1. 拷贝原理

Javascript内的引用类型数据（array，object，function），在内存中是独立存在的一段数据，对于字符串类型，浅复制是对值的复制，对于对象来说，浅复制是对对象地址的复制。对象类型的直接赋值，实际上是赋值的一个引用，或者说只是赋值了一个地址，对象的数据实际上并没有直接放在这个变量所代表的内存空间，该内存空间只是存放了一个地址，这个地址里面索引到的部分才是真正的数据，当对对象进行直接赋值，实际上是把变量所代表的内存空间里面存放的地址给赋值到了新变量，实际指向的数据依然是同一份数据。而深复制则是开辟新的栈，两个对象对应两个不同的地址。

1. 浅拷贝的实现
2. 直接赋值

var a = 1;

var b = a;

1. 遍历对象重新赋值

var obj = { a:1, arr: [2,3] };

var shallowObj = shallowCopy(obj);

function shallowCopy(src) {

var dst = {};

for (var prop in src) {

if (src.hasOwnProperty(prop)) {

dst[prop] = src[prop];

}

}

return dst;

}

1. 深拷贝的实现
2. 递归

var china = {

nation : '中国',

birthplaces:['北京','上海','广州'],

skincolr :'yellow',

friends:['sk','ls']

}

//深复制，要想达到深复制就需要用递归

function deepCopy(o,c){

var c = c || {}

for(var i in o){

if(typeof o[i] === 'object'){

//要考虑深复制问题了

if(o[i].constructor === Array){

//这是数组

c[i] =[]

}else{

//这是对象

c[i] = {}

}

deepCopy(o[i],c[i])

}else{

c[i] = o[i]

}

}

return c

}

var result = {name:'result'}

result = deepCopy(china,result)

console.dir(result)

1. 通过JSON API来解析

var test ={

name:{

xing:{

first:'张',

second:'李'

},

ming:'老头'

},

age :40,

friend :['隔壁老王','宋经纪','同事']

}

var result = JSON.parse(JSON.stringify(test))

result.age = 30

result.name.xing.first = '往'

result.friend.push('fdagldf;ghad')

console.dir(test)

console.dir(result)

1. 其他框架实现

jQuery.extend = jQuery.fn.extend = function() {

var src, copyIsArray, copy, name, options, clone,

target = arguments[0] || {},

i = 1,

length = arguments.length,

deep = false;

// Handle a deep copy situation

if ( typeof target === "boolean" ) {

deep = target;

// skip the boolean and the target

target = arguments[ i ] || {};

i++;

}

// Handle case when target is a string or something (possible in deep copy)

if ( typeof target !== "object" && !jQuery.isFunction(target) ) {

target = {};

}

// extend jQuery itself if only one argument is passed

if ( i === length ) {

target = this;

i--;

}

for ( ; i < length; i++ ) {

// Only deal with non-null/undefined values

if ( (options = arguments[ i ]) != null ) {

// Extend the base object

for ( name in options ) {

src = target[ name ];

copy = options[ name ];

// Prevent never-ending loop

if ( target === copy ) {

continue;

}

// Recurse if we're merging plain objects or arrays

if ( deep && copy && ( jQuery.isPlainObject(copy) || (copyIsArray = jQuery.isArray(copy)) ) ) {

if ( copyIsArray ) {

copyIsArray = false;

clone = src && jQuery.isArray(src) ? src : [];

} else {

clone = src && jQuery.isPlainObject(src) ? src : {};

}

// Never move original objects, clone them

target[ name ] = jQuery.extend( deep, clone, copy );

// Don't bring in undefined values

} else if ( copy !== undefined ) {

target[ name ] = copy;

}

}

}

}

// Return the modified object

return target;

};

jQuery.extend(true, { a : { a : "a" } }, { a : { b : "b" } } );

jQuery.extend( { a : { a : "a" } }, { a : { b : "b" } } );

1. 存在的问题

深克隆都没考虑循环引用问题，以及对象内函数的闭包问题。