深入11种数组循环和ES6数组新增的东西

jCodeLife

0.306 2020.09.02 18:17:33 字数 2,220 阅读 25

js数组的使用，一般都离不开循环；那我们首先先来梳理一下数组中常用的10种循环，再来讲ES6数组中新增的一些东西

1. 普通for循环

```
for(let i=0; i<arr.length; i++){
  console.log(arr[i])
}
```

以上为例来讲下普通for循环的执行过程：
① 执行let i = 0
② 判断i是否小于arr.length；满足条件返回true，不满足条件返回false，如果为true，循环结束，如果为false，执行循环体中语句
③ 执行循环体语句后，执行i++
④ 重复以上②③两个步骤，直到满足条件循环结束

普通for循环可以通过变量i来遍历获取数组中对应下标的数组成员

2. while循环

```
let arr = [1, 2, 3]
let i = 0;
while (i < arr.length) {
    console.log(arr[i]);
    i++;
}
```

while循环是一直重复判断，当()中的表达式结果转成boolean值值为真时，执行{}里面的代码内容，当转成boolean值值为假时，while循环结束

3. for in循环

for in 循环大多数情况是用于遍历对象，但是也能遍历数组，由于比较常用，所以也纳入进来

```
let arr = [1, 2, 3]
for (let i in arr) {
    console.log(arr[i]);
}
```

值得注意的是，for in枚举数组是key，后面会讲for of可以直接枚举到数组的value

...

4. forEach循环

```
arr.forEach(function(val, index, arr){
  console.log(val, index, arr);
});
```

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

- 第一个参数表示每次循环执行的回调函数
- 第二个参数表示this指向问题

其中回调函数接受三个参数，分别表示与循环圈数对应的value值、数组下标index以及原数组arr
需要注意的是，如果回调函数是箭头函数，那么通过第二个参数修改this执行时不能修改成功的。因为箭头函数this默认是执行外围非箭头函数执行的this，call和apply以及bind是不能修改的。

```
let arr = [1,2,3,4];
const obj = {name:"alice"}
arr.forEach((val,index,array)=>{
    console.log(this,val,index,array);//箭头函数不能通过第二个参数来修改其里头的this指向
},obj);
```

了解了基本使用，下面来手动封装：

```
Array.prototype._forEach = function (fn, thisTo) {
    for (let i = 0; i < this.length; i++) {
        fn.call(thisTo,this[i], i, this);
    }
}
//test code
let arr = [1,2,3,4];
const obj = {name:"alice"}
arr._forEach(function(val,index,array){
    console.log(this,val,index,array);//正常函数可以通过call来修改this指向
},obj);
arr._forEach((val,index,array)=>{
    console.log(this,val,index,array);//箭头函数不能通过第二个参数来修改其里头的this指向
},obj);
```

完美实现

下面再来看一个跟forEach很像的：map循环

5. map循环

map接受的参数跟forEach一模一样，第一个参数是回调函数，第二个是this指向，而且回调函数中的参数也是一样。
正常情况下，map需要配合return，返回是一个新的数组；若是没有return，用法相当于forEach。
所以map常常用于重新整理数组里头的数据结构

```
let arr = [
   {title:'aaaaa', read:100, hot:true},
   {title:'bbbb', read:100, hot:true},
   {title:'cccc', read:100, hot:true},
   {title:'dddd', read:100, hot:true}
];
let newArr = arr.map((item, index, arr)=>{
   let json={}
   json.t = `^_^${item.title}-----`;
   json.r = item.read+200;
   json.hot = item.hot == true && '真棒!!!';
   return json;
});
console.log(newArr);
```

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

下面再来封装一下map

```
Array.prototype._map = function(fn,thisTo){
    let res = []
    for(let i = 0;i<this.length;i++){
        res[i] = fn.call(thisTo,this[i],i,this)
    }
    return res;
}
//test code
let arr = [
   {title:'aaaaa', read:100, hot:true},
   {title:'bbbb', read:100, hot:true},
   {title:'cccc', read:100, hot:true},
   {title:'dddd', read:100, hot:true}
];
let newArr = arr._map((item, index, arr)=>{
   let json={}
   json.t = `^_^${item.title}-----`;
   json.r = item.read+200;
   json.hot = item.hot == true && '真棒!!!';
   return json;
});
console.log(newArr);
```

完美实现

6. filter

filter传参跟forEach也是一致，第一个参数是回调函数，第二个是this指向，回调函数的参数也是value、index、array。
filter是用于过滤一些不合格“元素”， 如果回调函数返回true，就留下来

```
let arr = [1,2,3,4,5,6,7,8,9,10];
let res = arr.filter(function(val,index,array){
    return val%2 == 0;
})
console.log(res)//[2,4,6,8,10]
```

下面来手动实现：

```
Array.prototype._filter = function (fn, thisTo) {
    let newArr = [];
    let key = 0;
    for (let i = 0; i < this.length; i++) {
        if (fn.call(thisTo, this[i], i, this)) {
            newArr[key] = this[i];
            key++
        }
    }
    return newArr;
}

let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
let res = arr._filter(function (val, index, array) {
    return val % 2 == 0;
})
console.log(res)
```

完美实现

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

some用于表示查找，只要数组里面某个元素符合条件，就返回true

```
let arr = [1,2,3,4,"a"];
let res = arr.some(function(val,index,array){
    return val === "a";
});
console.log(res);//true
```

手动封装也比较简单

```
Array.prototype._some=function(fn,thisTo){
    let res = false;
    for(let i = 0;i<this.length;i++){
        if(fn.call(thisTo,this[i],i,this)){
            res = true;
            break;
        }
    }
    return res;
}
//test code
let arr = [1,2,3,4,"b"];
let res = arr._some(function(val,index,array){
    return val === "a";
});
console.log(res);//false
```

完美实现

8. every

跟some类似，但不同的是some只需要某一个元素满足条件就返回true，而every是需要数组中所有元素都满足条件才返回true；

```
let ints = [1, 2, 3, 4];
let res = ints.every((val, index, array) => {
    return typeof val === 'number';
});
console.log(res);//true
```

手动封装every

```
Array.prototype._every = function (fn, thisTo) {
    let res = true;
    for (let i = 0; i < this.length; i++) {
        if (!fn.call(thisTo, this[i], i, this)) {
            res = false;
            break;
        }
    }
    return res;
}
let ints = [1, 1, 1, 1];
let res = ints._every((val, index, array) => {
    return val == 1;
});
console.log(res)//true
```

完美实现

小结：

以上五个方法forEach/map/filter/some/every传参都是一样，可以接收两个参数，分别是：循环回调函数和this指向，其

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

9. reduce

reduce回调函数中接受四个参数：
第一个参数：total表示初始值, 或者上次累积计算结束后的返回值
第二个参数：当前值value
第三个参数：下标index
第四个参数：原数组
例如可以使用reduce计算数组的和或者阶层

```
let arr = [1,2,3,4,5,6,7,8,9,10];
let res = arr.reduce((total, cur, index, arr) =>{
  return total+cur;
});
 console.log(res);//55
```

为了看清楚，我们干脆直接打印出来四个参数来看看

```
let arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
arr.reduce((total, cur, index, arr) => {
    console.log(total, cur, index, arr);
    return total + cur;
});
```

```
1 2 1 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
3 3 2 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
6 4 3 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
10 5 4 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
15 6 5 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
21 7 6 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
28 8 7 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
36 9 8 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
45 10 9 ▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

结果

可以看出来，每一次的tatal其实就是上一次的循环执行的返回结果
下面来看具体实现

```
Array.prototype._reduce = function (fn, thisTo) {
    let total = this[0];
    for (let i = 1; i < this.length; i++) {
        total = fn.call(thisTo, total, this[i], i, this)
    }
    return total;
}
let arr = [1, 2, 3, 4, 5];
let res = arr._reduce((total, cur, index, arr) => {
    console.log(total, cur, index, arr);//每一次的tatal就是上一次的循环执行的返回结果
    return total + cur;
});
console.log(res)
```

经过测试，成功实现

10. reduceRight

reduceRight与reduce极其相似，只不过reduce中的回调累积total的时候，是从左往右，而reduceRight是从右往左。

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

```
console.log(res)
```

```
5 4 3 ▶ (5) [1, 2, 3, 4, 5]
9 8 2 ▶ (5) [1, 2, 3, 4, 5]
12 2 1 ▶ (5) [1, 2, 3, 4, 5]
14 1 0 ▶ (5) [1, 2, 3, 4, 5]
15
```

结果

手动实现也很类似

```
Array.prototype._reduceRight = function (fn, thisTo) {
    let total = this[this.length-1];
    for (let i = this.length-1-1; i >= 0; i--) {
        total = fn.call(thisTo, total, this[i], i, this)
    }
    return total;
}
let arr = [1, 2, 3, 4, 5];
let res = arr._reduceRight((total, cur, index, arr) => {
    console.log(total, cur, index, arr);//每一次的tatal就是上一次的循环执行的返回结果
    return total + cur;
});
console.log(res)
```

没有问题
再来看一个比较特别的for....of循环

11. for of循环

for...of适用遍历数/数组对象/字符串/map/set等拥有迭代器对象的集合，这里只以数组为例

```
let arr = [1, 2, 3, 4, 5];
for (let val of arr) {
    console.log(val);//1,2,3,4,5
}
```

默认遍历的是value值

以上是11中数组循环的全部内容

下面来看在ES6中数组身上新增的东西，一起来看看

ES6新增数组创建方法

1. Array.from()

作用: 把类数组(获取一组元素、arguments...) 对象转成数组
个人观点： 具备 length这个东西，就靠谱
Array.from的设计目的是快速便捷把一个类似数组的可迭代对象创建成一个新的数组实例。
返回新的数组，而不改变原对象。

2. Array.of():

作用：把一组值，转成数组

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

Array.of 将参数依次转化为数组中的一项，然后返回这个新数组，不管这个参数是数字还是其它什么。
Array.of 总是返回参数值组成的数组。如果没有参数，就返回一个空数组。

新增数组修改方法

1. copyWithin()

使用该方法会修改当前数组;
可以在当前数组内部，将指定位置的数组项复制到其他位置，会覆盖原数组项，然后返回当前数组;
它接受三个参数：
（1）target（必需）：从该位置开始替换数据。如果为负值，表示倒数。
（2）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。
（3）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。

2. fill(填充的东西，开始位置，结束位置不含此位置)

使用给定值，填充一个数组。

```
[1,2,3,4,5].fill('a');
// ["a", "a", "a", "a", "a"]
new Array(3).fill(12)
// [12, 12, 12]
```

可以接受第二个和第三个参数，用于指定填充的起始位置和结束位置。
注意，如果填充的类型为对象，那么被赋值的是同一个内存地址的对象，而不是深拷贝对象。

新增数组查找遍历方法

1. arr.find()

查找，找出第一个符合条件的数组成员，如果没有找到，返回undefined

2. arr.findIndex():

找的是位置， 没找到返回-1

3. includes()

包不包含,返回一个布尔值
表示某个数组是否包含给定的值，与字符串的 includes 方法类似;
includes 第二个参数表示搜索的起始位置
如果第二个参数为负数，则表示从倒数第几位向后搜索

4. entries、keys、values

它们都返回一个遍历器对象，都可以用 for...of 循环进行遍历。
唯一的区别是 keys 是对键名的遍历、 values 是对键值的遍历， entries 是对键值对的遍历。

```
for (let index of ['a', 'b', 'c'].keys()) {
  console.log(index);//0 1 2
}
for (let elem of ['a', 'b', 'c'].values()) {
  console.log(elem);//'a', 'b', 'c'
}
```

推荐阅读

Es6基础语法
阅读 796

web前端达到什么水平，才能找到工作？
阅读 300

我：CSS垂直居中还有什么另类方法？求职者：不太了解了
阅读 407

Vue 之 组件深入
阅读 132

vue + webSocket 实时任务信息通知
阅读 402

数组降维方法

1. flat(num/Infinity)

flat用于将嵌套的数组“拉平”“拍平”。该方法返回一个新数组，对原数据没有影响。
参数表示嵌套层数。

2. flatMap()

方法对原数组的每个成员执行一个函数（相当于执行 Array.prototype.map），然后对返回值组成的数组执行 flat() 方法。
该方法返回一个新数组，不改变原数组。
注意：flatMap() 只能展开一层数组。

以上就是ES6数组新增的东西，最后总结一下：

```
1. 数组新增的方法，一方面起到了增强型作用，一方面让代码变得更加简洁
2. 其中 Array.from 和 Array.of 属于构造函数方法
3. 从是否改变数组自身的角度看：
copyWithin、fill 会改变数组自身；
includes、flat、flatMap不会改变数组自身；
entries、keys、values、find、findeIndex属于数组遍历方法
```


jCodeLife 书山有路勤为径，学海无涯苦作舟
总资产22 (约1.80元) 共写了13.5W字 获得295个赞 共23个粉丝