JS使用技巧与函数积累

# JS使用技巧

## 默认值

1. /\*\* 可以采用默认值强化参数\*/

const required = () => {throw new Error('missing parameter')}

const add = (a=required(), b=3) => a+b;

🡪add(1,2) //3

🡪add(1) //4

🡪add() //Uncaught Error: missing parameter

## 映射加过滤(reduce)

* reduce函数可以代替map和filter函数

1. /\*\* 映射和过滤数组\*/

const numbers = [10,20,30,40]

const doubleOver50 = numbers.reduce((finalList,num)=>{

num = num \* 2;

if(num > 50){

finalList.push(num);

}

return finalList;

},[]);

🡪 doubleOver50 // [60,80]

1. /\*\* reduce统计数字中的吃饭数据\*/

var cars = ['BMW','Benz','Benz','Telsa','BMW','Toyota']

var carsobj = cars.reduce(function(obj,name){

obj[name] = obj[name] ? ++obj[name] : 1;

return obj;

},{});

🡪 carsobj //{BMW: 2, Benz: 2, Telsa: 1, Toyota: 1}

## 对象解构

1. /\*\* 删除不必要的属性\*/

let {\_i,t,...c} = {e1:'1',\_i:'s',t:{},e2:'2',e3:'3'};

🡪c // {e1: "1", e2: "2", e3: "3"}

1. /\*\* 在函数参数中使用解构\*/

const car = {

model:'bmw 2018',

engine:{

V6:true,

turbo:true,

vin:12345

}}

const modelAndVin = ({model,engine:{vin}}) => {console.log(`model:${model} vin:${vin}`);}

🡪 modelAndVin(car) // model:bmw 2018 vin:12345

1. /\*\* 合并对象(扩展运算符:…)\*/

let o1 = {a:1,b:2,c:3}

let o2 = {b:20,c:30,d:40}

let merged = {...o1,...o2}

🡪 merged // {a: 1, b: 20, c: 30, d: 40}

1. /\*\* 使用Sets去重\*/

let arr = [1,2,1,2,3,4,2,5,6]

let deduped = [...new Set(arr)]

🡪 deduped //(6) [1, 2, 3, 4, 5, 6]

## 数组解构

1. /\*\* 交换值\*/

let p1 = 1

let p2 = 2

[p1,p2] = [p2,p1]

🡪 p1 //2

🡪P2 //1

1. /\*\* 从一个函数中接收和赋值多个值\*/

async function grtFullPost(){

return await Promise.all([

fetch('/post'),

fetch('/comment')

]);

}

const [post,comment] = grtFullPost();

## 遍历函数(foreach)

1. /\*\* Fearch函数没有返回值，只能遍历数组\*/

var ary = [12,23,24,42,1];

ary.forEach(function (item,index,input) {

input[index] = item\*10;

})

1. /\*\* Jquery写法\*/

$.each(ary,function(item,index,input){

input[index] = item\*10;

})

🡪ary // (5) [120, 230, 240, 420, 10]

## 映射函数(map)

1. /\*\* map函数有返回值，比Fearch强大一些，只能映射数组\*/

var ary = [12,23,24,42,1];

var res = ary.map(function (item,index,input) {

input[index] = item\*10;

return item\*10;

});

🡪res // (5) [120, 230, 240, 420, 10]

## 过滤函数(filter)

1. /\*\* filter将返回值为true的元素装入一个新数组返回\*/

var arr = [1,5,20,8,9];

var narr= arr.filter(function(item,index,array){

array [index] = item \* 10;

if(item >6){

return true;

}

return false;

})

🡪arr // (5) [10, 50, 200, 80, 90]

🡪narr // (3) [20, 8, 9]

## 字符串与JSON转换

* 除了eval()函数是js自带的之外，其他的几个方法都来自json.js包。新版本的 JSON 修改了 API，将 JSON.stringify() 和 JSON.parse() 两个方法都注入到了 Javascript 的内建对象里面，前者变成了 Object.toJSONString()，而后者变成了 String.parseJSON()

1. /\*\* 字符串转JSON\*/

var obj = eval('(' + str + ')');

var obj = str.parseJSON();

var obj = JSON.parse(str);

1. /\*\* JSON转字符串\*/

var last=obj.toJSONString();

var last=JSON.stringify(obj);

# JS函数积累

* 新建构造函数

function

## 判断对象是否为空

$.