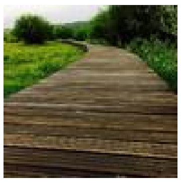
如何对比JS中两个对象是否相等



木A木 关注

12018.10.29 17:10:43字数 259阅读 35,931

① 方法一：通过`JSON.stringify(obj)`来判断两个对象转后的字符串是否相等

优点：用法简单，对于顺序相同的两个对象可以快速进行比较得到结果
缺点：这种方法有限制就是当两个对比的对象中key的顺序不是完全相同时会比较出错

② 方法二：

```
// 对Object扩展一个方法chargeObjectEqual
Object.prototype.chargeObjectEqual = function(obj){
    // 当前Object对象
    var propsCurr = Object.getOwnPropertyNames(this);
    // 要比较的另外一个Object对象
    var propsCompare = Object.getOwnPropertyNames(obj);
    if (propsCurr.length != propsCompare.length) {
        return false;
    }
    for (var i = 0,max = propsCurr.length; i < max; i++) {
        var propName = propsCurr[i];
        if (this[propName] !== obj[propName]) {
            return false;
        }
    }
    return true;
}
```

`getOwnPropertyNames`该方法可以将Object对象的第一层key获取到并返回一个由第一层key组成的数组。

优点：相对方法一进行了优化，可以应对不同顺序的Object进行比较，不用担心顺序不同而对比出错
缺点：从方法中可以看到只能获取到第一层的key组成的数组，当对象是复合对象时无法进行多层对象的比较

③ 方法三：

```

    function deepCompare(x, y) {
        var i, l, leftChain, rightChain;

        function compare2Objects(x, y) {
            var p;
        // remember that NaN === NaN returns false
        // and isNaN(undefined) returns true
        if (isNaN(x) && isNaN(y) && typeof x === 'number' && typeof y === 'number') {
            return true;
        }

        // Compare primitives and functions.
        // Check if both arguments link to the same object.
        // Especially useful on the step where we compare prototypes
        if (x === y) {
            return true;
        }

        // Works in case when functions are created in constructor.
        // Comparing dates is a common scenario. Another built-ins?
        // We can even handle functions passed across iframes
        if ((typeof x === 'function' && typeof y === 'function') ||
            (x instanceof Date && y instanceof Date) ||
            (x instanceof RegExp && y instanceof RegExp) ||
            (x instanceof String && y instanceof String) ||
            (x instanceof Number && y instanceof Number)) {
            return x.toString() === y.toString();
        }

        // At last checking prototypes as good as we can
        if (!(x instanceof Object && y instanceof Object)) {
            return false;
        }

        if (x.isPrototypeOf(y) || y.isPrototypeOf(x)) {
            return false;
        }

        if (x.constructor !== y.constructor) {
            return false;
        }

        if (x.prototype !== y.prototype) {
            return false;
        }

        // Check for infinitive linking loops
        if (leftChain.indexOf(x) > -1 || rightChain.indexOf(y) > -1) {
            return false;
        }

        // Quick checking of one object being a subset of another.
        // todo: cache the structure of arguments[0] for performance
        for (p in y) {
                if (y.hasOwnProperty(p) !== x.hasOwnProperty(p)) {
                    return false;
                } else if (typeof y[p] !== typeof x[p]) {
                    return false;
                }
            }

            for (p in x) {
                if (y.hasOwnProperty(p) !== x.hasOwnProperty(p)) {
                    return false;
                } else if (typeof y[p] !== typeof x[p]) {
                    return false;
                }

                switch (typeof(x[p])) {
                    case 'object':
                    case 'function':

                        leftChain.push(x);
                        rightChain.push(y);

                        if (!compare2Objects(x[p], y[p])) {
                            return false;
                        }

                        leftChain.pop();
                        rightChain.pop();
                        break;

                    default:
                        if (x[p] !== y[p]) {
                            return false;
                        }
                        break;
                }
            }

            return true;
        }

        if (arguments.length < 1) {
            return true; //Die silently? Don't know how to handle such case, please help...
            // throw "Need two or more arguments to compare";
        }

        for (i = 1, l = arguments.length; i < l; i++) {

            leftChain = []; //Todo: this can be cached
            rightChain = [];
        if (!compare2Objects(arguments[0], arguments[i])) {
            return false;
        }
    }

    return true;
}
```

深度对比两个对象是否完全相等，可以封装成一个组件方便随时调用。