深入理解jQuery（3）——extend

Alex Sun
2014-10-13

jQuery中，有个extend方法，常用来进行对象的扩展以及实现插件。其常见用法有：

- jQuery.extend(target [, object1] [, objectN])：扩展对象
- jQuery.extend([deep], target, object1 [, objectN])：扩展对象，并声明是浅拷贝还是深拷贝
- jQUery.extend(target)：扩展类方法（静态方法、工具方法）
- jQuery.fn.extend(object)：扩展实例方法

下面将通过源码进行具体分析。

1. 源码分析

```
jQuery.extend = jQuery.fn.extend = function() {
    var options, name, src, copy, copyIsArray, clone,
        target = arguments[0] || {},
        i = 1,
        length = arguments.length,
        deep = false;

    // Handle a deep copy situation
    // 如果第一个参数为布尔值，那么第二个参数为目标对象，之后的参数为扩展对象，i标识了起始扩展对象的位置
    if ( typeof target === "boolean" ) {
        deep = target;
        target = arguments[1] || {};
        // skip the boolean and the target
        i = 2;
    }

    // Handle case when target is a string or something (possible in deep copy)
    // 处理不合法的target
    if ( typeof target !== "object" && !jQuery.isFunction(target) ) {
        target = {};
    }
```

```
    // extend jQuery itself if only one argument is passed
    // 如果只有一个参数，说明是扩展类方法或者实例方法，target指向this，i值回退
    if ( length === i ) {
        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
                // 避免循环引用，例如 var a = { prop : a }
                if ( target === copy ) {
                    continue;
                }

                // Recurse if we're merging plain objects or arrays
                // 如果为深拷贝并且被拷贝元素为数组或者对象，则递归调用
                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
                // 浅拷贝直接改变指针，或者简单数据类型直接赋值
                } else if ( copy !== undefined ) {
                    target[ name ] = copy;
                }
            }
        }
    }

    // Return the modified object
    // 返回target，从而可以进行链式操作
```

```
    return target;
};
```

2. 深拷贝与浅拷贝

考虑下面的例子：

```
$(function() {
    var user = {
        name: "alex"
    };
    var info = {
        age: "21",
        skill: {
            java: "good",
            javascript: "good",
            c: "bad"
        }
    };

    $.extend(user, info);
    // $.extend(true, user, info);
    info.skill.java = "bad";
    console.log(user);
});
```

当没有声明为深拷贝时，user的skill只是指向了info的skill，因此最终结果的java为bad，因为它们其实是同一个对象；当声明了深拷贝时，由于此时user的skill和info的skill其实是两个对象，因此最终结果的java为good。

对于一般纯数据类型的对象，即对象的属性值不为函数，可以通过转JSON的方式来实现深拷贝，例如：

```
var user = {
    name: "alex",
    age: "21",
    skill: {
        java: "good",
        javascript: "good",
        c: "bad"
    }
};
```

```
var newUser = JSON.parse(JSON.stringify(user));

user.skill.java = "bad";

console.log(user.skill.java); // bad
console.log(newUser.skill.java); // good
```

也可以通过如下方法实现对象的深拷贝（只考虑的比较常见的情形，对于一些比较特殊的情况没有考虑，例如属性值为null等）：

```
function deepClone(obj) {
    var newObj = {};
    for (var key in obj) {
        var val = obj[key];
        newObj[key] = typeof val === "object" ? (val instanceof Array ?
val.slice() : deepClone(val)) : val;
    }
    return newObj;
}
```

3. jQuery插件机制

jQuery的extend方法很好地实现了插件机制，并且在源码中，很多方法也都是通过extend来实现的。通常的插件开发中，有时候我们会用jQuery.extend，有时候会用jQuery.fn.extend，基本规则如下：

- 所有的类方法应该使用jQuery.extend，而实例方法应该使用jQuery.fn.extend
- 插件的文件命名规范为jquer.pluginName.js，从而与其它插件区分开
- 通常插件应当返回一个对象，从而可以保证连式操作
- 可以使用jQuery.extend()来设置插件的默认参数，增强定制性
- 插件内部尽量使用jQuery而非$，从而尽量避免冲突