drupal & nodejs

beyond chat.

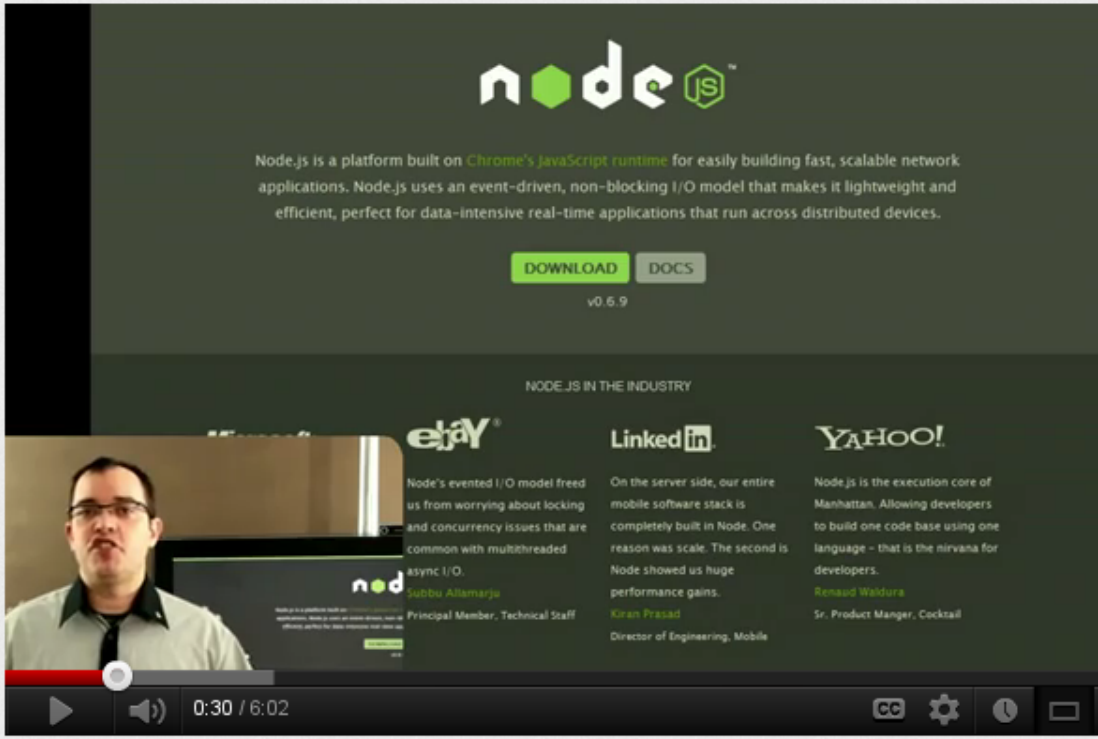
@frega - flink solutions, berlin

Node.js Is Bad Ass Rock Star Tech

gar1t | Abonnieren | 10 Videos

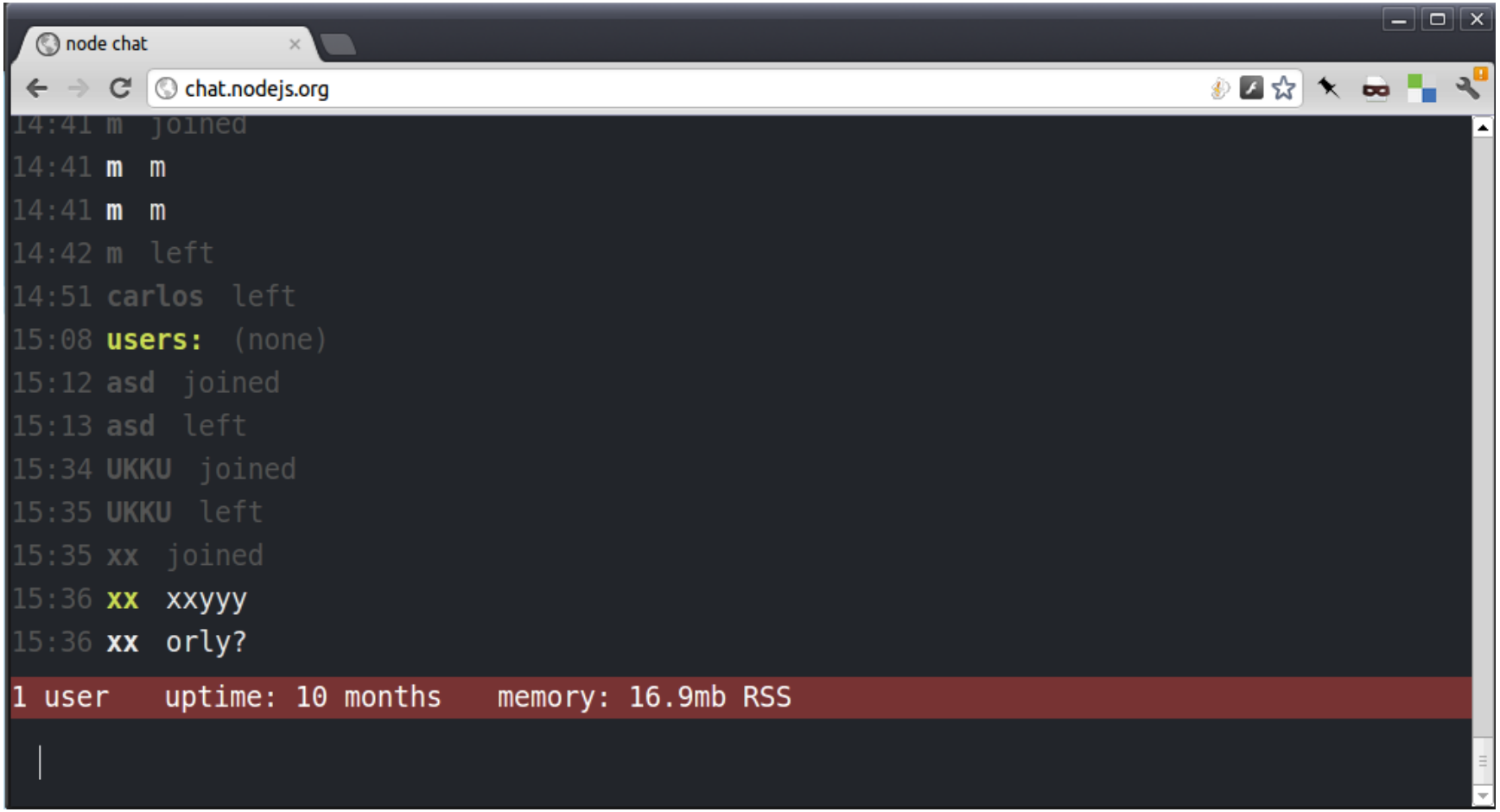
Node.JS Is Stupid And If You Use It So Are You

BlackwaterOpsDotCom | Abonnieren | 448 Videos



http://www.youtube.com/watch?v=bzkRVzciAZg http://www.youtube.com/watch?v=1e1zzna-dNw

why beyond chat?



"nodejs, javascript & the future"

jeff miccolis' pain points w/ php (+ drupal)

http://denver2012.drupal.org/program/sessions/nodejs-javascript-and-future

- handling/pulling external data-sources efficiently, (feeds, "idling", network latency)
- big uploads / big (dynamic) downloads
- sending / receiving lots of emails (pubsub)

if you have those kind of problems, **php is perhaps not the right tool**.

nodejs might be better.

... non-pivoting drupal mortals ...

how to **leverage both**?
our **drupal toolchain** as well as **nodejs**?

drupal_add_serverside_js?

no finished solutions.
a few patterns.

a) nodejs.module

quite tightly scoped, **RC1, most widely used**

- focusses on message passing (drupal<->nodejs<->browser)
- socketio / client-side websockets
- **some extensions possible but e.g. no registry**

http://drupal.org/project/nodejs

b) node-drupal

tightly scoped, rather tightly coupled.

- "consume" drupal from nodejs not vice-versa.
- direct mysql operations

https://github.com/mikl/node-drupal

c) dnode.module

leverages dnode-rpc-protocol; lightweight JSON-based RPC/RMI in nodejs.

- PHP <-> nodejs (api module)
- Server registry, configuration, DX helpers

http://drupal.org/sandbox/frega/1321342

addressing php pain points

- big files - download
  - nodejs strength: i/o streams
  - dnode_fpd - fast (private) downloads
- third-party data
  - nodejs strength: non-blocking "event"-handling
  - integrating external sources w/ entity, views + rules.
- drupal + nodejs integration: architecture

example 1: fast (private) file downloads "big images with watermarks"

"offloading" big private file downloads

1. big files in **drupal** (managed private files)
2. **drupal** validates/authorises request
3. **drupal** generates token (& "firms" it)
4. pass token + data (file path etc.) to **nodejs**
5. **drupal** redirects to **nodejs** server w/ token
6. **nodejs** validates token
7. **nodejs** (manipulates &) pumps file

(approach is a little different from beejeebus' fpd.module)

let's first see how nodejs "pumps"

nodejs - a small file server server

nodejs - streams (basic.js)

simple webserver (streaming IO)

```
1  var fs = require('fs'), http = require('http');
2  http.createServer(function(req, res) {
3    if (req.url == '/favicon.ico') { return res.end(); }
4    // this is a big image ...
5    var filePath = __dirname + '/test.png';
6    var readStream = fs.createReadStream(filePath);
7    readStream.pipe(res);
8  }).listen(2000);
9  console.log('Started a test.png-server listening on :2000');
```

streams w/ a little extra (convert.js)

```
1  var fs = require('fs'),
2    http = require('http'),
3    spawn = require('child_process').spawn;
4  var reqCount = 0;
5  http.createServer(function(req, res) {
6    if (req.url == '/favicon.ico') { return res.end(); }
7    console.log('Incoming request %s', req.url);
8    reqCount++;
9    var filePath = __dirname + '/test.png';
10   var now = reqCount + ':' + new Date().toString();
11   var args = [filePath,
12     '-resize', '200x200',
13     '-pointsize', 10, '-fill', 'white',
14     '-draw', 'text 0,100 "' + now + '"', '-'];
15   var convert = spawn('convert', args);
16   convert.stdout.pipe(res);
17   convert.stderr.pipe(process.stderr);
18 }).listen(2000);
19 console.log('Started a "watermarking" imagemagick server on :2000');
```

<demo>

node basic.js

node convert.js

node.js - characteristics

- light-weight, single-threaded server
- spawn cpu-intensive/blocking sub-processes
- streaming, non-blocking IO ("pipe" readable stream into writeable streams)

- many small servers, re-composability and re-usability

http://substack.net/posts/b96642/the-node-js-aesthetic

back to integration

1. big files in **drupal** (managed private files)
2. **drupal** validates/authorises request
3. **drupal** generates token (& "firms" it)
4. pass token + data (file path etc.) to **nodejs**
5. **drupal** redirects to **nodejs** server
6. **nodejs** validates token
7. **nodejs** (manipulates &) pumps file

(approach is a little different from beejeebus' fpd.module)

dnode_fpd.module - RPC details

drupal - "client" on the php side:

```
114   /** Page callback for the private file redirect. ...*/
120   function dnode_fpd_redirect($file) {
121     if ($info = dnode_fpd_get_redirect_info($file)) {
122       dnode_rpc('dnode_fpd', 'setDownloadFileToken', array($info['token'], $info['data']), function($err) use ($info) {
123         if ($err) {
124           drupal_set_message(t('Could not set download file token.'), 'error');
125           return FALSE;
126         }
127         return drupal_goto($info['url']);
128       });
129     }
130     else {
131       drupal_access_denied();
132     }
133   }
```

declare "dnode_fpd" - server in a registry

```
139   /**
140    * Implements hook_dnode_info().
141    */
142   function dnode_fpd_dnode_info() {
143     return array(
144       'dnode_fpd' => array(
145         'type' => 'ssjs',
146         'port' => 10010,
147         'category' => 'server',
148         'config' => array(...),
156       ),
157     );
158   }
```

ssjs/dnode_fpd.dnode.js

```
1   var spawn = require('child_process').spawn;
2   // DrupalDnode - provide by dnode.dnode.js
3   DrupalDnode = global.DrupalDnode;
4   var config = DrupalDnode.getModuleConfig('dnode_fpd');
5   var debug = config.debug || false;
6   // Expressjs server instance.
7   var app = require('express').createServer();
8   app.configure(function(){...});
11  app.get('/download/:token', function (req, res) {...});
74  app.listen(config.http_port, config.http_host);
75
76  // Primitive token store.
77  var tokens = {};
78  // Retrieve and validate token.
79  function validToken(token, ip) {...}
98  // Simple GC for tokens.
99  setInterval(function() {...}, 1000);
105 // Setup the the dnode server.
106 exports.setDownloadFileToken = function(token, data, cb) {
107   debug && console.log("setDownloadFileToken", token, data);
108   tokens[token] = data;
109   cb(null, true);
110 }
```

dnode_fpd.dnode.js - close-up

```
41  app.get('/download/:token', function (req, res) {
42    var token = req.params.token || '';
43    var tokenData = validToken(token);
44    if (tokenData) {
45      if (tokenData.user_agent) {...}
51      if (tokenData.ip_address) {...}
60      debug && console.log('Token - ', token, ' data', tokenData);
61      if (tokenData.watermark) {
62        var now = new Date().toString();
63        var args = [tokenData.realpath, '-pointsize', 100, '-fill', 'white',
64          '-draw', 'text 0,100 "' + now + '"', '-'];
65        var convert = spawn('convert', args);
66        res.attachment(tokenData.filename);
67        convert.stdout.pipe(res)
68        convert.stderr.pipe(process.stderr);
69        return ;
70      }
71      // Pass on headers for original otherwise.
72      for (var header in tokenData.headers) {...}
75      // Check for tries ...
76      if (tokenData.tries == 1) {...} else {...}
88      // this is just a expressjs wrapper around a fs.createReadStream().pipe().
89      res.download(tokenData.realpath, tokenData.filename);
90    } else {
91      debug && console.log('No tokendata could be retrieved for token ', token);
92      res.send(404);
93    }
94  });
```

dnode_fpd - dnode DX and config.

drush dnode-nodejs-server

node some-path/ssjs/dnode-server.js --modules='{"dnode_fpd":"
sites\/default\/modules\/dnode_fpd\/ssjs\/dnode_fpd.dnode.js"}' --
config='{"dnode_fpd":{"debug":true,"port":10010,"https":false,"http_host":false,"
http_port":10011}}'

Configuration via Drupal ...

* drush vset dnode_ftp_base_url "http://somepublichost*

<demo>

http://localhost/demo/d7_dnode_fpd/

drush vset dnode_fpd_watermark_images

dnode_fpd - why this hazzle?

downsides: more moving parts, more ports, more "daemons" (xsend? nginx? ...)

for production maybe haproxy/nginx/varnish config additionally

dnode_fpd - why this hazzle?

- **it's hackable and efficient**
  - watermark a "remote" file, e.g. pass a "protected" S3-URI, watermark it and then pipe it to client
  - other types of watermarks (mp3?, videos?)

- **it's expandable**
  - throttle downloads (i.e. pause + resume streams).
  - progressive downloads (start from stream "offset")
  - callbacks on "end" of stream etc.

big uploads also "solved" via nodejs ... https://transloadit.com/

dnode_fpd - why dnode?

Matter of taste and architecture

- **dnode - instead of a signed http request?**
  - ease of use, blocking/non-blocking, sometime responses matter, think beyond message-passing,
- "**hook_dnode_info**" **-** gives us a registry (abstraction/encapsulation), basic configuration sharing (ports?) and better DX.

example 2: subpub

external datasources, more interoperation

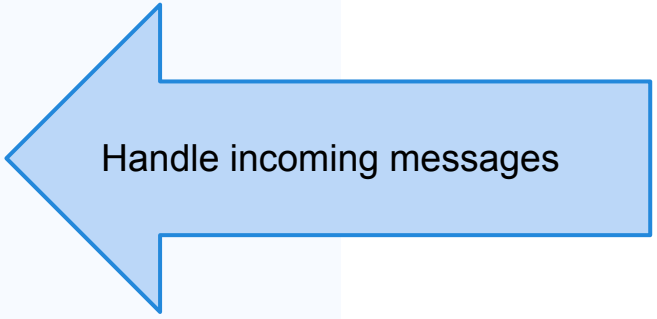
subpub: external datasources

IRC because it's easier to demonstrate than RSS aggregation. same principle.

subpub: subpub.module

```php
<?php
/** Implements hook_dnode_info(). ...*/
function subpub_dnode_info() {
  return array(
    'subpub' => array(
      'type' => 'ssjs',
      'port' => 11000,
      'category' => 'server',
      'config' => array(
        'debug' => TRUE,
        'bot_name' => variable_get("subpub_bot_name", "Henry"),
        'irc_server' => variable_get("subpub_irc_server", "flinker"),
        'irc_channels' => variable_get("subpub_irc_channels", array("#test")),
      ),
    ),
  );
}

/** Implements hook_dnode_message(). ...*/
function subpub_dnode_message($event, $message) {
  if ($event=='subpub') {
    $m = message_create('subpub_message', array(
      'arguments' => array(
        '@from' => $message['from'],
        '@channel' => $message['channel'],
        '@message' => $message['message'],
      ),
    ));
    if (is_object($m)) {
      $m->save();
    }
  }
}
```



subpub: ssjs/subpub.dnode.js

```
1  DrupalDnode = global.DrupalDnode;
2  var config = DrupalDnode.getModuleConfig('subpub');
3  var irc = require('irc'), c = require('irc-colors');
4  var bot = new irc.Client(
5    config.irc_server, config.bot_name, {
6    debug: config.debug,
7    channels: config.irc_channels
8  });
9
10 bot.addListener('message', function (from, channel, message, more) {
11   DrupalDnode.connectByServerId('dnodeq', function(remote, conn) {
12     remote.sendMessageToDrupal('subpub', {
13       from: from,
14       channel: channel,
15       message: message,
16       data: more,
17       timestamp: +( new Date() )
18     }, {}, function(err, data) {
19       console.log('Sent to drupal - %s => %s: %s', from, channel, message);
20       conn.end();
21     });
22   });
23 });
24
25 exports.say = function(channel, message, cb) {
26   console.log('SAY %s in %s', channel);
27   bot.say(channel, message);
28   cb(null);
29 }
```

subpub: nodejs -> dnode -> drupal

- on 'message'
  - use dnode to call dnodeq that creates a (signed) http requests to drupal.

  - **DRUPAL** module_invoke_all('dnode_message', 'pubsub', $message)

- hook_dnode_message($event, $message)
  - create message entity

subpub:
IRC -> nodejs -> dnode/http -> drupal

<demo>

admin/content/messages

subpub: rules integration

```php
<?php
/** @file ...*/

/** Implements hook_rules_event_info(). ...*/
function subpub_rules_action_info() {
  return array(...);
}

/**
 * Rules action callback to say something in irc.
 */
function subpub_rules_say($channel, $message, $cb = NULL, $element = NULL) {
  dnode_rpc('subpub', 'say', array($channel, $message), $cb ? $cb : FALSE);
}
```

Rules integration. E.g. on "Create Node", on "Create Comment".

drupal -> dnode -> nodejs

Send notifications to IRC.

rules_subpub_new_node

<demo>

node/add

Bonus I - notifications: drupal<-dnode->nodejs(->browser)

Notifying

rules_subpub_message_broadcast

on message create: broadcast to everyone

subpub: summary

- **nodejs can do high-latency, long-lasting** things easily
- but: **let's keep "our" toolchain** (php, entities, views)

- do latency-related things **before** data get's into drupal
  - e.g. pulling metadata from remote "sources" (e.g. geodata, sentiment analysis)

conclusion

quite a lot of what Jeff Miccolis outlined as the "pain points", is possible in this way w/o giving up on drupal. YAY!

but this is not without costs. polyglotism has overhead, complexity and more moving parts.

drupal_add_serverside_js()?

architectural challenges ahead

- RPC/RMI via dnode or "generic" webservices + message passing?
- sharing "state" and data duplication
  - more "states", duplicated data?
  - coupling systems: too loosely, too tightly?
- devop "issues"
  - configuration, versioned deployment
  - monitoring

questions?

@frega on d.o + twitter

links

dnode_fpd

http://drupal.org/sandbox/frega/1321342

(also have a look at beejeebus' fpd: http://drupal.org/project/fpd)

subpub

http://drupal.org/sandbox/frega/1624652

dnode

http://drupal.org/sandbox/frega/1321342

dnode_faye

http://drupal.org/sandbox/frega/1357240

dnode-php

https://github.com/bergie/dnode-php