博学谷
www.boxuegu.com

第五章 数组的扩展

一样的在线教育，不一样的教学品质

博学谷
www.boxuegu.com

目录 Contents

Array.from( )方法的应用

- 将**类似数组的对象**或者是**可迭代对象(可遍历对象)**转换成数组。

类似数组的对象：

必须要**有length属性**，同时元素属性名必须是**数值或者可以转成数值的字符**。

属性名代表了数组的索引。

```
let arrList = {
    '0': 'zs',
    '1': 'ls',
    '2': 'ww',
    length: 3
    }
    console.log(Array.from(arrList));
```

使用ES5来进行处理：

```
[].slice.call(arrList)
```

Array.from( )方法的应用

- **类似数组对象转换成数组**

常见的类似数组的对象：**DOM操作返回的NodeList集合、函数内部的arguments对象。**

- **可迭代对象转换成数组**

可迭代对象：**只要是部署Iterator接口的数据结构；例如：String, Map,Set, NodeList对象等；**

Array.from( )方法的应用

- **接受第二个参数**；对每个元素进行处理，将处理后的结果放入返回的数组中；

```
let newArray = Array.from('itcast', function(item) {
            return item + '1'
        })
        console.log(newArray);
```

博学谷
www.boxuegu.com

目录 Contents

Array.of( )方法的应用

◆ **将一组值转换为数组。**

```
console.log(Array.of(5, 6, 7));

console.log(Array.of('a', 'b', 'c'));

// 以前可以使用Array来实现
 console.log(Array('d', 'e', 'f'));
```

Array.of( ) 和 Array 构造函数之间的区别在于**处理整数参数**：

Array.of(7) 创建一个具有单个元素 **7** 的数组，而 Array(7) 创建一个长度为7的空数组 。

目录 Contents

1.find( )方法基本应用

◆ **查找出第一个符合条件的数组成员。**

参数是一个回调函数，所有数组成员依次执行该回调函数，直到找出第一个返回值为true的成员，进而返回该成员。如果没有符合条件的成员，则返回undefined。

```
let arr = [1, 2, 3, 4, 5, 6, 7].find(function(value) {
          if (value % 2 === 0) {
               return value;
           }
       })
       console.log(arr);
```

1.find( )方法基本应用

- find()方法的回调函数可**接受3个参数**，依次为**当前的值**、**当前的位置**和**原数组**。

```
let arr = [1, 2, 3, 4, 5, 6, 7].find(function(value, index, array) {
            if (value % 2 === 0) {
                console.log('index=', index);
                console.log('array=', array);
                return value;
            }

        })
        console.log(arr);
```

2.find( )方法原理

模拟 find 方法的实现：

```
Array.prototype.findTest = function(fn) {
            for (let i = 0; i < this.length; i++) {
                let f = fn(this[i]);
                if (f) {
                    return this[i];
                }
            }
        }
    let result = [1, 2, 3, 4, 5, 6, 7].findTest(function(item) {
            return item === 6
        })
        console.log(result);
```

3.findIndex( )方法基本使用与应用案例

- 返回第一个符合条件的数组成员的位置，如果所有成员都不符合条件，则返回-1。

```
let index = [1, 2, 3, 4, 5, 6, 7].findIndex(function(value) {
            if (value % 2 == 0) {
                return value;
            }
        })
        console.log(index);
```

说明：findIndex()方法的回调函数也有三个参数，与find() 方法的回调函数参数含义一样。

3.findIndex( )方法原理

模拟 findIndex( )方法的实现：

```
Array.prototype.findIndexTest = function(fn) {
            for (let i = 0; i < this.length; i++) {
                let f = fn(this[i]);
                if (f) {
                    return i;
                }
            }
        }
        let result = [1, 2, 3, 4, 5, 6, 7].findIndexTest(function(item) {
            return item === 6
        })
        console.log(result);
```

博学谷
www.boxuegu.com

目录 Contents

fill( )方法的应用

◆ **使用给定的值填充数组**

```
let arr = new Array(3).fill(6);

console.log(arr);
```

创建一个长度为3的数组，用fill( )方法全部填充6。

注意：数组中已经有元素，那么，会被覆盖掉。

博学谷
www.boxuegu.com

目录 Contents

entries( ),keys( )和values( )方法应用

◆ **3个方法都用于遍历数组。**

keys( )： 对**键名**进行遍历

values( )：对**值**的遍历

entries( )：对**键值对**的遍历

```
let arr = ['a', 'b', 'c', 'd', 'e'];
    for (let index of arr.keys()) {
        console.log(index);
    }
    for (let item of arr.values()) {
        console.log(item);
    }
    for (let [index, item] of arr.entries()) {
        console.log(index + '-----' + item);
    }
```

博学谷
www.boxuegu.com

目录 Contents

includes( )方法的应用

- 该方法表示某个数组是否包含给定的值。

```
let arr = ['a', 'b', 'c’];

console.log(arr.includes('a’));

console.log(arr.includes('f'));
```

说明：如果数组中包含指定的值，返回 true，否则返回false。

博学谷
www.boxuegu.com

目录 Contents

1.filter( )方法基本使用

◆ 对数组中的数据进行过滤，把过滤后的数据放到一个新数组中。

示例：

要求把数组中大于等于60的数据过滤出来。

```
let array = [65, 56, 89, 53];

let arr = array.filter(function(item) {
    return item >= 60
})
console.log(arr);
```

2.filter( )方法的原理

模拟一个filter函数：

```
Array.prototype.filterOne = function(fn) {
   let newArray = [];
   for (let i = 0; i < this.length; i++) {
        let f = fn(this[i]);
        if (f) {
           newArray.push(this[i]);
        }
   }
   return newArray;
}
let array = [65, 56, 89, 53];
let arr = array.filterOne(function(item) {
    return item >= 60
})
console.log("arr=", arr);
```

目录 Contents

1.some( )方法基本使用

- 让数组中的每一个元素执行一次回调函数，在该回调函数中执行一些操作；

  只要有一个操作结果为真，会返回true，否则返回false。

示例：

检测数组中是否有元素大于10。

```
let array = [1, 3, 5, 7, 9];
let result = array.some(function(item) {
    return item > 10
})
console.log(result);
```

2.some( )方法原理

模拟一个some方法：

```
Array.prototype.someTest = function(fn) {

    for (let i = 0; i < this.length; i++) {
        let f = fn(this[i]);
        if (f) {
            return f;
        }
    }
    return false;
}

let array = [1, 3, 5, 7, 9];

let result = array.someTest(function(item) {
    return item > 10;
})

console.log("result=", result);
```

目录 Contents

1.every( )方法基本使用

- 该方法与some()方法不同，some( )方法只要有一个符合条件就返回true；
  every( ) 方法：数组中所有元素都要符合指定的条件，才会返回 true。

案例：

检测数组中的所有元素是否都大于 10 。

```
let array = [1, 3, 5, 7, 90];

let result = array.every(function(item) {
    return item > 10
})

console.log(result);
```

2.every( )方法原理

模拟一个every方法实现：

```
    Array.prototype.everyTest = function(fn) {
     let f = true;
     for (let i = 0; i < this.length; i++) {
         let f = fn(this[i]);
         if (!f) {
             // 只要有一个不符合，就立即返回false.
             return false;
         }
     }
     return f;
}
let array = [11, 31, 5, 71, 90];
let result = array.everyTest(function(item) {
    return item > 10
})
console.log("result=", result);
```

一样的在线教育，不一样的教学品质