# [js实现深拷贝](https://www.cnblogs.com/rusr/p/8984604.html)

## type函数

首先我们要实现一个getType函数对元素进行类型判断，直接调用Object.prototype.toString 方法。

[复制代码](javascript:void(0);)

function getType(obj){

//tostring会返回对应不同的标签的构造函数

var toString = Object.prototype.toString;

var map = {

'[object Boolean]' : 'boolean',

'[object Number]' : 'number',

'[object String]' : 'string',

'[object Function]' : 'function',

'[object Array]' : 'array',

'[object Date]' : 'date',

'[object RegExp]' : 'regExp',

'[object Undefined]': 'undefined',

'[object Null]' : 'null',

'[object Object]' : 'object'

};

if(obj instanceof Element) {

return 'element';

}

return map[toString.call(obj)];

}

[复制代码](javascript:void(0);)

## 深拷贝(deepClone)

对于一个引用类型，如果直接将它赋值给另一个变量，由于这两个引用指向同一个地址，这时改变其中任何一个引用，另一个都会受到影响。当我们想复制一个对象并且切断与这个对象的联系，就要使用深拷贝。对于一个对象来说，由于可能有多层结构，所以我们可以使用递归来解决这个问题

[复制代码](javascript:void(0);)

function deepClone(data){

var type = getType(data);

var obj;

if(type === 'array'){

obj = [];

} else if(type === 'object'){

obj = {};

} else {

//不再具有下一层次

return data;

}

if(type === 'array'){

for(var i = 0, len = data.length; i < len; i++){

obj.push(deepClone(data[i]));

}

} else if(type === 'object'){

for(var key in data){

obj[key] = deepClone(data[key]);

}

}

return obj;

}

[复制代码](javascript:void(0);)

对于function类型，这里是直接赋值的，还是共享一个内存值。这是因为函数更多的是完成某些功能，有个输入值和返回值，而且对于上层业务而言更多的是完成业务功能，并不需要真正将函数深拷贝。

## 广度优先遍历

上面是使用递归来进行深拷贝，显然我们可以使用树的广度优先遍历来实现

[复制代码](javascript:void(0);)

//这里为了阅读方便，只深拷贝对象，关于数组的判断参照上面的例子

function deepClone(data){

var obj = {};

var originQueue = [data];

var copyQueue = [obj];

//以下两个队列用来保存复制过程中访问过的对象，以此来避免对象环的问题（对象的某个属性值是对象本身）

var visitQueue = [];

var copyVisitQueue = [];

while(originQueue.length > 0){

var \_data = originQueue.shift();

var \_obj = copyQueue.shift();

visitQueue.push(\_data);

copyVisitQueue.push(\_obj);

for(var key in \_data){

var \_value = \_data[key]

if(typeof \_value !== 'object'){

\_obj[key] = \_value;

} else {

//使用indexOf可以发现数组中是否存在相同的对象(实现indexOf的难点就在于对象比较)

var index = visitQueue.indexOf(\_value);

if(index >= 0){

// 出现环的情况不需要再取出遍历

\_obj[key] = copyVisitQueue[index];

} else {

originQueue.push(\_value);

\_obj[key] = {};

copyQueue.push(\_obj[key]);

}

}

}

}

return obj;

}

[复制代码](javascript:void(0);)

## JSON

深拷贝对象还有另一个解决方法，在对象中不含有函数的时候，使用JSON解析反解析就可以得到一个深拷贝对象