目录

**[目录 2](#_Toc17917)**

**[基础变化 2](#_Toc22204)**

[1. 变量声明let和const 2](#_Toc1335)

[2. 用${}来将字符串格式化。 2](#_Toc12239)

[3. 字符串处理 2](#_Toc17761)

[4. 方法参数可提供默认值 2](#_Toc24140)

[5. 可省略冒号与 function 2](#_Toc31130)

[6. Object.assign()这个方法来实现浅复制 3](#_Toc23727)

[7. 解构 3](#_Toc7904)

**[一、 箭头函数Arrow Function 3](#_Toc13998)**

[1. fliter //todo 3](#_Toc12823)

[2. forEach、map 3](#_Toc7568)

[3. some //todo 3](#_Toc3960)

[创建一个新的数组，不会改变原始数组的值。 3](#_Toc29817)

[forEach是没有返回值。 3](#_Toc20025)

[map有返回值，可以return 出来。 3](#_Toc15328)

[其它方面基本都一样 3](#_Toc682)

[遍历一个数组，对象型数组同样适用 3](#_Toc15804)

[当有3个变量时，第一个为当前item，第二个为序号，第三个为原变量 4](#_Toc25127)

**[二、 4](#_Toc10295)**

**[三、 展开语法Spread Operator 4](#_Toc24447)**

# 

# 基础变化

## 变量声明let和const

let表示变量、const表示常量。

let和const都是块级作用域。说白了 {}大括号内的代码块即为let 和 const的作用域。

## 用${}来将字符串格式化。

备注：这里不是单引号，是1左边的那个

const name = 'lux'

console.log(`hello ${name}`)

## 字符串处理

1.includes：判断是否包含然后直接返回布尔值 let str = 'hahay' console.log(str.includes('y')) // true

2.repeat: 获取字符串重复n次 let s = 'he' console.log(s.repeat(3)) // 'hehehe'

3.如果你带入小数, Math.floor(num) 来处理

## 方法参数可提供默认值

 function action(num = 200){  
        console.log(num)   
    }

## 可省略冒号与 function

const people = {   
        name: 'lux', getName: function() {  
            console.log(this.name)   
        }  
    }

可写为

const people = {   
        name: 'lux', getName (){  
            console.log(this.name)   
        }   
    }

## Object.assign()这个方法来实现浅复制

 const obj = Object.assign({}, objA, objB)

{}为对象型

[]为数组型

## 解构

对象:

const people = { name: 'lux', age: 20 }   
const { name, age } = people   
console.log(`${name} --- ${age}`)

数组:

const color = ['red', 'blue']   
const [first, second] = color   
console.log(first) //'red'   
console.log(second) //'blue'

## 用同步的方式去写异步代码//todo

发起异步请求：fetch('/api/todos') .then(res => res.json()).then(data =>({ data })).catch(err => ({ err }));

## 生成器（ generator）是能返回一个迭代器的函数//todo

# 箭头函数Arrow Function

数组常用方法fliter,map,forEach,some

## fliter

filter() 方法创建一个新的数组，新数组中的元素是通过检查指定数组中符合条件的所有元素。

注意： filter() 不会对空数组进行检测。

注意： filter() 不会改变原始数组。

格式为：数组.filter(条件方法);

let arr = [  
 {'title':'标题标题标题标题','status':0},  
 {'title':'标题标题标题标题','status':1},  
 {'title':'标题标题标题标题','status':2},  
 {'title':'标题标题标题标题','status':3},  
 {'title':'标题标题标题标题','status':4},  
]  
function checkAdult(arr) {  
 return arr.status !=2;  
}  
var b= arr.filter(checkAdult);  
console.log(b)

## forEach

//data中定义

data() {  
 return {  
 arr: ['a1', 'a2', 'a3', 'a4'],  
 }  
},

//方法中遍历 只有一个变量

this.arr.forEach(item => {  
 alert(item)  
})

当有3个变量时，第一个为当前item，第二个为序号，第三个为原变量

变量.map((item ,index,arr) => {  
 console.log(item)  
 console.log(index)  
 console.log(arr)  
}

## Map

map有返回值，可以return 出来。

let arr = [

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

{'title':'标题标题标题标题','status':0},

]

let a = arr.map(item =>{

return {

title:item['title']

}

})

console.log(a)}

## some

some() 方法用于检测数组中的元素是否满足指定条件（函数提供）。

some() 方法会依次执行数组的每个元素：

如果有一个元素满足条件，则表达式返回true , 剩余的元素不会再执行检测。

如果没有满足条件的元素，则返回false。

注意： some() 不会对空数组进行检测。

注意： some() 不会改变原始数组。

var ages = [3, 10, 18, 20];

function checkAdult(age) {

return age > 19;

}

var b = ages.some(checkAdult);

# 展开语法Spread Operator

## 组装对象或者数组

数组:

const color = ['red', 'yellow']

const colorful = [...color, 'green', 'pink']

console.log(colorful) //[red, yellow, green, pink]

对象:

const alp = { fist: 'a', second: 'b'}

const alphabets = { ...alp, third: 'c' }

console.log(alphabets) //{ "fist": "a", "second": "b", "third": "c" }

## 获取数组或者对象除了前几项或者除了某几项的其他项

数组：

number = [1,2,3,4,5]

const [first, ...rest] = number

console.log(rest) //2,3,4,5

对象：

const user = { username: 'lux', gender: 'female', age: 19, address: 'peking' }

const { username, ...rest } = user

console.log(rest) //{"address": "peking", "age": 19, "gender": "female" }

## 组合成新的Object

备注：如果有重复的属性名，右边覆盖左边

 const first = { a: 1, b: 2, c: 6, }

const second = { c: 3, d: 4 }

const total = { ...first, ...second }

console.log(total) // { a: 1, b: 2, c: 3, d: 4 }

# import导入模块、export导出模块//todo