其他

2019年12月25日

9:47

* + **解构** : 从理解上,结构就是ES6提供的数组和对象进行整体赋值的新方案
    1. 数组的元素是按次序排列的,变量的取值由其位置决定
    2. 对象的位置没有次序,变量必须和属性同名,才能取到正确的值

**例子:**

数组: let [x, y]= [1, 2];//x=1,y=2

对象: let { foo , bar } = { foo: "aaa", bar: "bbb" };// foo = "aaa",bar = "bbb"

**用途:**

* + 交换变量的值

let x = 1;

let y = 2;

[x, y] = [y, x];

* + 从函数返回多个值

// 返回一个数组

function example() {

return [1, 2, 3];

}

let [a, b, c] = example();

* + 函数参数的定义

// 参数是一组有次序的值

function f([x, y, z]) { ... }

f([1, 2, 3]);

// 参数是一组无次序的值

function f({x, y, z}) { ... }

f({z: 3, y: 2, x: 1});

* + 提取JSON的值

var jsonData = {

id: 42,

status: "OK",

data: [867, 5309]

};

var { id, status, data: number } = jsonData;

* + 遍历Map结构
  + 输入模块的指定方法

import {ReactComponent} from './xxxComponent.jsx';

* + **扩展运算符**

扩展运算符由三个点号组成,用来将数组或类数组对象展开成一系列用逗号隔开的值

**例子:**

var foo = function(a,b,c){}

var arr = [1,2,3];

//传统写法

foo(arr[0],arr[1],arr[2])

//使用扩展运算符

foo(…arr);

**用法:**

* + 数组深拷贝

var arr2 = arr;

var arr3 = [...arr];

console.log(arr===arr2); //true, 说明arr和arr2指向同一个数组

console.log(arr===arr3); //false, 说明arr3和arr指向不同数组

* + 把一个数组插入另一个数组

var arr4 = [...arr, 4, 5, 6];

console.log(arr4);//[1, 2, 3, 4, 5, 6]

* + 字符串转成数组

var str = 'love';

var arr5 = [...str];

console.log(arr5);//[ 'l', 'o', 'v', 'e' ]

* + **rest运算符** : rest运算符实际上跟扩展运算符是一样的,放到某个变量前来定义该变量为数组

例子:

var [a, ...rest] = [1, 2, 3, 4];

console.log(a);//1

console.log(rest);//[2, 3, 4]

* + **clearTimeout以及clearInterval,用于取消定时事件**

例子:

var aaa = setTimeout(function(){},time);

clearTimeout(aaa);

var bbb = setInterval(function(){},time);

setInterval(bbb);

* + **手动触发事件**

try {

let evt = document.createEvent('Event');

evt.initEvent('click', true, true);

self.clickReact = false;

self.elem.dispatchEvent(evt);

} catch (e) {

console.log(e)

}

* + InterSectorObserver(注意 : 此属性当前仅有chrome支持)

const callback= function(entries,observer){

console.log(entries);

};

const option ={

root: document.querySelector('.SMSContainer'),

rootMargin: '0px',

threshold:0

};

let observer = new IntersectionObserver(callback,option);

observer.observe(document.querySelectorAll('.list')[5])

* + 特殊字符的转义和反转义

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | //HTML转义  function HTMLEncode(html) {  var temp = document.createElement("div");  (temp.textContent != null) ? (temp.textContent = html) : (temp.innerText = html);  var output = temp.innerHTML;  temp = null;  return output;  }    //HTML反转义  function HTMLDecode(text) {  var temp = document.createElement("div");  temp.innerHTML = text;  var output = temp.innerText || temp.textContent;  temp = null;  return output;  }  var tagText = "<p><b>123&456</b></p>";  var encodeText = HTMLEncode(tagText);  console.log(encodeText);//&lt;p&gt;&lt;b&gt;123&amp;456&lt;/b&gt;&lt;/p&gt;  console.log(HTMLDecode(encodeText)); //<p><b>123&456</b></p> |

* + **JavaScript 获取CSS媒体查询信息**
    1. **获取对象**

var result = window.matchMedia('(max-width: 700px)');

* + **添加监听事件**

// 指定回调函数

mql.addListener(mqCallback);

function mqCallback(mql) {

if (mql.matches) {

// 宽度小于等于700像素

} else {

// 宽度大于700像素

}

}

* + **移除监听事件**

mql.removeListener(mqCallback);

* + **获取对象的属性信息**

<<获取对象属性.txt>>

* + **阻止元素被选中及清除选中的方法**

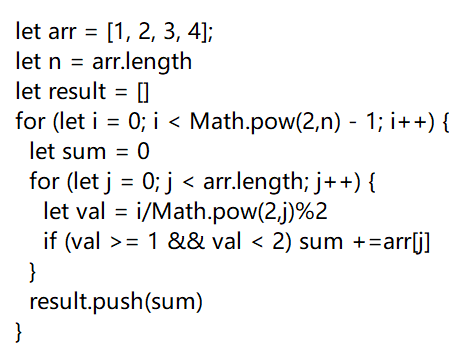
<<阻止元素被选中及清除选中的方法.pdf>>

* + **Commonjs 与ES6的区别**
    1. 导出导入不同
    2. 值的类型不同
       - Commonjs导出的是值的拷贝,值会被缓存。
       - ES6输出的是值的引用，不会被缓存，脚本真正运行的时候，根据引用到相应的模块去取值。
    3. 加载时间不同
       - Commonjs是运行时加载
       - ES6是编译时输出接口
    4. 加载内容不同
       - Commonjs加载整个模块
       - ES6可以单独加载某个接口
    5. this指向不同
       - Commonjs指向的是当前模块
       - ES6指向的是undefined

注意：export default导出的值是拷贝



* + 算法-遍历2^N次



* + 为什么要跨域?

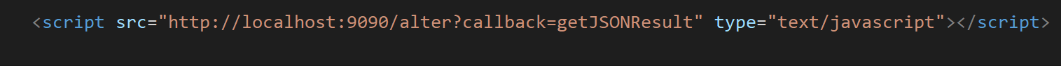
禁止获取其他域名下的网站的cookie, session等信息, 避免用户信息泄漏

nginx是否对用户安全有影响?

nginx不会带有被访问域名的身份信息, 因此是安全的

有什么跨域的办法

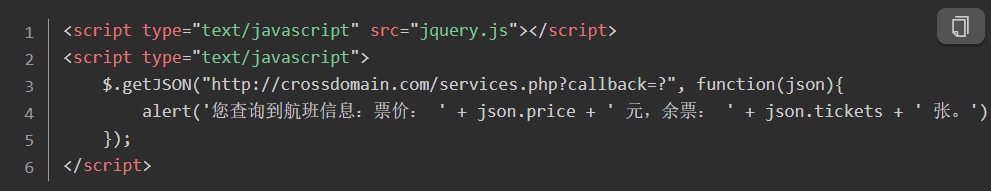
* + Nginx反向代理
  + JSONP



或者



或者



* + CORS跨域资源共享

浏览器在响应头添加允许的源