

# Nodejs web 开发简介

# 大纲

- nodejs 简介 ( 安装 ,v8, ECMA 标准兼容性 ,npm )
- nodejs web 开发框架 express 简介 (restful api, query, params, 文件上传下载, 主要结合代码实例进行介绍)
- nodejs 开发调试简介 ( console.log, util.inspect, JSON.stringify, node-dev...)
- js 元编程简介 ( ECMA proxy 规范, \_\_noSuchMethod\_\_ )
- Nodejs 异步编程
- Express web 开发框架
- ria-packager 开发中遇到的部分问题及解决方案

# About nodejs

<http://nodejs.org/>

- <https://github.com/joyent/node/wiki>

Node.js is a platform built on ***Chrome's JavaScript runtime*** for easily building fast, scalable network applications. Node.js uses an ***event-driven, non-blocking I/O model*** that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

# How to install node

- Windows mac 用户直接下载 2 进制安装包即可 .
- 最新的 node 也有 linux (64 位 ) 二进制包 ( 预编译好的 )
- Read the document!
- <https://github.com/joyent/node/wiki/Installation>

# Http server

- A tiny http server

```
var http = require('http');
http.createServer(function(req, res) {
  console.log(req.headers);
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  res.end('Hello World\n');
}).listen(8080, '127.0.0.1');
console.log('Server running at http://127.0.0.1:8080/');
```

# ECMA 5(6) Mozilla Features Implemented in V8

- 没有浏览器端开发要考虑那么多兼容性问题
- <https://github.com/joyent/node/wiki/ECMA-5-Mozilla-Features-Implemented-in-V8>
- <http://dailyjs.com/2012/10/15/preparing-for-esnext/>

# ecma5

- `Array.forEach`
- `Object.keys`
- `Function.bind`

```
[ "a", "b", "c" ].forEach(function(item, index){
    console.log(item, index);
});

var obj = {
    'name' : 'dyw',
    'test' : function(){
        console.log(this.name);
    }
};
console.log(Object.keys(obj));

setTimeout(obj.test, 10);
setTimeout(obj.test.bind(obj), 10);
```

# JSON

- JSON.stringify
- JSON.parse

```
var Tree = {  
  "name" : 'root',  
  "children" : [  
    {  
      "name" : "node-level-1"  
    },  
    {  
      "name" : "sub-node-1"  
    }  
  ]  
};  
console.log('stringified : ',JSON.stringify(Tree));  
console.log('stringified : ',JSON.stringify(Tree,null,3));  
console.log('parsed : ',JSON.parse(JSON.stringify(Tree)));
```



# \_\_defineGetter\_\_

- 

- 

```
var obj = {  
  get a() {  
    return "something"  
  },  
  set a() {  
    console.log(this.a)  
  }  
};  
obj.__defineGetter__('foo', function(){  
  return 'bar'  
});  
  
obj.a = 123;  
console.log(obj.a, obj.foo);
```

# node --v8-options

```
#node --v8-options > v8-options.js
```

Usage:

```
shell [options] -e string
  execute string in V8
shell [options] file1 file2 ... filek
  run JavaScript scripts in file1, file2, ..., filek
shell [options]
shell [options] --shell [file1 file2 ... filek]
  run an interactive JavaScript shell
d8 [options] file1 file2 ... filek
d8 [options]
d8 [options] --shell [file1 file2 ... filek]
  run the new debugging shell
```

Options:

```
--use_strict (enforce strict mode)
  type: bool default: false
--es5_readonly (activate correct semantics for inheriting readonliness)
  type: bool default: false
--es52_globals (activate new semantics for global var declarations)
  type: bool default: false
--harmony_typeof (enable harmony semantics for typeof)
  type: bool default: false
--harmony_scoping (enable harmony block scoping)
  type: bool default: false
--harmony_modules (enable harmony modules (implies block scoping))
  type: bool default: false
--harmony_proxies (enable harmony proxies)
  type: bool default: false
--harmony_collections (enable harmony collections (sets, maps, and weak maps))
  type: bool default: false
--harmony (enable all harmony features (except typeof))
  type: bool default: false
```

# node --harmony\_proxies( 动态元编程 )

- node --harmony or node --harmony\_proxies
- [http://soft.vub.ac.be/~tvcutsem/invokedynamic/proxies\\_tutorial](http://soft.vub.ac.be/~tvcutsem/invokedynamic/proxies_tutorial)

```
//v8(nodejs),firefox 支持Proxy  
//node --harmony harmony-proxy.js |  
var model = Proxy.create({  
  get: function(proxy, name) {  
    return 'Hello, ' + name;  
  }  
});  
  
console.log(model.dyw); // should print 'Hello, dyw'
```

# Spider monkey \_\_noSuchMethod\_\_

- \_\_noSuchMethod\_\_
- 仅 Firefox 浏览器支持

```
var obj = {
  __noSuchMethod__: function(methodName, args) {
    console.info(methodName, args);
    obj[methodName] = function(args){//define method dynamically!
      console.warn(methodName + ' invoked with args:',args);
    };
  }
};
obj.testFunction(1, 'abc', 234); //run it in firebug!
obj.testFunction({
  "test" : 123
}); //run it in firebug!
```

# node-dev

- `node test.js`
- `node-dev test.js` // 监控 js 内容变化，即时重启 node

# npm

- npm publish(npm adduser)
- npm install express -g
- npm update express -g
- **package.json**

# 异步 vs 同步

- `fs.readFileSync('net.js', 'utf-8');`
- `fs.readFile('net.js', 'utf-8', function(err, data) {});`
- nodejs 编程推荐都用异步方式，以保证无阻塞。
- 

```
var fs = require('fs');

fs.readFile('net.js', 'utf-8', function(err, data) {
  if (err) throw err;
  console.log('async: ', data);
});

console.log('Sync: ', fs.readFileSync('net.js', 'utf-8'));
```

# 异步回调的嵌套陷阱

- promises

```
Parse.User.logIn("user", "pass", {  
  success: function(user) {  
    query.find({  
      success: function(results) {  
        results[0].save({ key: value }, {  
          success: function(result) {  
            // the object was saved.  
          }  
        });  
      }  
    });  
  }  
});
```

```
Parse.User.logIn("user", "pass").then(function(user) {  
  return query.find();  
}).then(function(results) {  
  return results[0].save({ key: value });  
}).then(function(result) {  
  // the object was saved.  
});
```



# 使用 promise 解决回调嵌套

- promise

```
Parse.User.logIn("user", "pass", {
  success: function(user) {
    query.find({
      success: function(results) {
        results[0].save({ key: value }, {
          success: function(result) {
            // the object was saved.
          },
          error: function(result, error) {
            // An error occurred.
          }
        });
      },
      error: function(error) {
        // An error occurred.
      }
    });
  },
  error: function(user, error) {
    // An error occurred.
  }
});
```

```
Parse.User.logIn("user", "pass").then(function(user) {
  return query.find();
}).then(function(results) {
  return results[0].save({ key: value });
}).then(function(result) {
  // the object was saved.
}, function(error) {
  // there was some error.
});
```

# jQuery Promise

- <http://api.jquery.com/jquery.ajax/>
- The jqXHR objects returned by `$.ajax()` as of jQuery 1.5 implement the Promise interface, giving them all the properties, methods, and behavior of a Promise (see **Deferred** object for more information).

```
var jqxhr = $.ajax("example.php").done(function() {  
    alert("success");  
}).fail(function() {  
    alert("error");  
}).always(function() {  
    alert("complete");  
});|
```

# Express web framework

- **Sinatra inspired** web development framework for node.js -- insanely fast, flexible, and simple
- Built on ***Connect*** Middleware
- <http://expressjs.com>
- `npm install -g express`
- Read the doc!!

```
//npm install -g express
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.send('Hello World');
});

app.listen(3000);
```

# What Express Does

- Parses arguments and headers
- Routing
- Views
- Partials
- Layouts
- Configuration
- Sessions

# Request Object

- req.param
- req.query
- req.body
- Req.files
- req.flash ---- show error msg
- req.session , req.cookies express.cookieParser;
- req.headers

# Response Object

- `res.redirect`
- `res.write` ---Transfer-Encoding:**chunked**
- `res.end`
- `res.download` `res.sendfile` `res.attachment`
- `res.send`
- `res.status`
- `res.json` `res.jsonp`
- `res.format` `res.render`

# express--CRUD

- CRUD <---> HTTP Get,Post,Put,Delete
- restful

```
//npm install -g express
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.send('get!');
});
app.post('/', function(req, res){
  res.send('post!');
});
app.put('/', function(req, res){
  res.send('put!');
});
app.delete('/', function(req, res){
  res.send('delete!');
});
app.listen(3000);
```

# Get-Post-Put-Delete

The screenshot displays the Postman application interface. On the left sidebar, the 'History' tab is active, showing a list of recent requests:

- POST** http://127.0.0.1:3000/upload
- DELETE** http://127.0.0.1:3000/
- PUT** http://127.0.0.1:3000/
- POST** http://127.0.0.1:3000/
- GET** http://127.0.0.1:3000/

The main panel on the right shows the details of the selected POST request:

- Environment:** Normal (Basic Auth, Digest Auth, OAuth 1.0, No environment)
- URL:** http://127.0.0.1:3000/upload
- Body Type:** form-data (x-www-form-urlencoded, raw)
- Form Data:** A single entry with 'Key' as 'test' and 'Value' as 'uploaded'. A 'Choose Files' button and 'No file chosen' text are visible next to the value field.
- Buttons:** Send, Preview, Add to collection
- Response:** STATUS 200 OK, TIME 44 ms
- Body View:** Pretty (Raw, Preview, JSON, XML)
- Response Content:** 1 uploaded



# express.methodOverride

- `<input type="hidden" name="_method" value="delete"/>` 附加参数 `_method`.
- `app.use(express.bodyParser());`
- `app.use(express.methodOverride());`
- `jQuery.ajax()` 通过 `type` 参数支持 http PUT 及 DELETE 方法 ( 部分浏览器 XMLHttpRequest 对象支持 put,delete 方法 )

# express—file upload

- `app.use(express.bodyParser)`
- <https://github.com/felixge/node-formidable>

```
var express = require('express'), http = require('http');

var app = express();

app.set('port', process.env.PORT || 3000);
app.use(express.bodyParser({
  keepExtensions: true,
  uploadDir: '/tmp/express/files' //mkdir -p /tmp/express/files
}));

app.post('/upload', function(req, res){
  console.log(req.files);
  res.send('uploaded');
});

http.createServer(app).listen(app.get('port'), function() {
  console.log('Express server listening on port ' + app.get('port'));
});
```

# ria-packager 开发中遇到的部分问题及解决方案

- `require('a/b/c/_test/main.json');`
- less 中背景图片 url 处理 ( 相对路径调整 , 加 md5 版本号 )
- Mustache 模板渲染
- 进程管理

# 按需加载模板数据

- require('a/b/c/\_test/main.json');
- 使用 eval 解析 json
- 重载 require.

```
//可以加载子json文件-----  
//{ "test" : require("widget/demo/_test/main.json"), "aaa" : 123 }  
var _require_ = module.require;  
module.require = function(id){  
    if(id.match(/\.json$/)){  
        id = path.join(root,id);  
        return eval('(' + fs.readFileSync(id, 'utf-8') + ')')  
    }else{  
        return _require_(id);  
    }  
};  
//-----
```

# less 中背景图片 url 处理

- path.relative
- 打包时先 copy 所有图片，并计算其 md5 hash.

```
config['_root_less_'] = file;

var parser = new(less.Parser)({
  // Specify search paths for @import directives (相对路径的起始目录)
  paths      : [path.dirname(file)],
  // Specify a filename, for better error messages
  filename   : file,
  syncImport : true,
  files      : {
    '_config_' : config
  }
});
```

```
var config = this.env.files._config_, host;
var img = absolute(this.env.filename, val.value).replace(/\\/g, '/');
val.value = relative(config['_root_less_'], this.env.filename, val.value);
if(md5Mapping[img]){
  val.value = val.value + '?v=' + md5Mapping[img];
}
```

# mustache 模板渲染

- 清除模板缓存 `mustache.clearCache();`
- 加载子模板：

```
output = Mustache.render(content, data, function(partial) { //auto load partial template
  return fs.readFileSync(path.join(root, partial), 'utf8').trim();
});
```

# 进程管理

- process.pid

```
process.on('uncaughtException', function(err) {
    console.error('Caught exception: ', err);
});

var pidPath = path.join(os.tmpDir(), '.node_pid');
fs.writeFile(pidPath, process.pid);

process.on('SIGTERM', function() { //SIGKILL是kill -9 的信号,无法捕获; SIGTERM是kill的信号,可以捕获
    console.log('HTTPD killed');

    fs.unlink(pidPath, function() {
        process.exit(0);
    });
});

process.title = 'ria-server'; //linux only
```

```
function kill(){
    if(fs.existsSync(pid)){
        try{
            process.kill(fs.readFileSync(pid, 'utf-8'));
        }catch(e){
        }
        fs.unlinkSync(pid);
    }
}
```

# spawn 进程

- Spawn or fork(clone) or exe

```
var ps = spawn('node', [path.resolve(__dirname, '..', 'lib', 'server', 'httpd.js'),
                        '-port', conf['-port'],
                        '-root', conf['-root']]);

ps.stdout.on('data', function (data) {
  console.log(data.toString('utf-8'));
});

ps.stderr.on('data', function (data) {
  console.log(data.toString('utf-8'));
});
```



# 信号处理

- SIGTERM , SIGKILL,... 软重启 .
- process.exit(0|1); 0 成功 ,1 出错 .

```
process.on('uncaughtException', function(err) {  
    console.error('Caught exception: ', err);  
});  
  
var pidPath = path.join(os.tmpDir(), '.node_pid');  
fs.writeFile(pidPath, process.pid);  
  
process.on('SIGTERM', function() { //SIGKILL是kill -9 的信号,无法捕获; SIGTERM是kill的信号,可以捕获  
    console.log('HTTPD killed');  
  
    fs.unlink(pidPath, function() {  
        process.exit(0);  
    });  
});  
  
process.title = 'ria-server'; //linux only
```

# The end

- Q & A?