freeshineit / javascript-array

javascript-array

Edit

Add topics

5 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

freeshineit aad demo — Latest commit 75d8b8a on Oct 16

demo	aad demo	a month ago
images	add images	5 months ago
.gitignore	add .gitignore	5 months ago
README.md	add images	5 months ago
package.json	Initial project	5 months ago

README.md

数组是值的有序集合。

```
> []
< ▼ [] ℹ
    length: 0
  ▼ __proto__: Array(0)
    ▶ concat: function concat()
    ▶ constructor: function Array()
    ▶ copyWithin: function copyWithin()
    ▶ entries: function entries()
    ▶ every: function every()
    ▶ fill: function fill()
    ▶ filter: function filter()
    ▶ find: function find()
    ▶ findIndex: function findIndex()
    ▶ forEach: function forEach()
    ▶ includes: function includes()
    ▶ indexOf: function indexOf()
    ▶ join: function join()
    ▶ keys: function keys()
    ▶ lastIndexOf: function lastIndexOf()
      length: 0
    ▶ map: function map()
    ▶ pop: function pop()
    ▶ push: function push()
    ▶ reduce: function reduce()
    ▶ reduceRight: function reduceRight()
    ▶ reverse: function reverse()
    ▶ shift: function shift()
    ▶ slice: function slice()
    ▶ some: function some()
    ▶ sort: function sort()
    ▶ splice: function splice()
    ▶ toLocaleString: function toLocaleString()
    ▶ toString: function toString()
    ▶ unshift: function unshift()
    ▶ Symbol(Symbol.iterator): function values()
    ▶ Symbol(Symbol.unscopables): Object
    ▶ __proto__: Object
>
```

Array.prototype.concat() [ES3]

（concat()方法合并两个或两个以上数组。此方法不更改现有数组，而是返回新数组）

语法：

> var new_array = old_array.concat(value1[,value2[, ...[,valueN]]])

实例：

```
var arr1=['a','b','c'];

var arr2=['d','e','f'];

var arr3=arr1.concat(arr2);

// arr3 is a new array [ "a", "b", "c", "d", "e", "f" ]
```

Array.prototype.copyWithin() [ES6]

（copyWithin方法，在当前数组内部，将指定位置的成员复制到其他位置（会覆盖原有成员），然后返回当前数组。也就是说，使用这个方法，会修改当前数组。target（必需）：从该位置开始替换数据。start（可选）：从该位置开始读取数据，默认为0。如果为负值，表示倒数。end（可选）：到该位置前停止读取数据，默认等于数组长度。如果为负值，表示倒数。）

（ copyWithin()方法是数组的浅拷贝部分， 而数组的大小没有改变。）

语法：

> arr.copyWithin(target)

> arr.copyWithin(target,start)

> arr.copyWithin(target,start,end)

实例：

```
['alpha','bravo','charlie','delta'].copyWithin(2,0);

// results in ["alpha", "bravo", "alpha", "bravo"]
```

Array.prototype.entries() [ES6]

(entries()方法返回一个新数组的迭代器对象包含数组中的每个索引的键/值对。)

语法：

> a.entries()

实例：

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

var iterator = a.entries();

console.log(iterator.next().value);  // [0, 'a']

console.log(iterator.next().value);  // [1, 'b']

console.log(iterator.next().value);  // [2, 'c']


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

var iterator = a.entries();

for(lete of iterator){

    console.log(e);

}// [0, 'a']// [1, 'b']// [2, 'c']
```

Array.prototype.every() [ES5]

(every方法是数组的逻辑判定：对数组元素应用指定对函数进行判定，返回true 或 false)

语法：

> arr.every(callback[,thisArg])

实例：

```
functionisBigEnough(element,index,array){
       return element >= 10;
}
[12,5,8,130,44].every(isBigEnough);// false
[12,54,18,130,44].every(isBigEnough);// true
[1,2,3,4,5,6,7,8].every(function(x){ x < 10; } ) //true 所有值<10
```

Array.prototype.fill() [ES6]

(fill()方法填充数组中的所有从一开始指数结束指标与静态值的元素)

语法：

> arr.fill(value)

> arr.fill(value,start = 0)

> arr.fill(value,start = 0,end = this.length)

(value 值是必须的；start 开始位置 ， 可选、默认值为；end结束位置，可选、默认值为数组的length)

实例:

```
varnumbers=[1,2,3]
numbers.fill(1);

[1,2,3].fill(4);// [4, 4, 4]
[1,2,3].fill(4,1);// [1, 4, 4]
[1,2,3].fill(4,1,2);// [1, 4, 3]
[1,2,3].fill(4,1,1);// [1, 2, 3]
[1,2,3].fill(4,-3,-2);// [4, 2, 3]
[1,2,3].fill(4,NaN,NaN);// [1, 2, 3]
Array(3).fill(4);// [4, 4, 4]
[].fill.call({length:3},4);// {0: 4, 1: 4, 2: 4, length: 3}
```

Array.prototype.filter() [ES5]

(filter()方法返回的是数组的一个子集<也是一个数组，新数组>，传递的函数是用来逻辑判定的)

语法：

> var newArray = arr.filter(callback[,thisArg])

(element:数组中正在处理的当前元素。index:数组中正在处理的当前元素的索引。array:当前数组。thisArg:fn函数中this指向.)

实例：

```
var words = ["spray","limit","elite","exuberant","destruction","present"];

var longWords = words.filter(function(word){return word.length > 6;})

// Filtered array longWords is ["exuberant", "destruction", "present"]
```

Array.prototype.find() [ES6]

（find() 方法返回符合规定的函数的数组的第一个元素的值。否则返回 undefinded）

语法：

> arr.find(callback[,thisArg])

(element:数组中正在处理的当前元素。index:数组中正在处理的当前元素的索引。array:当前数组。thisArg:fn函数中this指向.)

实例：

```
function isBigEnough(element){
    return element>=15;
}
[12,5,8,130,44].find(isBigEnough);  // 130

function isPrime(element,index,array){
    var start=2;
    while(start<=Math.sqrt(element)){
        if(element%start++<1){
             returnfalse;
        }
    }
     return element>1;
}
console.log([4,6,8,12].find(isPrime));  // undefined, not found
console.log([4,5,8,12].find(isPrime));  // 5
```

Array.prototype.findIndex() [ES6]

(findIndex()方法和find()很相似，findIndex(0) 的返回值是数组的索引，返回满足条件的数组的第一个元素的索引。否则返回 -1)

语法：

> arr.findIndex(callback[,thisArg])

(element:数组中正在处理的当前元素。index:数组中正在处理的当前元素的索引。array:当前数组。thisArg:fn函数中this指向.)

实例：

```
function isBigEnough(element){
    return element >= 15;
}
[12,5,8,130,44].findIndex(isBigEnough);
// index of 4th element in the Array is returned,
// so this will result in '3'


functionis Prime(element,index,array){
      var start = 2;
      while(start <= Math.sqrt(element)) {
            if(element%start++<1){
                  return false;
             }
       }
        return element>1;
 }
console.log([4,6,8,12].findIndex(isPrime));   // -1, not found
console.log([4,6,7,12].findIndex(isPrime));   // 2
```

Array.prototype.forEach() [ES5]

（foreach()方法用来遍历数组中的每个元素）

语法：

> arr.forEach(functioncallback(currentValue, index, array) { //your iterator }[,thisArg]);

（currentValue ： 数组中正在处理的当前元素的值。index:数组中正在处理的当前元素的索引。 array:当前数组。thisArg:回调执行时，使用此值（即referenceobject））

实例：

```
vara=['a','b','c'];
a.forEach(function(element){
      console.log(element);
});
//a// b// c
```
（其实数组的遍历有很多方法 `for...of..` 、`for...in..` 等都可以）

Array.prototype.includes() [ES7]

(includes()判定数组包含某元素，返回true或false适当。)

语法：

> arr.includes(searchElement)

> arr.includes(searchElement,fromIndex)

(searchElement:搜索的元素，fromIndex：开始搜索元素的索引，从该位置开始搜索)

实例：

```
[1,2,3].includes(2);// true

[1,2,3].includes(4);// false

[1,2,3].includes(3,3);// false

[1,2,3].includes(3,-1);// true

[1,2,NaN].includes(NaN);// true
```

Array.prototype.indexOf() [ES5]

（indexOf()方法搜索整个数组中具有给定值的元素，返回找到的第一个元素的索引或者如果没有找到就返回-1，从头至尾搜索）

语法：

> arr.indexOf(searchElement[,fromIndex])

(searchElement:搜索的元素，fromIndex：开始搜索元素的索引，从该位置开始搜索)

实例：

```
var array=[2,9,9];

array.indexOf(2);// 0

array.indexOf(7);// -1

array.indexOf(9,2);// 2

array.indexOf(2,-1);// -1

array.indexOf(2,-3);// 0
```

Array.prototype.lastIndexOf() [ES5]

（lastIndexOf()方法和indexOf()方法类似，不过lastIndexOf()搜索是从尾开始的）

语法：

> arr.lastIndexOf(searchElement)

> arr.lastIndexOf(searchElement,fromIndex)

实例：

```
var numbers=[2,5,9,2];

numbers.lastIndexOf(2);// 3

numbers.lastIndexOf(7);// -1

numbers.lastIndexOf(2,3);// 3

numbers.lastIndexOf(2,2);// 0

numbers.lastIndexOf(2,-2);// 0
```

```
numbers.lastIndexOf(2,-1);// 3
```

Array.prototype.join() [ES3]

(join()方法将数组中所有元素都转化为字符串并连接在一起，返回最后生成的字符串，默认连接元素之间的连接符为‘，’)

语法：

> arr.join()

> arr.join(separator)

(separator：连接字符串的连接符)

实例：

```
var a=['Wind','Rain','Fire'];

a.join();// 'Wind,Rain,Fire'

a.join(', ');// 'Wind, Rain, Fire'

a.join(' + ');// 'Wind + Rain + Fire'a.join('');// 'WindRainFire'
```

Array.prototype.keys() [ES6]

（ keys()方法返回一个新数组的迭代器，包含数组中的每个索引键）

语法：

> arr.keys()

实例：

```
var arr = ['a',,'c'];

var sparseKeys=Object.keys(arr);

var denseKeys=[...arr.keys()];

console.log(sparseKeys);// ['0', '2']

console.log(denseKeys);// [0, 1, 2]
```

Array.prototype.map() [ES5]

(map()方法将调用的数组的每个元素传递给指定的函数，并返回一个数组，它包含该函数的返回值，其实它也是对数组的遍历)

语法：

> var new_array = arr.map(callback[,thisArg])

（currentValue： 数组中正在处理的当前元素的值。index:数组中正在处理的当前元素的索引。 array:当前数组。thisArg:执行回调时使用此值<可选>）

实例：

```
var kvArray = [ {key:1,value:10},

                {key:2,value:20},

                {key:3,value:30} ];

var reformattedArray = kvArray.map(
```

```
    function(obj) {
        var rObj = {};
        rObj[obj.key] = obj.value;
        returnr Obj;
      }
);
console.log(reformattedArray);  // [{1: 10}, {2: 20}, {3: 30}],
```

Array.prototype.pop() [ES3]

（pop()方法允许将数组当做栈来使用，它的作用是删除数组的最后一个元素，减少数组长度并返回它删除的值，如果数组为空，返回undefined）

语法：

> arr.pop()

实例：

```
var myFish = ['angel','clown','mandarin','sturgeon'];
var popped = myFish.pop();
console.log(myFish);  // ['angel', 'clown', 'mandarin' ]
console.log(popped);  // 'sturgeon'
```

Array.prototype.push() [ES3]

（push()方法允许将数组当做栈来使用，它的作用是添加元素在数组的末尾，返回数组变化后的长度）

语法：

> arr.push([element1[, ...[,elementN]]])

（elementN： 添加到数组末尾到元素）

实例：

```
var sports=['soccer','baseball'];
var total=sports.push('football','swimming');
console.log(sports);  // ['soccer', 'baseball', 'football', 'swimming']
console.log(total);// 4
```

Array.prototype.reduce() [ES5]

（reduce()方法使用指定的函数将数组元素进行组合，生成单个值）

语法：

> arr.reduce(callback, [initialValue])

（callback：回调函数，initialValue：初始化值<可选>）

实例：

```
var sum = [0,1,2,3].reduce(function(a,b){
         returna+b;
},0);// sum is 6


var flattened = [[0,1],[2,3],[4,5]].reduce(
        function(a,b){
              return a.concat(b);
       },[]
);
// flattened is [0, 1, 2, 3, 4, 5]
//  flattened中的匿名函数执行步骤:
//         a                    b                    retrun
//         []                 [0,1]                  [0,1]
//        [0,1]               [2,3]               [0,1,2,3]
//      [0,1,2,3]             [4,5]           [0,1,2,3,4,5]
```

Array.prototype.reduceRight() [ES5]

(reduceRight()方法和reduce()方法类似，只不过reduceRight()方法组合元素的顺序是从右往左，生成单个值)

语法:

> arr.reduceRight(callback[, initialValue])

实例:

```
var flattened = [[0, 1], [2, 3], [4, 5]].reduceRight(function(a, b) {
    return a.concat(b);
}, []);

// flattened is [4, 5, 2, 3, 0, 1]
```

Array.prototype.reverse() [ES3]

（reverse()方法将数组中的元素颠倒🙃顺序，返回逆序的数组，是在原来数组的基础上进行重新排列）

语法:

> a.reverse()

实例:

```
var a=['one','two','three'];
var reversed=a.reverse();
console.log(a);// ['three', 'two', 'one']
console.log(reversed);// ['three', 'two', 'one']
```

Array.prototype.shift() [ES3]

(shift()方法的作用是删除数组的第一个元素并将其返回，然后把所有后面是元素前移一个位置来填补空缺，如果数组为空，则返回undefinded)；

语法：

> arr.shift()

实例：

```
var myFish=['angel','clown','mandarin','surgeon'];

console.log('myFish before:',myFish);

// myFish before: ['angel', 'clown', 'mandarin', 'surgeon']

var shifted=myFish.shift();

console.log('myFish after:',myFish);

// myFish after: ['clown', 'mandarin', 'surgeon']
```

Array.prototype.unshift() [ES3]

（unshift()方法的作用是在数组首部添加元素,返回新的数组的长度）

语法：

> arr.unshift([element1[, ...[,elementN]]])

实例：

```
var arr=[1,2];

arr.unshift(0);// result of call is 3, the new array length

// arr is [0, 1, 2]

arr.unshift(-2,-1);// = 5

// arr is [-2, -1, 0, 1, 2]

arr.unshift([-3]);// arr is [[-3], -2, -1, 0, 1, 2]
(shift() )
```

Array.prototype.slice() [ES3]

slice()方法返回指定数组的一个片段或子数组，它的两个参数分别指定了片段的开始位置和结束位置。返回的数组包含第一个参数指定的位置和所有到但不含第二个参数指定的位置之间的所有数组元素。

语法：

> arr.slice()

> arr.slice(begin)

> arr.slice(begin, end)

实例：

```
var a = ['zero', 'one', 'two', 'three'];
var sliced = a.slice(1, 3);

console.log(a);      // ['zero', 'one', 'two', 'three']
```

```
console.log(sliced); // ['one', 'two']
```

Array.prototype.some() [ES5]

some()方法是数组的逻辑判定，他对数组元素应用指定对函数进行判定，返回true和false。some()就像数学中的'存在'量词一样，当数组中至少有一个元素满足调用的判定函数就返回true。

语法：

> arr.some(callback[, thisArg])

实例：

```
function isBiggerThan10(element, index, array) {
   return element > 10;
}

[2, 5, 8, 1, 4].some(isBiggerThan10);  // false
[12, 5, 8, 1, 4].some(isBiggerThan10); // true
```

Array.prototype.sort() [ES3]

sort()方法将数组中的元素排序并返回排序后的数组，当不带参数时，数组元素以字母表顺序排序（默认排序顺序是根据字符串unicode 编码进行的）。如果有undefined元素，它将被排到尾部。

语法：

> arr.some(callback[, thisArg])

实例：

```
var fruit = ['cherries', 'apples', 'bananas'];
fruit.sort(); // ['apples', 'bananas', 'cherries']
//unicode sort

var scores = [1, 10, 21, 2];
scores.sort(); // [1, 10, 2, 21]
// Note that 10 comes before 2,
// because '10' comes before '2' in Unicode code point order.

var things = ['word', 'Word', '1 Word', '2 Words'];
things.sort(); // ['1 Word', '2 Words', 'Word', 'word']
// In Unicode, numbers come before upper case letters,
// which come before lower case letters.


var num = [1, 10, 21, 2, 29, 3, 8];
num.sort((a,b) => {
    return a > b;
})
// 按照升序进行排序
console.log(num) // [1, 2, 3, 8, 10, 21, 29]
```

Array.prototype.splice() [ES3]

splice()方法是在数组中插入和删除元素的通用方法.如果只删除一个元素，则返回一个元素的数组。如果没有移除任何元素，则返回空数组。

语法：

> array.splice(start)

> array.splice(start, deleteCount)

> array.splice(start, deleteCount, item1, item2, ...)

start delectCount item1...

实例：

```
var myFish = ['angel', 'clown', 'mandarin', 'sturgeon'];

myFish.splice(2, 0, 'drum'); // insert 'drum' at 2-index position
// myFish is ["angel", "clown", "drum", "mandarin", "sturgeon"]

myFish.splice(2, 1); // remove 1 item at 2-index position (that is, "drum")
// myFish is ["angel", "clown", "mandarin", "sturgeon"]

var myFish = ['angel', 'clown', 'trumpet', 'sturgeon'];
var removed = myFish.splice(0, 2, 'parrot', 'anemone', 'blue');
// myFish is ["parrot", "anemone", "blue", "trumpet", "sturgeon"]
// removed is ["angel", "clown"]
```

Array.prototype.toLocaleString() [ES3]

toLocaleString()方法将数组的元素转化为字符串，并且使用本地化分隔符将这些字符串连接起来生成最终的字符串。

语法：

> arr.toLocaleString();

> arr.toLocaleString(locales);

> arr.toLocaleString(locales, options);

实例：

```
var number = 1337;
var date = new Date();
var myArr = [number, date, 'foo'];

var str = myArr.toLocaleString();

console.log(str);
// logs '1337,6.12.2013 19:37:35,foo'
// if run in a German (de-DE) locale with timezone Europe/Berlin


var prices = ['￥7', 500, 8123, 12];
prices.toLocaleString('ja-JP', { style: 'currency', currency: 'JPY' });
// "￥7,￥500,￥8,123,￥12"
```

Array.prototype.toSource() [Non-standard]

语法：

> arr.toSource()

实例：

```
var arr = new Array('a', 'b', 'c', 'd');

arr.toSource();
//returns ['a', 'b', 'c']
```

*** 现在多家浏览器厂商都是没有对其进行实现

Array.prototype.toString() [ES3]

toString()方法将数组的元素转化为字符串，并用逗号分割zi cu字符串列表。（和数组的join()方法不带参数相似）

语法：

> arr.toString()

实例：

```
var months = ['Jan', 'Feb', 'Mar', 'Apr'];
months.toString(); // "Jan,Feb,Mar,Apr"
```

Array.prototype.values() [ES6]

values()方法返回一个新数组的迭代器对象，包含每个索引的数组中的值。

语法：

> arr.values()

实例：

```
var a = ['w', 'y', 'k', 'o', 'p'];
var iterator = a.values();

console.log(iterator.next().value); // w
console.log(iterator.next().value); // y
console.log(iterator.next().value); // k
console.log(iterator.next().value); // o
console.log(iterator.next().value); // p


var arr = ['w', 'y', 'k', 'o', 'p'];
var iterator = arr.values();

for (let letter of iterator) {
  console.log(letter);
}

//  w
//  y
//  k
//  o
//  p
```

Array.prototype[@@iterator]() [ES6]

@@iterator属性和 values() 属性的初始值均为同一个函数对象

语法：

> arrSymbol.iterator

实例；

```
var arr = ['w', 'y', 'k', 'o', 'p'];
// 您的浏览器必须支持for...of循环
// 以及let —— 将变量作用域限定在 for 循环中
for (let letter of arr) {
  console.log(letter);
}


var arr = ['w', 'y', 'k', 'o', 'p'];
var eArr = arr[Symbol.iterator]();
console.log(eArr.next().value); // w
console.log(eArr.next().value); // y
console.log(eArr.next().value); // k
console.log(eArr.next().value); // o
console.log(eArr.next().value); // p
```

** 以上是javascript数组的简单操作，具体以MDN为标准