JS数组及手动封装ES3的数组核心方法

jCodeLife
0.123 2020.08.09 21:48:39 字数 3,002 阅读 40

跟我们平时理解的数组一样，js数组也是一种数据存储结构，用于用单个变量名存储多个值。js数组的本质是基于对象，都是通过Aarry构造出来的。
首先，我们来看看怎么创建数组

数组的两种创建方式

1. 数组字面量

```
var arr = [1,2,3];
```

2. 通过new Array(length/content)创建数组

```
var arr = new Array(1,2,3)
```

不过有问题，如下：

```
var arr = new Array();
console.log(arr);//[]

var arr = new Array(5);
console.log(arr);  //[empty × 5]

var arr = new Array(-5);
console.log(arr);//RangeError

var arr = new Array("aa");
console.log(arr);//["aa"]

var arr = new Array(true);
console.log(arr);//[true]

var arr = new Array({});
console.log(arr);//[{}]

var arr = new Array(1,2,3,4);
console.log(arr);//[1,2,3,4]
```

我们可以看出，通过new Array来创建数组时，不同的参数导致不同结果，当传入参数为：

a. 无参数时，返回一个空数组

b. 单个正整数参数，会当成的新数组的长度，创建对应长度的稀松数组,每个位置的值都是undefined

c. 单个非正整数的数值作为参数时，会报错

d. 单个非数值参数（比如字符串、布尔值、对象等），则该参数是返回的新数组的成员

e. 多个参数时，才返回正常的数组，多个参数都是数组的成员

所以正是因为Array作为构造函数，行为很不一致。因此，在生成新数组，不建议使用new Array()的方式来创建数组，而是直接使用数组字面量。并且效果是一样的，相当于是语法糖

```
var arr1 = [1, 2, 3];
var arr2 = new Array(1, 2, 3);
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

接着，来看数组的读、写和删除操作

数组的读和写以及删除

1. 读取数组元素；

格式：arr[index]

```
var arr = [1,2,3];
arr[0];//1
arr[10];//可以溢出读，结果是undefined而已
```

可以溢出读，结果是undefined而已

2. 写入取数组元素

格式：arr[index] = xxx

```
var arr = [1,2,3];
arr[10] = 10;
console.log(arr.length);//11
console.log(arr);//[1, 2, 3, empty × 7, 10]
console.log(arr[5]);//undefined
```

可以溢出写，中间空的就为空值。读取空置的位置，结果为undefined，length变成对应长度

可以看出，js数组的读和写极其松散，可以溢出读，也可以溢出写，基本不会报错。溢出读时，值为undefined；溢出写时，中间空的就为空值，读取空置的位置，结果为undefined，length会对应变化

3. delete arr[num];

通过delete关键字可以删除数组中对应下标的元素

```
var arr = [1, 2, 3];
arr[10] = 10;
console.log(arr.length);//11
console.log(arr);//[1, 2, 3, empty × 7, 10]
console.log(arr[5]);//undefined

delete arr[10];
console.log(arr.length);//11
console.log(arr);//[1, 2, 3, empty × 8]
console.log(arr[5]);//undefined
```

这里可以看到通过溢出写的方式改变数组，接着通过delete删除对应写入的值时，数组的length不会发生改变了，被删除的值为空了。
再来看看正常删除一个数组中的元素

```
var arr = [1,2,3]
delete arr[0];
console.log(arr);// [empty, 2, 3]
console.log(arr.length);// 3
```

原来，通过delete的方式删除数组成员，并不会改变原数组的length值。相当于将对应位置的内容清空，位置还是被占据

另外可通过for in 可以遍历数组，如下：

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
```

index遍历表示数组下标

ES3中数组常用方法（核心方法）

因为数组方法较多，我们分类记忆。大致可分两类：

1. 可改变原数组的方法
   push、pop、shift、unshift、reverse、splice、sort
2. 不改变原数组的方法
   concat、join、split、toString、slice

可改变原数组的方法

`1. push()`

`用于在数组最后添加一个或多个元素，并返回添加新元素后的数组长度`

- 基本使用

```
var arr = [1,2,3];
console.log(arr.push(4,5));//5
console.log(arr)//[1,2,3,4,5]
```

- 手动封装

```
Array.prototype._push = function(){
    for(var i=0;i<arguments.length;i++){
        this[this.length] = arguments[i];
    }
    return this.length;
}
//test code
var arr = [1,2,3];
console.log(arr._push(4,5));//5
console.log(arr)//[1,2,3,4,5]
```

- 使用场景：
  (1) 合并数组

```
var a = [1,2];
var b = [3,4];
var res = Array.prototype.push.apply(a,b);
console.log(a)//[1,2,3,4]
console.log(b)//[3,4]
console.log(res)//4;就是a.length
```

或者

```
var a = ['a','b'];
var b = ['c','d'];
var res = a.push.apply(a, b)
console.log(a)//[1,2,3,4]
console.log(b)//[3,4]
console.log(res)//4;就是a.length
```

(2) push() 还可以用于向对象添加元素，添加后的对象变成类数组

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
console.log(obj);
```

`2. pop()`

用于删除数组最后一位元素，并返回该元素；相当于剪切

- 基本使用

```
var arr = [1,2,3,4]
console.log(arr.pop());//4
console.log(arr);//[1,2,3]
```

- 手动封装

```
Array.prototype._pop = function(){
    var result = this[this.length-1]
    delete this[this.length];
    this.length--;
    return result;
}
//test code
var arr = [1,2,3,4]
console.log(arr._pop());//4
console.log(arr);//[1,2,3]
console.log(arr.length);//3
```

注意：数组最后一位的下标是length-1

还有个小问题，原pop方法，如果是空数组[]时，不会报错，而是返回undefined。

完善代码

```
Array.prototype._pop = function () {
    if (this.length) {
        var result = this[this.length - 1]
        delete this[this.length];
        this.length--;
        return result;
    }
    return;
}
//test code
//对空数组使用pop方法，不会报错，而是返回undefined。
console.log([].pop());
console.log([]._pop());
```

利用push和pop方法，可以构成了“后进先出”的栈结构（stack）

`3. unshift()`

用于在数组的开头添加一个或多个元素，并返回添加新元素后的数组长度

- 基本使用

```
var arr = [1,2,3]
console.log(arr.unshift('a','b','c'));//6
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

- 手动封装

```
Array.prototype._unshift = function () {
    //将数组本来元素移到后面去
    for (var i = this.length - 1; i >= 0; i--) {
        this[i + arguments.length] = this[i];
    }
    //将要添加的元素，一个一个添加进腾出来的空位上
    for (var j = 0; j < arguments.length; j++) {
        this[j] = arguments[j];
    }
    return this.length;
}

var arr = [1, 2, 3]
console.log(arr._unshift('a', 'b', 'c'));//6
console.log(arr);//['a','b','c',1,2,3]
console.log(arr.length);//6
```

`4. shift()`

`用于删除数组的第一个元素，并返回该元素。跟pop一样相当于剪切`

- 基本使用：

```
var arr = [1, 2, 3]
console.log(arr.shift());//1
console.log(arr);//[2,3]
console.log(arr.length);//2
```

- 手动封装

```
Array.prototype._shift=function(){
    var res =  this[0];
    //让前一位等于后一位，将第一位覆盖掉。
    for(var index in this){
        this[index-1] = this[index]
    }
    return res;
}
//test code
var arr = [1, 2, 3]
console.log(arr.shift());//1
console.log(arr);//[2,3]
console.log(arr.length);//2
```

核心在于数组遍历时前一位等于后一位，将第一位覆盖掉，最后length也会自动-1.

`5. reverse()`

`用于颠倒数组中元素的顺序，返回改变后的数组`

- 基本使用

```
var arr = [1, 2, 3]
console.log(arr.reverse());//[3,2,1]
console.log(arr);//[3,2,1]
```

- 手动封装

`实现数组的翻转主要就是不断将左边第一个数和右边最后一个数调换位置，接着将左边第二个数和右边倒数第二个数调换位置，这样一直实现左边和右边数字位置的调换，直到左边数的位置刚好等于右边数的位置即停止`

推荐阅读

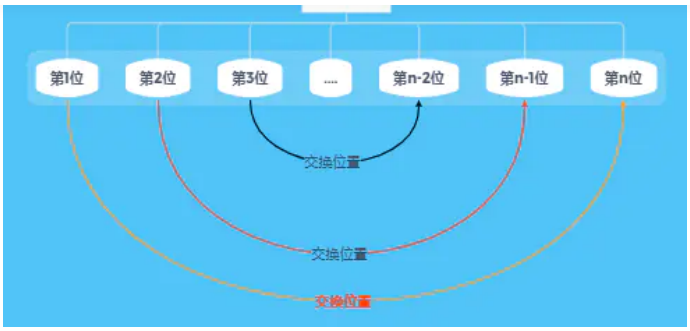
Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90



代码实现：

```
Array.prototype._reverse = function () {
    var left = 0;//存储左边第一个位置
    var right = this.length - 1;//存储右边最后一个位置
    while (left < right) {//停止进行的条件
        var temp = this[left];//利用一个中间变量来交换位置
        this[left] = this[right];
        this[right] = temp;
        left++;
        right--;

    }
}
var arr = [1, 2, 3, 4, 5, 6, 7]
console.log(arr._reverse());//undefined
console.log(arr);// [7, 6, 5, 4, 3, 2, 1]
```

6. splice()

截取并添加,返回被截取部分：添加在切口处添加，添加的新数据在原数组上

arr.splice(从第几位开始，截取的长度，在切口处添加的数据1，在切口处添加的数据2，...);

- 基本使用

```
var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,1,'a','b'));//[4]
console.log(arr);// [1, 2, 3, "a", "b", 5, 6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3))//[4,5,6]
console.log(arr);// [1, 2, 3]

var arr = [1,2,3,4,5,6]
console.log(arr.splice())//[]
console.log(arr);// [1,2,3,4,5,6]
```

如果只提供第一个参数，那就是从该位置开始截取到最后返回；等同于将原数组拆分成两数组

如果没有传参，等同于没有截取、没有添加，原数组不变，返回的空数组[ ]

以上是最常见的情况，但还封装一样的功能，得再来深入具体每种情况，那我们接下来试试当传入其他数据类型的时候，会发生什么？

传入一个值时，各种各样的值类型情况：

```
//一个参数值
var arr = [1,2,3,4,5,6]
console.log(arr.splice(3))//[4,5,6]
console.log(arr);// [1,2,3]

var arr = [1,2,3,4,5,6]
console.log(arr.splice("3"))//[4,5,6]
console.log(arr);// [1,2,3]
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
console.log(arr.splice("aaa"))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice(null))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice(undefined))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice(false))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice(true))//[2,3,4,5,6]
console.log(arr);// [1]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(0))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice([]))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice([1]))//[2,3,4,5,6]
console.log(arr);// [1]


var arr = [1,2,3,4,5,6]
console.log(arr.splice([3]))//[4,5,6]
console.log(arr);// [1,2,3]


var arr = [1,2,3,4,5,6]
console.log(arr.splice({}))//[1,2,3,4,5,6]
console.log(arr);// []

var arr = [1,2,3,4,5,6]
console.log(arr.splice(-3))//[4,5,6]
console.log(arr);// [1,2,3]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(100))//[]
console.log(arr);// [1,2,3,4,5,6]
```

我们发现，当传入一个正常的number数字时，那就是从该位置开始截取到最后返回；等同于将原数组拆分成两数组

当传入的是字符型数字时，会将它转成正常数字，然后从该位置开始截取到最后

当传入的是非数字型字符串、null、undefined、false、非一个数字的数组、对象时，都相当于arr.splice(0)，从第0位开始截取到最后

当传入的是单位数字的数组时，会将它变成数字，表示从该位置开始截取到最后

当传入的是true时，会将它变成1，表示从第一位开始截取到最后

可以传入负数，表示倒数位开始

可以传入大于arr.length的数字，返回截取部分是[]，表示没有截到

再看当传入两个值时，第二个值各种类型是

```
var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,1))//[4]
console.log(arr);// [1,2,3,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,'1'))//[4]
console.log(arr);// [1,2,3,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,'1aa'))//[]
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,-1))//[]
console.log(arr);// [1,2,3,4,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,100))//[4,5,6]
console.log(arr);// [1,2,3]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,false))//[]
console.log(arr);// [1,2,3,4,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,true))//[4]
console.log(arr);// [1,2,3,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,[]))//[]
console.log(arr);// [1,2,3,4,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,[1]))//[4]
console.log(arr);// [1,2,3,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,{}))//[]
console.log(arr);// [1,2,3,4,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,null))//[]
console.log(arr);// [1,2,3,4,5,6]

var arr = [1,2,3,4,5,6]
console.log(arr.splice(3,undefined))//[]
console.log(arr);// [1,2,3,4,5,6]
```

第二个参数表示截取位数：

普通数字就直接表示截取多少位，负数表示截取不到，超过数组长度的表示截取到最后一位

数字型字符串和单位数字数组会转成对应数字表示截取多少位

true和false会转成数字1和0，分别表示截取1位和0位

对象、非数字型字符串、非单位数字数组以及null、undefined都表示截取不到，返回[]

剩下从第三个参数开始就表示要添加的数据了。

经过上述的测试，我们发现，原来前面两个参数存在隐式类型转换，都是将传入的数据通过Number()转成了数字类型，再代入函数中。

这个函数会比较麻烦比较难理解，个人未实现。网上看到有源码大家可以参考：JS中从Array.slice()与Array.splice()的底层实现原理分析区别—作者：coder_chenz

7. sort()

用于对数组成员进行排序，默认是按照ASCII码顺序排序

注意。sort方法不是按照大小排序，而是按照对应字符串的字典顺序排序。也就是说，数值会被先转成字符串，再按照字典顺序进行比较，所以101排在11的前面。

如果想让sort方法按照自定义方式排序，可以传入一个函数作为参数，该函数本身又接受两个参数,表示进行比较的两个元素。如果比较返回大于0，表示第一个元素排在第二个元素后面；否则第一个元素排在第二个元素前面。

传入的函数的规则是：

1.必须写2个形参；

2.当返回值为负数时，那么前面的数放在前面；

当返回值为正数时，那么后面的数放在前面；

当返回值为0时，不动。

```
arr.sort(function(a,b){
    //return a-b;升序
    //return b-a;降序
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
arr.sort(function(){return Math.random()-0.5})
```

里面返回的是随机正负，所以是随机换位置或者不换位置实现乱序。
要实现这个sort函数会比较麻烦，个人未实现，我在网上找到V8引擎中sort的源码，给大家研究：

```
function InnerArraySort(array, length, comparefn) {
 // In-place QuickSort algorithm.
 // For short (length <= 22) arrays, insertion sort is used for efficiency.

 if (!IS_CALLABLE(comparefn)) {
   comparefn = function (x, y) {
     if (x === y) return 0;
     if (%_IsSmi(x) && %_IsSmi(y)) {
       return %SmiLexicographicCompare(x, y);
     }
     x = TO_STRING(x);
     y = TO_STRING(y);
     if (x == y) return 0;
     else return x < y ? -1 : 1;
   };
 }
 var InsertionSort = function InsertionSort(a, from, to) {
   for (var i = from + 1; i < to; i++) {
     var element = a[i];
     for (var j = i - 1; j >= from; j--) {
       var tmp = a[j];
       var order = comparefn(tmp, element);
       if (order > 0) {
         a[j + 1] = tmp;
       } else {
         break;
       }
     }
     a[j + 1] = element;
   }
 };

 var GetThirdIndex = function(a, from, to) {
   var t_array = new InternalArray();
   // Use both 'from' and 'to' to determine the pivot candidates.
   var increment = 200 + ((to - from) & 15);
   var j = 0;
   from += 1;
   to -= 1;
   for (var i = from; i < to; i += increment) {
     t_array[j] = [i, a[i]];
     j++;
   }
   t_array.sort(function(a, b) {
     return comparefn(a[1], b[1]);
   });
   var third_index = t_array[t_array.length >> 1][0];
   return third_index;
 }

 var QuickSort = function QuickSort(a, from, to) {
   var third_index = 0;
   while (true) {
     // Insertion sort is faster for short arrays.
     if (to - from <= 10) {
       InsertionSort(a, from, to);
       return;
     }
     if (to - from > 1000) {
       third_index = GetThirdIndex(a, from, to);
     } else {
       third_index = from + ((to - from) >> 1);
     }
     // Find a pivot as the median of first, last and middle element.
     var v0 = a[from];
     var v1 = a[to - 1];
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
        v1 = tmp;
      } // v0 <= v1.
      var c02 = comparefn(v0, v2);
      if (c02 >= 0) {
        // v2 <= v0 <= v1.
        var tmp = v0;
        v0 = v2;
        v2 = v1;
        v1 = tmp;
      } else {
        // v0 <= v1 && v0 < v2
        var c12 = comparefn(v1, v2);
        if (c12 > 0) {
          // v0 <= v2 < v1
          var tmp = v1;
          v1 = v2;
          v2 = tmp;
        }
      }
      // v0 <= v1 <= v2
      a[from] = v0;
      a[to - 1] = v2;
      var pivot = v1;
      var low_end = from + 1;   // Upper bound of elements lower than pivot.
      var high_start = to - 1;  // Lower bound of elements greater than pivot.
      a[third_index] = a[low_end];
      a[low_end] = pivot;

      // From low_end to i are elements equal to pivot.
      // From i to high_start are elements that haven't been compared yet.
      partition: for (var i = low_end + 1; i < high_start; i++) {
        var element = a[i];
        var order = comparefn(element, pivot);
        if (order < 0) {
          a[i] = a[low_end];
          a[low_end] = element;
          low_end++;
        } else if (order > 0) {
          do {
            high_start--;
            if (high_start == i) break partition;
            var top_elem = a[high_start];
            order = comparefn(top_elem, pivot);
          } while (order > 0);
          a[i] = a[high_start];
          a[high_start] = element;
          if (order < 0) {
            element = a[i];
            a[i] = a[low_end];
            a[low_end] = element;
            low_end++;
          }
        }
      }
      if (to - high_start < low_end - from) {
        QuickSort(a, high_start, to);
        to = low_end;
      } else {
        QuickSort(a, from, low_end);
        from = high_start;
      }
    }
  };
```

下面是不会改变原数组的方法

concat、toString、slice、join、split

1. concat()

用于连接两个或多个数组，并形成新数组，返回新数组。即用于多个数组的合并。

```
var arr1 = [1,2,3];
var arr2 = ['a','b','c'];
var arr = arr1.concat(arr2);
```

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

```
console.log(arr1.concat(111,222,333));// [1, 2, 3, 111, 222, 333]
console.log(arr2.concat(arr1,[3,3,3]))// ["a", "b", "c", 1, 2, 3, 3, 3, 3]
console.log(arr2.concat(arr1,[3,3,[4,4]]))// ["a", "b", "c", 1, 2, 3, 3, 3, [4,4]]
```

参数可以是一个一个的值，也可以是一个数组，传入的数组第一层会被拍平

2. toString()

用于返回数组的字符串形式，undefined、null会被认为是空，嵌套数组不管多少层一律会被拍平

```
var arr = [1,2,3];
console.log(arr.toString());//1,2,3

var arr = [1,2,3,['a','b','c']];
console.log(arr.toString());//1,2,3,a,b,c

var arr = [1,2,3,['a','b','c',[1111,2222]]];
console.log(arr.toString());//1,2,3,a,b,c,1111,2222

var arr = [1,,3,null,false,undefined,];
console.log(arr.toString());//1,,3,,false,
```

3. slice()

用于截取数组，传递两个参数分别表示从第几位截取到第几位，返回截取部分。

如果没有参数，从0开始截取截取到最后；即可实现数组的拷贝

传一个参数表示从该位开始，截取到最后

两个参数，即是从第几位开始截取，截取到第几位

```
var arr = [1,2,3,4,5];
console.log(arr.slice());//截取整个数组        [1,2,3,4,5];
console.log(arr.slice(2));//截取从第2位到最后        [3,4,5];
console.log(arr.slice(2,3));//截取第2位开始到第3位，注意不包括第3位哦     [3];
```

注意：

当slice(n,m)传递两个参数时，截取的东西是从第n位开始，截取到第m位，不包括第m位！！

splice功能也强大，但是不好的是，会改变原数组。通常我们想不破坏原数组，slice会较为常用

5. join()

用于将数组元素连成一个字符串，返回该字符串。元素间通过传参作为指定分隔符分割，不传参会按照","连接。

```
var arr = [1, 2, 3, 4, 5];
console.log(arr.join());//1,2,3,4,5
console.log(arr.join("-"));//1-2-3-4-5
console.log(arr.join(null));//1null2null3null4null5
```

4. split()

其实这是字符串的方法

但是与join方法可逆，join方法时通过指定分隔符连成字符串；split是将字符串通过指定分隔符拆分成数组。按照什么拆分，什么就没有了。

```
var arr = [1, 2, 3, 4, 5];
var str = arr.join("-");//1-2-3-4-5
console.log(str.split("-"));//["1", "2", "3", "4", "5"]
```

注意字符串通过split方法拆分成数组，数组元素仍然是字符串。

推荐阅读

Vue干货小技巧——学会再也不做加班狗
阅读 1,144

vue优化页面
阅读 701

面试了几个前端，给爷整哭了！
阅读 19,306

HTML5抽奖转盘-CSS3超简单版本
阅读 55

vue 3.0 的初次体验
阅读 90

JS数组及手动封装ES3的数组核心方法

JS中的实例方法与静态方法—作者：maxlove
JavaScript Array 对象—w3school
数组的shift和unshift方法的封装—作者：I Believe in
JavaScript常用方法push、pop、shift、unshift、concat、join的封装与使用—作者：麦麦麦麦兜
JS中自己封装方法，实现sort、reverse等方法—作者：Curry3Ooo
数据结构与算法——使用原生js实现js中自带的reverse()方法—作者：tozeroblog
JS中从Array.slice()与Array.splice()的底层实现原理分析区别—作者：coder_chenz
JS V8 引擎中sort的源码

jCodeLife 书山有路勤为径，学海无涯苦作舟
总资产13 (约1.21元) 共写了11.8W字 获得216个赞 共21个粉丝