方法浏览器支持情况：

Opera 11+ ,Firefox 3.6+ ,Safari 5+ ,Chrome 8+ ,Internet Explorer 9+

对于支持的浏览器版本，可以通过Array原型扩展来实现

1.1 concat()

concat():法用于连接两个或多个数组。该方法不会改变现有的数组，而仅仅会返回被连接数组的一个副本。

```
var a = 1;
var arrayOne = [2,3,4,5];
var arrayTwo = [6,7,8];
console.log(arrayOne.concat(a,arrayTwo))//[1,2,3,4,5,6,7,8]
```

1.2 join()

join(separator):将数组的元素组起一个字符串，以separator为分隔符，省略的话则用默认用逗号为分隔符，该方法只接收一个参数:即分隔符。

```
var array = [1,2,3];
console.log(array.join());//1,2,3
console.log(array.join("-"));//1-2-3
console.log(array)//[1,2,3]   原数组不变

//通过join()方法可以实现重复字符串，只需传入字符串以及重复的次数，就能返回重复后的字符串
function repeatString(str, n) {
    return new Array(n + 1).join(str);
}
console.log(repeatString("Hi", 5)); // HiHiHiHiHi
```

1.3 push()和pop()

push():可以接收任意数量的参数，把它们逐个添加到数组末尾，并返回修改后数组的长度。

pop():数组末尾移除最后一项，减少数组的length值，然后返回移除的项。

```
var array = ["a","b","c"];
var count = array.push("d","e");
console.log(count);//5
console,log(array);//["a","b","c","d","e"];
var item = array.pop();
console.log(item);//e
console.log(array);//["a","b","c","d"]
```

1.4 shift()和unshift()

shift():删除原数组第一项，并返回删除元素的值；如果数组为空则返回undefined。

unshift:将参数添加到原数组开头，并返回数组的长度。

```
var array = ["a","b","c"];
var count = array.unshift("d","e");
console.log(count);//5
console.log(array);//["d","e","a","b","c"];
var item = array.shift();
console.log(item);//d
console.log(array);//["e","a","b","c"]
```

1.5 reverse()

reverse():反转数组项的顺序

```
var arrayOne = [1,2,3,4,5];
var arrayTwo = arrayOne.reverse();
console.log(arrayOne);//[5,4,3,2,1]
console.log(arrayTwo);//[5,4,3,2,1]
```

1.6 slice()

slice(start,end):从已有的数组中返回选定的元素象。

start:必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2 指倒数第二个元素，以此类推。

end:可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 start 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。

```
var arr = ["a","b","c","d","e","f"];
var arrOne = arr.slice(1);
var arrTwo = arr.slice(1,4);
var arrThree = arr.slice(1,-2);
var arrFour = arr.slice(-4,-1);
console.log(arr);//["a", "b", "c", "d", "e", "f"]
console.log(arrOne);//["b", "c", "d", "e", "f"]
console.log(arrTwo);//["b", "c", "d"]
console.log(arrThree);//["b", "c", "d"]
console.log(arrFour);//["c", "d", "e"]
```

1.7 sort()

sort(sortby):方法用于对数组的元素进行排序。数组在原数组上进行排序，不生成副本。

sortby：可选。规定排序顺序。必须是函数。

```
var arrOne = ["c","b","a","d"];
console.log(arrOne.sort());//["a","b","c","d"]

var arrTwo = [24,13,51,3];
console.log(arrTwo.sort());//[13,24,3,51]
console.log(arrTwo);//[24,13,51,3]  原数组被改变

//为了解决上述问题，sort()方法可以接收一个比较函数作为参数，以便我们指定哪个值位于哪个值的前面。比较函数接收两个参数，如果第一个参数应该位于第二个之前则返回一个负数，如果两个参数相等则返回 0，如果第一个参数应该位于第二个之后则返回一个正数。

function compare(a, b) {
  if(a < b) {
    return -1;
  }else if (a > b) {
    return 1;
  }else {
    return 0;
  }
}
console.log(arrTwo.sort(compare)); // [3, 13, 24, 51]
```

1.8 splice()

splice(index,howmany,item1,.....,itemX):方法向/从数组中添加/删除项目，然后返回被删除的项目。该方法会改变原始数组。

```
var arr = [1,3,5,7,9,11];
var arrRm = arr.splice(0,2);
console.log(arr);//[5,7,9,11];
console.log(arrRm);//[1,3]
var arrRmTwo = arr.splice(2,0,4,6);
console.log(arr);//[5,7,4,6,9,11]
console.log(arrRmTwo);[]
var arrRmThree = arr.splice(1,1,2,4);
console.log(arr);//[5,2,4,4,6,9,11]
console.log(arrRmThree);//[7]
```

1.9 toString()

toString():把数组转换为字符串，并返回结果。

```
var arr = ["martin","que","ting"];
console.log(arr.toString());//martin,que,ting
```

1.9 toLocaleString()

toLocaleString():方法可根据本地时间把 Date 对象转换为字符串，并返回结果。

```
var date = new Date();
console.log(date.toLocaleString());//2018/7/4 上午9:13:02
```

2.0 valueOf()

valueOf():数组的元素被转换为字符串，这些字符串由逗号分隔，连接在一起。其操作与 Array.toString 和 Array.join 方法相同。

对象	返回值
Array	数组的元素被转换为字符串，这些字符串由逗号分隔，连接在一起。其操作与 Array.toString 和 Array.join 方法相同。
Boolean	Boolean 值。
Date	存储的时间是从 1970 年 1 月 1 日午夜开始计的毫秒数 UTC。
Function	函数本身。
Number	数字值。
Object	对象本身。这是默认情况。
String	字符串值。

2.1 indexOf()和lastIndexOf()

indexOf(): 接收两个参数，要查找的项和（可选的）表示查找起点位置的索引。其中，从数组的开头（位置0）开始向后查找。

lastIndexOf: 接收两个参数，要查找的项和（可选的）表示查找起点位置的索引。其中，从数组的末尾开始向前查找。

```
var arr = [1,3,5,7,7,5,3,1];
console.log(arr.indexOf(5)); //2
console.log(arr.lastIndexOf(5)); //5
console.log(arr.indexOf(5,2)); //2
console.log(arr.lastIndexOf(5,4)); //2
console.log(arr.indexOf("5")); //-1
```

2.2 forEach()

forEach():对数组进行遍历循环，对数组中的每一项运行给定函数。参数都是function类型，默认有传参，参数分别为：遍历的数组内容；第对应的数组索引，数组本身。这个方法没有返回值。

```
var arr = [1,2,3,4,5];
arr.forEach(function(item,index,array){
    console.log(item,index,(array===arr));
    //1 0 true
    //2 1 true
    //3 2 true
    //4 3 true
    //5 4 true
})
```

2.3 map()

map():指“映射”，对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组。

```
var arrOne = [1,2,3,4,5];
var arrTwo = arrOne.map(function(item){
    return item*item
});
console.log(arrTwo);//[1,4,9,16,25]
```

2.4 filter()

filter(function(currentValue,index,arr),thisValue):“过滤”功能，数组中的每一项运行给定函数，返回满足过滤条件组成的数组。

thisValue:可选。对象作为该执行回调时使用，传递给函数，用作 "this" 的值。如果省略了 thisValue ，"this" 的值为 "undefined"。

```
var arrs = [
  {
    name: "tom",
    age: 18,
    sex: "boy"
  },
  {
```

```
    name: "jim",
    age: 19,
    sex: "boy"
  },
  {
    name: "anchor",
    age: 20,
    sex: "boy"
  },
  {
    name: "lucy",
    age: 18,
    sex: "girl"
  },
  {
    name: "lily",
    age: 19,
    sex: "girl"
  },
  {
    name: "andy",
    age: 20,
    sex: "girl"
  }
];

// 过滤条件
var limits = {
  name: 'tom',
  age: 18,
  sex: 'boy'
};

function myfilter(element, index, array) {
  if(this.name && this.name != element.name){ // 姓名过滤
    return false;
  }else if(this.age && this.age != element.age){ // 年龄过滤
    return false;
  }else if(this.sex && this.sex != element.sex){ // 性别过滤
    return false;
  }
  return true;
}

var filtered = arrs.filter(myfilter,limits);
console.log(filtered);//[{name:'tom',age:18,sex:'boy'}]
```

2.5 every()

every():判断数组中每一项都是否满足条件，只有所有项都满足条件，才会返回true。

```
var arr = [1, 2, 3, 4, 5];
var resOne = arr.every(function(item) {
  return item < 10;
});
console.log(resOne); //true
var resTwo = arr.every(function(item) {
  return item < 3;
});
console.log(resTwo); // false
```

2.6 some()

some():判断数组中是否存在满足条件的项，只要有一项满足条件，就会返回true。

```
var arr = [1, 2, 3, 4, 5];
var arr2 = arr.some(function(x) {
  return x < 3;
});
console.log(arr2); //true
var arr3 = arr.some(function(x) {
  return x < 1;
});
console.log(arr3); // false
```

2.7 reduce()和 reduceRight()

reduce(callback[, initialValue]): 方法接收一个函数作为累加器，数组中的每个值（从左到右）开始缩减，最终为一个值

- callback（一个在数组中每一项上调用的函数，接受四个函数）
- previousValue（上一次调用回调函数时的返回值，或者初始值）
- currentValue（当前正在处理的数组元素）
- currentIndex（当前正在处理的数组元素下标）
- array（调用reduce()方法的数组）
- initialValue（可选的初始值。作为第一次调用回调函数时传给previousValue的值）

```
var arr = [0,1,2,3,4];
arr.reduce(function(preValue,curValue,index,array) {
  return preValue + curValue;
}, 5); //15
```

table	previousValue	currentValue	currentIndex	array	返回值
第一次回调	5	0	0	[0,1,2,3,4]	5
第二次回调	5	1	1	[0,1,2,3,4]	6
第三次回调	6	2	2	[0,1,2,3,4]	8
第四次回调	8	3	3	[0,1,2,3,4]	11
第五次回调	11	4	4	[0,1,2,3,4]	15

reduceRight()方法的功能和reduce()功能是一样的，不同的是reduceRight()从数组的末尾向前将数组中的数组项做累加。

```
var arr = [0,1,2,3,4];
arr.reduceRight(function (preValue,curValue,index,array) {
  return preValue + curValue;
}); //10
```

table	previousValue	currentValue	currentIndex	array	返回值
第一次回调	4	3	3	[0,1,2,3,4]	7
第二次回调	7	2	2	[0,1,2,3,4]	9
第三次回调	9	1	1	[0,1,2,3,4]	10
第四次回调	10	0	0	[0,1,2,3,4]	10