

# WebOS Log重构

@k1ic  
chenkai116@gmail.com

# 目录

- WebOS模板转换
- Node.js 和 MongoDB
- jQuery 和 Bootstrap
- Q&A



THINK BEYOND

webOS<sub>1.8</sub> on board

## 特性—日志系统



最新的日志系统，日志信息存储在mongoDB中

增加单次操作修改的文件列表信息

增加文件修改对比信息

增加按照日期查询功能

地址：<http://180.149.136.76:3000/log.html>

# WebOS模板转换

- WebOS模板转换流程

`jsjade --> phpjade --> phtml --> commit`

- jsjade文件转换为phpjade文件（正则替换）
- 调用接口将phpjade文件转换为phtml文件（远程服务）
- phtml文件提及至php Svn目录





# WebOS模板转换 --> jsjade to phpjade

```
42 //++++++全局判断区++++++
43 var reg0 = /^(s*)script\./; //判断script标签起始位置
44 var reg00 = /^(s*)(.)/; //取得第一个字符(为reg0服务)
45 var reg01 = /static/; //判断script区是否是静态的(静态无需转义)
46 var reg1 = /^(s*)\\script\./; //判断script标签结束位置
47 var reg2 = /^(s*)\s*/gim; //判断是否以"|"开头
48 var reg3 = /^(s*)\s*/im; //判断是否以"-"开头
49 var reg4 = /\b([\w\[\]\\"\\]+\b)\.length/gim; //找到js中的***.length转换成count($***)
50 var reg6 = /^doctype/gim; //当第一行是doctype时转化成!!!
51 var reg7 = /^[a-zA-Z\_\{1}\w*/; //判断是否符合js语法的变量
52 var reg8 = /\s*([\w\[\]\\"\\]+\b)\.join\(\(["']{1}\)(.*)\2\);{0,1}$/gim; //支持join方法
53
54 /*****只有是以"-"开头时才进行以下替换*****/
55 var reg30 = /^(s*)\s*for\s*/im; //判断此行是否是for语句
56 var reg33 = /(\s*if)|(\s*if else)|(\s*else)/im; //判断此行是否是if/else if/else 开头
57 var reg35 = /\s*(var)?\s*([a-zA-Z\_\{1}\w*)\s*=\s*([\w\[\]\\"\\]+\b|u4E00-u9FA5)+)/; //判断此行是否是防护数组赋值
58
59 /*****以下是不以"-"和"|"开头的情况*****/
60 var reg50 = /\s*([\w\[\]\\"\\]+\b)\s*(\w+)/gim; //替换=右边的变量
61 var reg501 = /\s*([\w\[\]\\"\\]+\b)\s*(\w+)/gim; //替换=右边的变量
62 var reg500 = /\s*([\w\[\]\\"\\]+\b)\s*(\w+)/gim; //替换=右边的加号
63 var reg51 = /([\w\[\]\\"\\]+\b)\s*([\w\[\]\\"\\]+\b)/gim; //替换#{name}、!{name}、#{obj[name]}
64 var reg52 = /\bcase\b +([\w\[\]\\"\\]+\b)\s*/gim; //替换case
65 var reg53 = /\bwhen\b +([\w\[\]\\"\\]+\b)\s*/gim; //替换 when
66 var reg54 = /\bdefault\b\s*/gim; //替换 default
67 var reg55 = /\binclude\b\s+([\w\[\]\\"\\]+\b)\s*/gim //替换include
68 var reg57 = /\s*(\s*)\bif\b\s*([\w\[\]\\"\\]+\b)\s*(\s*)\s*([\w\[\]\\"\\]+\b)\s*/gim;
```

# WebOS模板转换 --> phpjade to phtml

//调用php接口将phpjade转换为phtml,并写文件到本地tmp/phtml

**getPhtml:** function (phpJadeContent) {

if (!phpJadeContent) {...3 行} else {

var **opts** = {

host: 'tplmgr.wap.grid.sina.com.cn',

method: 'POST',

path: '/parse',

headers: {...3 行}

};

var req = http.request(opts, function(res) {

res.setEncoding('utf8');

var phpdata = '';

res.on('data', function(data) {

phpdata += data;

}).on('end', function() {

try {

//检查phpjade转换phtml文件是否成功 ( 如果原始jsjade文件内容格式有误, 会导致转换phtml失败 )

var jsonObj = JSON.parse(phpdata);

onFail('phpjade->phtml failed, errno : ' + jsonObj.code + ',error : ' + jsonObj.msg);

} catch (e) {...4 行}

tfFun.writePhtmlFile(phpdata);

});

});

# WebOS模板转换 --> phtml commit

```
function doRealSvnSubmit() {  
    var errMsg = "";  
    var phpSvnPath = config.phpSvnPath;  
    var cmdSvnInfo = 'sh svnInfo.sh ' + phpSvnPath;  
    var cmdCommit = 'sh commit.sh ' + phpSvnPath + ' ' + config.name + ' ' + config.isAll;
```

```
    //提交前获取Svn版本库的head revision  
    var ciSvnInfo = exec(cmdSvnInfo);  
    ciSvnInfo.stdout.on('data', function (data) {  
        var tmpArr = data.split("\nRevision: ");  
        var tmp = tmpArr[1];  
        if (typeof tmp == 'string' && tmp != "") {  
            oldSvnRevision = tmp.split("\n")[0];  
        }  
    });
```

```
    var ciCommit = exec(cmdCommit);  
    ciCommit.stdout.on('data', function (data) {  
        if (data.indexOf("\nCommitted revision") === 0) {  
            newSvnRevision = data.replace(/\n/g, "  
            .replace('Committed revision ', "  
            .replace('.', "  
        }  
    });
```

```
    ciCommit.on('exit', function (code) {...19 行});
```

```
}
```



# WebOS模板转换 --> commit.sh

```
1 #!/bin/bash
2
3 svn checkout --quiet --non-interactive $1 tmp/commit/$2
4
5 cp -r tmp/phtml/$2/_views/* tmp/commit/$2/
6
7 {
8     svn add --non-interactive --force tmp/commit/$2/*
9 } || {
10     echo 'svn add passed.'
11 }
12
13 svn commit --non-interactive -m "template transform." tmp/commit/$2
14
15 #清理临时工作目录
16 rm -rf tmp/phtml/$2
17 rm -rf tmp/commit/$2
18
19 exit $?
```

## WebOS模板转换 --> 记录日志

```
ciCommit.on('exit', function (code) {  
    var ret = {  
        'code': 0,  
        'msg': 'code:' + code  
    };  
  
    if (errMsg) {...5 行} else {  
        ret.code = 0;  
        ret.msg = 'svn commit compelte';  
        ret.oldSvnRevision = oldSvnRevision;  
        ret.newSvnRevision = newSvnRevision;  
    }  
  
    onComplete(ret);  
});  
}
```



# WebOS模板转换 --> 记录日志

```
onComplete: function(ret) {
    res.end();
    //记录系统操作日志
    if (USE_MONGODB == 1) {
        var logInfo = {
            submitter: data.localSvnUserName,
            project_name: proName,
            operate_month: utils.getYearMonthNow(),
            operate_time: utils.getDateTime(),
            operate_type: 'Transform',
            operate_status: 10000,
            operate_result: '成功',
            svn_path: phpSvnPath,
            old_revision: ret.oldSvnRevision,
            new_revision: ret.newSvnRevision
        };
        mongo.insertTplCmLog(logInfo);
    } else {
        var logInfo = {...7行};
        sysLog.write(logInfo);
    }
},
```

//将一条模板提交日志记录插入collection中

```
insertTplCmLog : function(dataJson) {
    var cmLogModel = this.getTplCmLogModel();
    var cmLog = new cmLogModel(dataJson);
    cmLog.save();
},
```

# WebOS 服务启动

●# sh restart.sh

```
14
15 #启动webOS服务
16
17 count=`ps -wef|grep webos.js |grep -v grep |wc -l`
18 if [ "$count" -eq 0 ];
19 then
20     nohup node cluster.js 1>log/runtime.log 2>log/error.log &
21     echo 'WebOS server started.'
22 else
23     ps -ef | grep 'node cluster.js' | grep -v grep | awk '{print $2}' | xargs kill
24     nohup node cluster.js 1>log/runtime.log 2>log/error.log &
25     echo 'WebOS server is already running. Server restarted.'
26 fi
~
```





# WebOS 服务启动-->Cluster.js

```
1 /**
2  * Created by Lihan on 2014-8-23.
3  */
4 var cluster = require('cluster');
5 var syslog = require('./syslog');
6 var cpus = require('os');
7 var path = require('path');
8 var workerNum = cpus.cpus().length;
9 var workers = {};
10
11 cluster.setupMaster({
12     exec: path.join(__dirname, 'webos.js')
13 });
14
15 workerNum = workerNum < 3 ? 3 : workerNum;
16 for(var i=0; i<workerNum; i++){
17     var worker = cluster.fork();
18     workers[worker.pid] = worker;
19 }
20 console.log('webOS master server started ' + process.pid);
21
```

# Cluster

A single instance of Node runs in a single thread. To take advantage of multi-core systems the user will sometimes want to launch a cluster of Node processes to handle the load.

The cluster module allows you to easily create a network of processes that all share server ports.



## cluster.setupMaster([settings])

- settings `Object`
  - exec `String` file path to worker file. (Default=`process.argv[1]`)
  - args `Array` string arguments passed to worker. (Default=`process.argv.slice(2)`)
  - silent `Boolean` whether or not to send output to parent's stdio. (Default=`false`)

`setupMaster` is used to change the default 'fork' behavior. Once called, the settings will be present in `cluster.settings`.



# WebOS 服务启动-->webos.js

```
10 var server = {
11   start: function () {
12     http.createServer(function(request, response) {
13       var pathname = url.parse(request.url).pathname;
14       server.router(pathname, server.handle, request, response);
15     }).listen(3000);
16   },
17   router: function (pathname, handle, request, response) {
18     if (typeof handle[pathname] === 'function') {
19       console.log (pathname + " loaded");
20       handle[pathname](request, response, pathname);
21     } else {
22       //所有非函数名的pathname均重定向到 requestHandler.index
23       handle['/'](request, response, pathname);
24     }
25   },
26   //路由表
27   handle: {...16 行}
43
44 };
45 server.start();
46 console.log('webOS server started ' + process.pid);
```



## WebOS 服务启动-->webos路由表

```
27 //路由表
28 handle: {
29     "/": requestHandler.index,
30     "/submitAll": requestHandler.submitAll,
31     "/jadeFileSubmit": requestHandler.jadeFileSubmit,
32     "/phpjade2phtml": requestHandler.phpjade2html,
33     "/log": requestHandler.log,
34     "/create": create,
35     "/transformOnly": transformHandler.transform,
36     "/getDiffTpl": tplHandler.getDiffTpl,
37     "/submitTpl": tplHandler.submitTpl,
38     "/showProjInfo": tplHandler.showProjInfo,
39     "/submitProj": tplHandler.saveProjInfo,
40     "/getCommitDetail": ajaxHandler.getCommitDetail,
41     "/getFileContent": ajaxHandler.getFileContent,
42     "/getFileDiff": ajaxHandler.getFileDiff,
43 }
44
```

# WebOS 服务启动-->webos路由表

## ●问题:

<http://webos.com:3000/assets/css/font-awesome.min.css>  
如何响应?

WebOS Log System

开始日期: 2014-9-1 结束日期: 2014-10-11 搜索

类型	用户	项目	时间	结果	PhpSvn
Transform	chenkai3	pro_component	2014-9-30 14:36:46	成功	...../ent_platform/subcode/lancer/branches/v0.0.1/applications/lancer/controllers/
Transform	chenkai3	pro_component	2014-9-30 14:34:10	成功	...../ent_platform/subcode/lancer/branches/v0.0.1/applications/lancer/controllers/

Elements Network Sources Timeline Profiles Resources Audits Console

Preserve log Disable cache

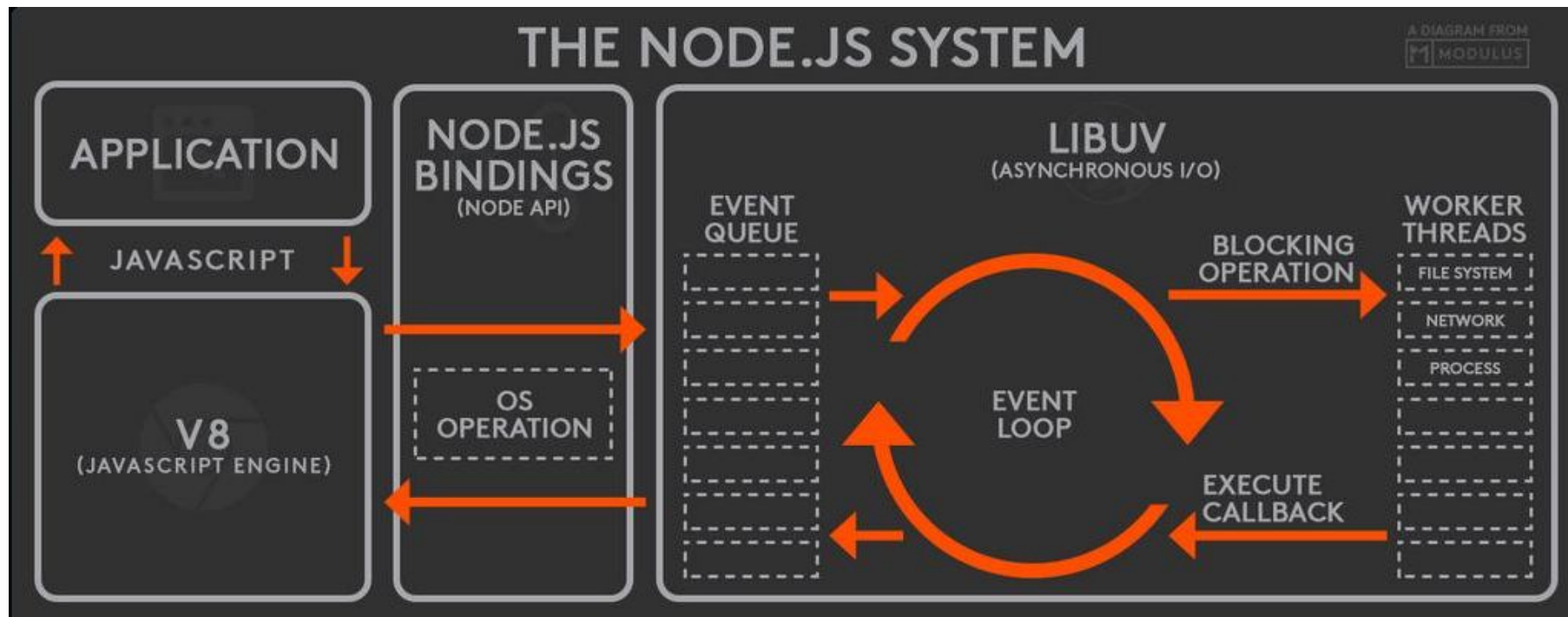
Name Path

- log.html?start=2014-9-1&end=2014-10-11
- font-awesome.min.css /assets/css
- fonts.googleapis.com.css?family=Open+Sans:400,300 /assets/css
- bootstrap.min.css /assets/css

# WebOS 服务启动-->webos路由表

```
375 index: function(request, response, pathname) {
376   var fileName;
377   if (pathname == '/') {...3 行} else {...3 行}
382   path.exists(fileName, function (exists) {
383     if (exists) {
384       var ext = path.extname(fileName);
385       ext = ext ? ext.slice(1) : 'unknown';
386       var contentType = mime[ext] || "text/plain";
387       fs.readFile(fileName, "utf-8", function (err, file) {
388         if (err) {...3 行} else {
391           response.writeHead(200, {...});
392           //如果数据适配器中提供相关数据接口，则获取数据并使用ejs进行渲染
393           var basename = path.basename(fileName, '.html');
394           basename = (basename == 'log' && USE_MONGODB == 1) ? 'logFromMongo' : basename;
395
396           if ((ext === 'html') && typeof(dataAdapter[basename]) === 'function') {...61 行}
457
458           if (fileName != 'log.html') {
459             response.write(file, "utf-8");
460             response.end();
461           }
462         }
375 }
```

## Node.js运行机制

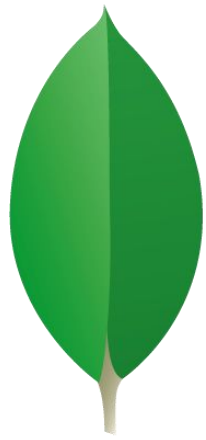




- Node.js运行机制

- 1). V8引擎解析JavaScript脚本
- 2). 解析后的代码，调用Node API
- 3). libuv库负责Node API的执行
- 4). V8引擎再将结果返回给用户





mongoDB



# MongoDB安装

1. 源码包下载, 地址: [https://fastdl.mongodb.org/linux/mongodb-linux-x86\\_64-2.2.7.tgz](https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-2.2.7.tgz)
2. 解压, `tar zxvf mongodb-linux-x86_64-2.2.7.tgz`
3. 将mongodb-linux-x86\_64-2.2.7目录 copy至/usr/local/lib目录, 并重命名为: mongodb
4. 创建相关目录及文件
  - a. 数据目录  
`cd mongodb //pwd: /usr/local/lib/mongodb`  
`mkdir -p -m 755 data/db`
  - b. 日志目录  
`mkdir -m 755 logs`
  - c. 创建日志文件  
`touch logs/mongo.log`
5. 创建配置文件 (以配置文件方式启动, 注: 另外可以以参数方式启动)  
`cd data //pwd: /usr/local/lib/mongodb/data`  
  
`vim mongo.conf`  
  
内容如下:  
`dbpath=/usr/local/lib/mongodb/data/db`  
`logpath=/usr/local/lib/mongodb/logs/mongo.log`  
`logappend=true`  
`fork=true`  
`port=27017`
6. 启动mongodb服务 (配置文件方式启动, 未加到开机启动中)  
`/usr/local/lib/mongodb/bin/mongod -f /usr/local/lib/mongodb/data/mongo.conf`

## ●问题

```
-bash: /usr/local/lib/mongodb/bin/mongod: cannot execute binary file
```

## ●解决

服务器与mongodb 的版本不匹配。 如果服务器是64位, 下载x86\_64的mongodb , 如果服务器是32位的, 下载i686的mongodb





# MongoDB安装成功

## ●启动

```
[root@vm13050022 ~]# /usr/local/lib/mongodb/bin/mongod -f /usr/local/lib/mongodb/data/mongo.conf
forked process: 23930
all output going to: /usr/local/lib/mongodb/logs/mongo.log
```

## ●查看

```
[root@vm13050022 ~]# ps -ef | grep mongod
root      523      1   0 Sep18 ?           01:51:19 /usr/local/lib/mongodb/bin/mongod
-f /usr/local/lib/mongodb/data/mongo.conf
root      24047 23333   0 21:57 pts/0    00:00:00 grep mongod
[root@vm13050022 ~]#
```



# MongoDB-->Collections

Database Explorer

Connect Refresh

10.210.213.190

- admin
- local
- webos
  - Collections
    - tpl\_commit\_infos
  - Stored JavaScript
  - GridFS
  - Users

Tree View Table View Text View

Json Text of Documents:

0 100 Refresh

[Open in Native Editor](#)

```
/* 0 */
{
  "_id" : ObjectId("54211346023ef73529b871ce"),
  "operate_status" : 10000,
  "old_revision" : 970927,
  "new_revision" : 970929,
  "svn_path" : "https://svn1.intra.sina.com.cn/weibo/weibo.com/subcode/ent_platform/subcode/lancer/branches/v0.0.1/applica",
  "operate_result" : "成功",
  "operate_type" : "Transform",
  "operate_month" : "2014-7",
  "operate_time" : ISODate("2014-07-23T06:29:26Z"),
  "project_name" : "pro_component",
  "submitter" : "chenkai3",
  "__v" : 0
}

/* 1 */
{
  "_id" : ObjectId("5421149f2a0d4f082b94e8b8"),
  "operate_status" : 10000,
  "old_revision" : 970937,
  "new_revision" : 970938,
  "svn_path" : "https://svn1.intra.sina.com.cn/weibo/weibo.com/subcode/ent_platform/subcode/lancer/branches/v0.0.1/applica",
  "operate_result" : "成功",
  "operate_type" : "Transform",
  "operate_month" : "2014-7",
  "operate_time" : ISODate("2014-07-23T06:35:11Z"),
  "project_name" : "pro_component",
  "submitter" : "chenkai3",
  "__v" : 0
}

/* 2 */
{
  "_id" : ObjectId("5421150964d8b8782b0b999fa"),
  "operate_status" : 10000,
  "old_revision" : 970944,
  "new_revision" : 970945,
  "svn_path" : "https://svn1.intra.sina.com.cn/weibo/weibo.com/subcode/ent_platform/subcode/lancer/branches/v0.0.1/applica",
  "operate_result" : "成功",
  "operate_type" : "Transform"
```

# mongoose

elegant `mongodb` object modeling for `node.js`



# Mongoose使用

```
var mongoose = require('mongoose');
var schema = mongoose.Schema; //schema maps to a MongoDB collection and defines the shape
var url = 'mongodb://' + dbName + ':' + dbPasswd + '@' + host + ':' + port + '/' + dbName;
var db = mongoose.connect(url);
var schemaObj = "";

var mongo = {
  //定义模板提交日志的schema
  getTplCmLogSchema : function() {...66 行},

  //创建模板提交日志的model
  getTplCmLogModel : function() {
    return mongoose.model(tplCmCollName, this.getTplCmLogSchema(), tplCmCollName);
  },

  //将一条模板提交日志记录插入collection中
  insertTplCmLog : function(dataJson) {
    var cmLogModel = this.getTplCmLogModel();
    var cmLog = new cmLogModel(dataJson);
    cmLog.save();
  },
};
```



# Mongoose使用-->Schema定义

## ●Schema定义

```
getTplCmLogSchema : function() {  
  if (schemaObj == "") {  
    schemaObj = new schema({  
      //提交者  
      submitter : {  
        type: String,  
        default: "",  
        required: true,  
        trim: true  
      },  
      //项目名称  
      project_name : {...6 行},  
      //操作月份 ( 如 : 2014-9 )  
      operate_month : {...6 行},  
      //操作时间  
      operate_time : {...4 行},  
      //操作类型 ( transform/create )  
      operate_type : {...6 行},  
    });  
  }  
  return schemaObj;  
},
```



# Mongoose使用-->Model创建

## ●Model创建

```
//创建模板提交日志的model  
getTplCmLogModel : function() {  
    return mongoose.model(tplCmCollName, this.getTplCmLogSchema(), tplCmCollName);  
},
```



# Mongoose使用-->Collection命名

- 向MongoDB中不存在的Collection插入数据时，数据库会自动创建Collection
- 通过Monogoose完成上述操作时发现如下问题：

`tpl_commit_info --> tpl_commit_infos`



# Mongoose使用-->Collection命名

- Mongoose源码中的mongoose/lib/util.js模块负责Collection命名 (/usr/local/lib/node\_modules/mongoose/lib/utils.js)

```
109
110 function pluralize (str) {
111   var rule, found;
112   if (!~uncountables.indexOf(str.toLowerCase())){
113     found = rules.filter(function(rule){
114       return str.match(rule[0]);
115     });
116     if (found[0]) return str.replace(found[0][0], found[0][1]);
117   }
118   return str;
119 };
120
```





# Mongoose使用-->Collection命名

```
/**
 * Pluralization rules.
 *
 * These rules are applied while processing the argument to `toCollectionName`.
 *
 * @deprecated remove in 4.x gh-1350
 */

exports.pluralization = [
  [/ (m)an$/gi, '$1en'],
  [/ (pe)rson$/gi, '$1ople'],
  [/ (child)$/gi, '$1ren'],
  [/ ^ (ox)$/gi, '$1en'],
  [/ (ax|test)is$/gi, '$1es'],
  [/ (octop|vir)us$/gi, '$1i'],
  [/ (alias|status)$/gi, '$1es'],
  [/ (bu)s$/gi, '$1ses'],
  [/ (buffal|tomat|potat)o$/gi, '$1oes'],
  [/ ([ti])um$/gi, '$1a'],
  [/ sis$/gi, 'ses'],
  [/ (?:( [^f])fel( [lr])f)$/gi, '$1$2ves'],
  [/ (hive)$/gi, '$1s'],
  [/ ([^aeiouy]|qu)y$/gi, '$1ies'],
  [/ (x|ch|ss|sh)$/gi, '$1es'],
  [/ (matr|vert|ind)ix|ex$/gi, '$1ices'],
  [/ ([m|l])ouse$/gi, '$1ice'],
  [/ (quiz)$/gi, '$1zes'],
  [/ s$/gi, 's'],
  [/ ([^a-z])$/gi, '$1'],
  [/ $/gi, 's']
];
var rules = exports.pluralization;
```

# Mongoose使用-->pluralize()函数

- 判断模型名是否是不可数的，如果是直接返回模型名；否则进行复数转化正则匹配
- 返回复数转化正则匹配结果（一个复数转化正则匹配是一个数组，有两个对象，[0]正则表达式，[1]匹配后处理结果）
- 如果复数转化正则匹配结果不存在，直接返回模型名；否则取匹配结果第一个，对模型名进行处理



# MongoDB-->访问控制

- 如何限制MongoDB只允许特定IP访问？（PS. :node. js如何限制？）
- 如何启动mongoDB客户端登录的认证机制？
- 如何限制用户访问MongoDB Web界面？



## MongoDB-->访问控制

```
dbpath = /usr/local/lib/mongodb/data/db #数据库存储目录
logpath = /usr/local/lib/mongodb/logs/mongo.log #日志输出路径
logappend = true #以追加方式记录日志
fork = true #以守护进程的方式来启动
port = 27017
bind_ip = 10.210.213.190
auth = true #启动mongodb客户端登录的认证机制
nohttpinterface = true #禁用Http访问端口
~
```





## Bootstrap

简洁、直观、强悍的前端开发框架，让web开发更迅速、简单。



# jQuery & Bootstrap --> 问题

- 如何增加页面遮罩层？
- 如何生成带按钮的模态对话框？
- 如何为DOM元素批量绑定事件，并在回调函数中操作当前元素？
- 如何在textarea中嵌入DOM树（设置textarea域背景、字体大小和颜色）？



# jQuery & Bootstrap --> 遮罩层

## ●CSS

```
<!--页面遮罩层样式-->  
<style type="text/css">  
  .bodyCover {  
    position:fixed; top: 0px; right:0px; bottom:0px;filter: alpha(opacity=60); background-color: #777;  
    z-index: 1002; left: 0px; display:none;  
    opacity:0.5; -moz-opacity:0.5;  
  }
```

## ●Html

```
<!--遮罩层-->  
<div id="bodyCover" class="bodyCover" style="display:none;"></div>
```

# jQuery & Bootstrap --> 遮罩层

## ● jQuery

```
//点击 “查看” 按钮
$(".commit_view").click(function (item) {
    $("#bodyCover").show();

    var actionData = item.target.parentNode.getAttribute('actiondata');
    var url = host + 'getCommitDetail?' + actionData;
    var tpl = '<div>\
        <p>[cm_type]</p>\
        <p><a>[svn_path]<a></p>\
        <button actionData="[curr_view_params]" type="button" class="btn-su\
        <button actionData="[diff_view_params]" type="button" class="btn-suc\
    </div><hr />';

    //获取单次提交记录详情
    $.get(url, function (data) {
        $("#bodyCover").hide();

        var res = JSON.parse(data);
        if (res.code == 0) {...30 行} else {...3 行}
        //显示提交详情弹层
        $('#cmDetailModal').modal('show');
        //点击 “当前版本内容” 按钮
        $(".curr_revision_view").click(function (item) {...25 行});
    });
});
```



# jQuery & Bootstrap --> 模态对话框

## ●Html

```
<!--提交详情展示模板-->
<div class="modal fade" id="cmDetailModal" tabindex="-1" role="dialog" aria-labelledby="cmDetailModalLabel" >
  <div class="modal-dialog" style="width: 900px;">
    <div class="modal-content">
      <div class="modal-header">
        <button type="button" class="close" data-dismiss="modal">
          <span aria-hidden="true">&times;</span><span class="sr-only">Close</span>
        </button>
        <h4 class="modal-title" id="cmDetailModalLabel">提交详情</h4>
      </div>
      <div class="modal-body" id="cm_detail">
        提交详情
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-primary" data-dismiss="modal">
          &nbsp;&nbsp;&nbsp;关闭&nbsp;&nbsp;&nbsp;
        </button>
      </div>
    </div>
  </div>
</div>
```

# jQuery & Bootstrap --> 模态对话框

## ● jQuery

```
//点击 “查看” 按钮
$(".commit_view").click(function (item) {
    var url = host + 'getCommitDetail?' + item.target.parentNode.getAttribute('actiondata');
    var tpl = '<div>\
        <p>[cm_type]</p>\
        <p><a>[svn_path]<a></p>\
        <button actionData="[curr_view_params]" type="button" class="btn-success curr_revi\
        <button actionData="[diff_view_params]" type="button" class="btn-success diff_view'\
    </div> <hr />';

    //获取单次提交记录详情
    $.get(url, function (data) {
        var res = JSON.parse(data);
        var list = res.data.list.split('|');
        var pre = res.data.revisions.pre;
        var post = res.data.revisions.post;

        for (var i in list) {...21 行}

        $("#cm_detail").html(html);
        //显示提交详情弹层
        $('#cmDetailModal').modal('show');
    });
});
```

# jQuery & Bootstrap --> Div中嵌入DOM树

## ●Html

```
<!--当前版本内容div-->  
<div id="currRevisionContent" title="当前版本内容" style="display: none;">  
  <textarea id="currRevCont" rows="60" cols="180" style="background-color:#272727;font-size:16pt;line-height:20pt;col  
</div>
```

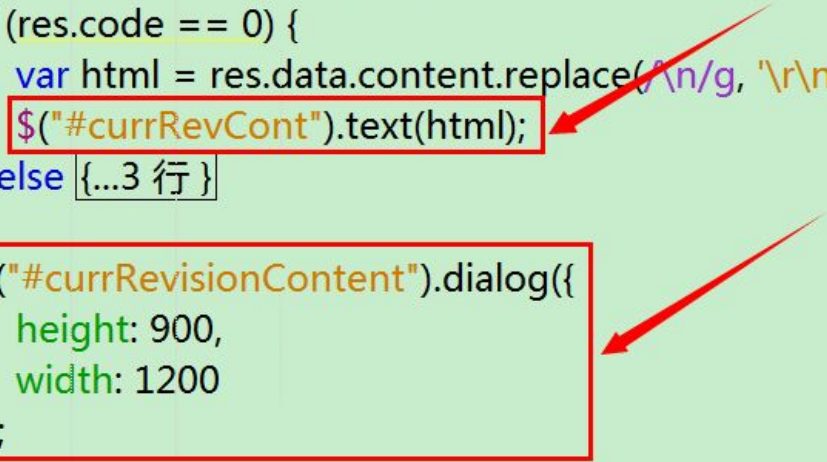


# jQuery & Bootstrap --> Div中嵌入DOM树

## ●jQuery

```
//点击“当前版本内容”按钮
$(".curr_revision_view").click(function (item) {
    var url = host + 'getFileContent?' + item.target.getAttribute('actionData');

    //获取当前版本内容
    $.get(url, function (data) {
        var res = JSON.parse(data);
        if (res.code == 0) {
            var html = res.data.content.replace(/\n/g, '\r\n');
            $("#currRevCont").text(html);
        } else { ...3 行 }
    });
});
```



```
$("#currRevisionContent").dialog({
    height: 900,
    width: 1200
});
```

## jQuery & Bootstrap --> Div中嵌入DOM树

- `$(selector).html(val)` vs `$(selector).text(val)`

后者会将“<”和“>”替换成相应的HTML实体





# jQuery & Bootstrap --> Div中嵌入DOM树

当前版本内容

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>
      微博打赏服务协议 微博
    </title>
  </head>
  <body class="B_register">
    <div class="W_nologin">
      <div class="W_reg_header">
        <div class="W_nologin_logo">
          <a href="#"></a>
        </div>
      </div>
      <div class="W_nologin_main">
        <div class="private_cont">
          <h2 class="head_title W_f16 W_fb W_tc">
            《 打赏服务协议 》
          </h2>
          <p class="para">
            感谢您选择使用
          </p>
          <p class="para">
            本协议由您与微
          </p>
          <h3 class="para_title W_f14 W_fb">
            一、定义：
          </h3>
          <p class="para">
            如无特别说明，下列术语在本协议中的定义为：
          <br />
            1.1 微博：基于用户关系的信息分享、传播以及获取平台。用户可以通过PC、手机等多种移动终端接入，以不超过
```

# jQuery & Bootstrap

- 连续操作

```
var tmp = tpl.replace('[cm_type]', type)
               .replace('[svn_path]', path)
               .replace('[curr_view_params]', curr_v_params)
               .replace('[diff_view_params]', diff_params);
```

- json格式参数

```
$("#currRevisionContent").dialog({
  height: 900,
  width: 1200
});
```



# jQuery & Bootstrap

- 回调导致嵌套过深

```
//点击 “查看” 按钮
$(".commit_view").click(function (item) {
    var url = host + 'getCommitDetail?' + item.target.parentNode.getAttribute('actiondata');
    //获取单次提交记录详情
    $.get(url, function (data) {
        var res = JSON.parse(data);
        if (res.code == 0) {...30 行} else {...3 行}

        //点击 “当前版本内容” 按钮
        $(".curr_revision_view").click(function (item) {
            var url = host + uri + 'getFileContent?' + item.target.getAttribute('actionData');
            //获取当前版本内容
            $.get(url, function (data) {
                var res = JSON.parse(data);
                if (res.code == 0) {...4 行} else {...3 行}
                $("#currRevisionContent").dialog({...4 行});
            });
        });

        //点击 “与上一版本Diff” 按钮
        $(".diff_view").click(function (item) {
            var url = host + uri + 'getFileDiff?' + item.target.getAttribute('actionData');
            //获取Diff详情
            $.get(url, function (data) {
                var res = JSON.parse(data);
                if (res.code == 0) {...4 行} else {...3 行}
                $("#revisionDiffContent").dialog({...4 行});
            });
        });
    });
});
```

# jQuery & Bootstrap

- 回调导致嵌套过深

```
var events = require( "events" );  
var event = new events.EventEmitter();
```

//按时间查询

```
logListSearch : function(start, end) {  
  var startDate = start != "" ? start : "";  
  var endDate = end != "" ? end : "";  
  
  if (startDate == "" || endDate == "") {...4 行}  
  
  $condition = {...};  
  
  mongo.getTplCmLogModel()  
    .find($condition)  
    .sort({...})  
    .limit(100)  
    .exec(function(err, docs) {  
      event.emit('getSearchRes', docs);  
    });  
},
```

```
dataAdapter.logListSearch(start, end);  
event.on('getSearchRes', function(list) {  
  var listArr = [];  
  
  //data for display  
  var initData = {...4 行};  
  
  for(var key in list){...19 行}  
  
  data.curLog = listArr;  
  data.curDate = curMonth;  
  data.initData = initData;  
  
  file = ejs.render(file, {...});  
  response.write(file, "utf-8");  
  response.end();  
});
```





# 感谢

- 感谢

@李晗、@吴侃、@谢飞、@吴迪、@高鑫、@雨琦、@连生 还有可爱的大家



以微博之力, 让世界更美!

weibo.com