搜狐公司研发中心版权所有，仅供技术交流，转载请保留上述文字

搜狗实验室技术交流文档

Vol2:1 轻量级 Ajax js 库

摘要

Ajax 作为一个非常常用的功能，在页面中的作用越来越举足轻重。而对于浏览器的支持，易用性，lib 本身大小就成为了衡量一个 lib 的指标。我重写了 ajax library 用来取代现有的 prototype.js。

Ajax

Ajax既Asynchronous JavaScript And XML。现在泛指在用户浏览的页面上，在不刷新页面的情况下，和server进行交互。具体的详细的介绍请参见Ajax: A New Approach to Web Applications。使用Ajax技术，可以使用户在浏览的时候，尽量的减少刷新的次数，在一些细节方面，可以很好的提升用户体验。

为什么要有 Ajax.js

现成的 Ajax/web2.0 相关的 library 有很多，其中以 prototype 最为有名，并且也是我们现在使用的最多的一个 library。

Prototype.js是一个跟随着ror一起发展起来的web2.0 类库，他包含了OO框架，IO（Ajax，cookie），一些页面简单的特效等功能，非常的丰富，但是带来的后果就是库本身就非常的大。1.4.*约为 49k，1.5.*更是多达 69k。

很多时候，我们只是需要一些简单的 ajax 功能，但是却要背负着沉重的包袱，这是非常痛苦的一件事情。

所以我重写了 ajax 相关功能，经过几个月的使用，基本上已经稳定。大小为 2.3k，经过压缩可以减少到 1.7k。

和 prototype.js 的兼容性

除了名字上的不同，其他地方，我都尽量保持兼容 prototype 的方式。

Prototype.js 调用的方式是

```
new Ajax.Request(url, args);
```

ajax.js 调用方式是

new Ajax(url, args);

名称上面的改变是因为，对于一个小的 lib 来说，叫做 Ajax.Request 实在是太恶心了。

Api

```
//需要的参数
ajaxargs = {
    method : 'get'
    onComplete : function (req) {
    }
    ...
};

new Ajax(
    url,                //想要访问的 url
    ajaxargs            //参数
);
```

其中 ajaxargs 为

Key	Value
mime	mime type，默认为 text/html
method	get 或者 post，默认为 post
sync	true 或者 false， true 为异步，false 为同步，默认为异步请求
parameters	需要传递的参数，可以是 ka=va&kb=vb 这样的字符串，也可以是一个 key/value 的 map {ka:va,kb:vb}。 **注意：在 method 为 get 的时候，忽略此参数，应把相关参数罗列在 URL 之后。**
onComplete	回调方法，当 ajax 请求成功之后，调用此方法，并以 request 作为参数，在方法里面调用 request.responseText 则可获取请求的全部结果，默认 alert
onError	回调方法，当 ajax 请求失败调用此方法，应以 error 信息作为参数，默认什么都不做
onLoading	回调方法，当 ajax 请求过程开始的时候调用此方法，默认什么都不做

实例程序

common.js

作用：在音乐盒里面的检查是否可以添加音乐的函数，其中就是使用ajax的方法，把需要添加的专辑和音乐信息传给/mbox/addmusic.jsp这 个页面进行检查，返回一段json。

搜狐公司研发中心版权所有，仅供技术交流，转载请保留上述文字

```
function checkAddMusic(uid, plid, ms, cb) {
    new Ajax(
        {
            url : "/mbox/addmusic.jsp",    //请求addmusic.jsp进行合法性检查
            mime : "text/a",
            params : {
                plid : plid,
                uid : uid,
                m : escape(ms)
            },

            onComplete : function (req) {
                (function (result) {
                    var is_full = result['music_full'];
                    if (is_full) {
                        var MAX_SONG = 50;
                        if (result['music_size'] < MAX_SONG) {
                            alert('专辑已满，您还可以添加' +
(MAX_SONG-result['music_size']) + '首歌');
                        } else {
                            alert('专辑已满');
                        }
                    } else {
                        if (result['duplicate_count'] > 0) {
                                var str = "有"+result['duplicate_count']+"
首歌重复:";
                            var dl = result['duplicate_list'];
                            for (var i = 0; i < dl.length; i += 1) {
                                var ds = dl[i];
                                str += "₩n" + ds['title'] + "₩t" +
ds['singer'];
                            }
                            str += '₩n继续添加？'
                            if (!confirm(str)) {
                                return;
                            }
                        }

                        cb();
                    }
                })(eval("("+req.responseText+")"));  //对返回json结果进行
eval，并使用
            }
        }
```

```
    );
}
```

addmusic.jsp
作用：检查用户准备添加的歌曲的合法性。

```
<%@page contentType="text/a; charset=GBK"%>
<%@include file="actions/user_service.so"%>
<%
String plidstr = RequestUtil.getStringParam("plid", request, "-1");
long plid = -1;
try {
    plid = Long.parseLong(plidstr);
} catch (Exception e) {
    plid = -1;
}
String uid = RequestUtil.getStringParam("uid", request, "-1");  //获取参数，和普通的没有区别
String m = RequestUtil.getStringParam("m", request, "-1");
String a = RequestUtil.getStringParam("a", request, "c");

out.clear();
if ("a".equals(a)) {
    out.print(addMusicList(plid, uid, m.split("@@@")));    //输出按照json的格式
} else if ("c".equals(a)) {
    out.print(checkAddMusic(plid, uid, m.split("@@@")));
}
out.flush();
%>
```

一些应该注意的问题

1. 页面的多次点击问题
   使用 Ajax 的时候，有一些问题可能会影响我们，比如多次点击等。当我们发出了一个请求之后，是需要一定的时间才能收到服务器的结果。由于采用了异步的方法，页面也就没有阻塞，在提高用户感受的同时，也给了用户多次（暴力）点击的机会。解决办法是可以在点击调用的函数里面做一个标记，表示正在等待返回结果，此时其他的请求则会被忽略。在 onComplete 或者 onError 方法里面取消这个标记。

```
function doSth(){
   if (isRunning) return;
```

```
        isRunning = true;
        new Ajax(url, {
            …
            onComplete : function(req) {
                …
                isRunning = false;
            }
        });
    }
```

2. json的使用

json作为一个轻量级的数据传输方案已经越来越多的使用在web2.0 的应用中。具体的信息可以参考http://json.org/

在这里想说明一下json和ajax.js的配合情况。当一个jsp文件接收到请求之后，组合一段js然后返回给js，但是此时js接收到的是一个字符串，并不能直接使用，那么我们就需要一个处理，来把这个字符串转换成为js可以识别的对象。

json字符串：{“email”:”tom@sohu.com”,”nickname”:”Tom”}

对应的js对象

```
var tomObj = {
    email : “tom@sohu.com” ,
    nickname : “Tom”
}
```

在这里，我们采用一个js的小技巧，使用eval函数和js当中的匿名方法。

```
new Ajax(url,
    {
        …..,
        onComplete : function(req) { //req.responseText 为一个 json 的字符串
            (function(data) {
                alert(data.name);
                alert(data.email);
            })(eval( ‘(’ +req.responseText+’ )’ ));  // 在这里把一个 json 的字符串 convert 成为 js 的对象，并使用。具体的原理，我会稍后详细解释，大家知道可以这么用就可以了。
        }
    }
);
```