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

## 判断对象是否为空

/\*\*

\* 判断对象为空函数

\* @author 莫非(muphy) 博客：http://www.muphy.me/

\* @method isNull

\* @param obj

\* @return

\*/

$.isNull = function (obj) {

if (!obj && typeof obj !== 'number') {

return true;

} else if (obj instanceof Array && obj.length === 0) {

return true;

} else if (typeof obj === 'string' && obj.trim() === '') {

return true;

} else if (typeof obj === 'object') {

for (let k in obj) {

return false;

}

return true;

}

return false;

};

## 转换空对象为指定值

/\*\*

\* 转换空对象为指定值

\* @author 莫非(muphy) 博客：http://www.muphy.me/

\* @method isNull

\* @param obj

\* @return

\*/

$.Nvl = function (obj, d = '') {

if ($.isNull(obj)) {

return d;

}

return obj;

};

## 自定义fetch函数请求

(function () {

'use strict'

//兼容性处理forEach函数

if (!Array.prototype.forEach) {

Array.prototype.forEach = function (callback, thisArg) {

var T, k;

if (this == null) {

throw new TypeError(" this is null or not defined");

}

var O = Object(this);

var len = O.length >>> 0; // Hack to convert O.length to a UInt32

if ({}.toString.call(callback) != "[object Function]") {

throw new TypeError(callback + " is not a function");

}

if (thisArg) {

T = thisArg;

}

k = 0;

while (k < len) {

var kValue;

if (k in O) {

kValue = O[k];

callback.call(T, kValue, k, O);

}

k++;

}

};

}

//兼容性处理indexOf函数

if (!Array.prototype.indexOf) {

Array.prototype.indexOf = function (elt /\*, from\*/) {

var len = this.length >>> 0;

var from = Number(arguments[1]) || 0;

from = (from < 0)

? Math.ceil(from)

: Math.floor(from);

if (from < 0)

from += len;

for (; from < len; from++) {

if (from in this &&

this[from] === elt)

return from;

}

return -1;

};

};

/\*\*

\* 网络请求方法，该方法仅支持IE8以上

\* @author 莫非(muphy) 博客：http://www.muphy.me/

\* @method $requst

\* @param url：请求地址

\* @param options = {

\* catchs: 异常处理，控制台抛出的异常是否自己处理：true 是，false 否 由公共方法统一处理优化显示给用户 默认 false

\* credentials: 请求带上cookies，是每次请求保持会话一直

\* method: 请求使用的方法，如 GET、POST

\* headers: 请求的头信息，形式为 Headers 对象或 ByteString。

\* body: 请求的 body 信息：可能是一个 Blob、BufferSource、FormData、URLSearchParams 或者 USVString 对象。注意 GET 或 HEAD 方法的请求不能包含 body 信息。

\* mode: 请求的模式，如 cors、no-cors 或者same-origin。是否允许跨域请求

\* cache: 请求的 cache 模式: default, no-store, reload, no-cache, force-cache, or only-if-cached.

\* }

\* @return

\*/

var $requst = function (url, options) {

var myHeaders = {

'Content-Type': 'text/html; charset=utf-8'

};

var init = {

credentials: 'include',

method: (options && options.method) || 'GET',

headers: (options && options.headers) || myHeaders,

cache: (options && options.cache) || 'default'

}

if (options && options.body) {

init.body = JSON.stringify(options.body)

}

return fetch(url, init)

.then(function (response) {

if (response.ok) {

return \_returnContentByType(response);

} else {

if (options && options.catchs) {

throw new Error(response.statusText);

} else {

var error = new Error(response.statusText);

throw new Error('');

}

}

});

};

/\*\*

\* 根据type返回不同格式的response

\*/

var \_returnContentByType = function (response) {

var type = response.headers.get('Content-Type').split(";")[0];

switch (type) {

case 'text/html':

return response.text();

break

case 'application/json':

return response.json();

break

default:

}

};

})();