drupal + nodejs

erste schritte.

@frega - fl@flink-solutions.de

nodejs?

comet? pubsub?

was ist nodejs?

- Javascript auf dem Server, "evented", asynchrones, non-blocking IO

- Ziel: einfach, schnelle, leichtgewichtige Server und Clients (networked applications) bauen

- Inzwischen in version 0.6, einfach per übliche package management und auch als node.exe zu installieren; guter node package manager (npm, "drush dl" für node)

nodejs "hello world"

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(2000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:2000/');
```

nodejs "hello world 2"

```
var http = require('http');
http.createServer(function (req, res) {
   var incomingNow = new Date();
   setTimeout( function() {
      res.writeHead(200, {'Content-Type': 'text/plain'});
      var respondingNow = new Date();
      res.end('Hello World - incoming request '
              + incomingNow + ' - sending response at '
              + respondingNow + '\n');
   }, 5000);
}).listen(2000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:2000/');
```

nodejs "hello world 99"

```
var _ = require('underscore');
var http = require('http');
var ress = [];
http.createServer(function (req, res) {
   // push responses onto responses array
   ress.push(res);
   res.writeHead(200, {'Content-Type': 'text/plain'});
}).listen(2000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:2000/');
var bottleCount = 99;
setInterval( function() {
   console.log( 'Sending ' + bottleCount + 'bottles ... to ' +
            ress.length + ' subscriber(s)' );
   // publish bottlecount
   _.each( ress, function(r) {
      r.write( bottleCount + "bottles ... \n");
   });
   bottleCount--;
   // terminate when there are any more bottles
   if (bottleCount==0) {
      _.each( ress, function(r) { r.end(); } );
   }
}, 1000);
```

warum nodejs?

- nodejs kann gut, was der LAMP-stack nicht kann
  - "Spezielle" Server (http, chatserver, pubsub)
  - Clients für Datenstreams (Twitter, aber auch z. B Videokonvertierung)
  - viele, lang-andauernde Verbindungen (C10k)

- Javascript ist (für Drupaler) zugänglicher als die anderen Sprachen, in denen man sowas implementieren würde (Java, Python, Erlang)

- nodejs is sehr viel mehr low-level und hat noch keine "CMS" => ergänzt Drupal, ersetzt es nicht!

konkretes beispiel

(inkl. code-schnipsel und live-demo, what could go wrong ...)

eine "shoutbox", d.h. ein kleinstmöglicher chatroom.

shoutbox "specs"

1. auf der startseite soll eine shoutbox sein, auf der jeder authentifizierte benutzer kurze nachrichten hinterlassen kann.

2. unauthentifizierte benutzer können mitlesen.

3. jedem benutzer immer sofort (bis zu) 10 letzten nachrichten anzeigen.

4. wenn neue nachrichten kommen, sollen diese möglichst zeitnah angezeigt werden.

lösungsansatz - traditionell (ajax) - shoutbox.module

- "model" definieren (schema in der .install => DB)
- block, der die letzten 10 aus der DB ausliest
- form-api + #ajax + hook_permission für das formular.
- damit das regelmäßig updated nutzen wir block_refresh. module o.ä.

mit hook_cron räumen wir regelmäßig auf ...

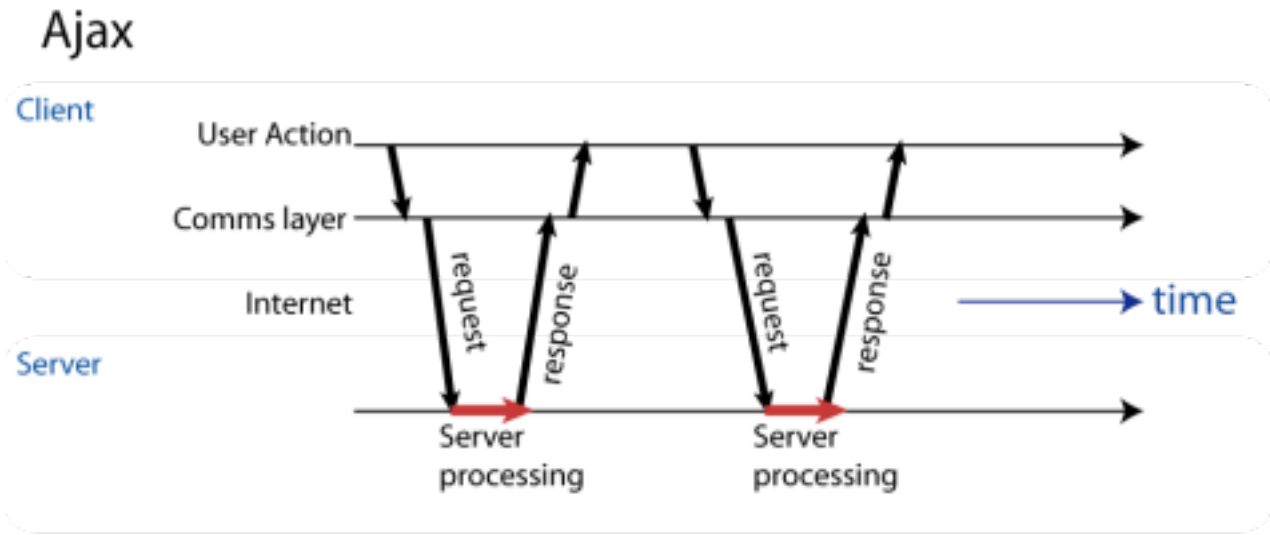
Pro Besucher viele volle Drupal-Bootstraps
Zielkonflikt: "zeitnahe/unmittelbarkeit" vs. Caching/Skalierbarkeit

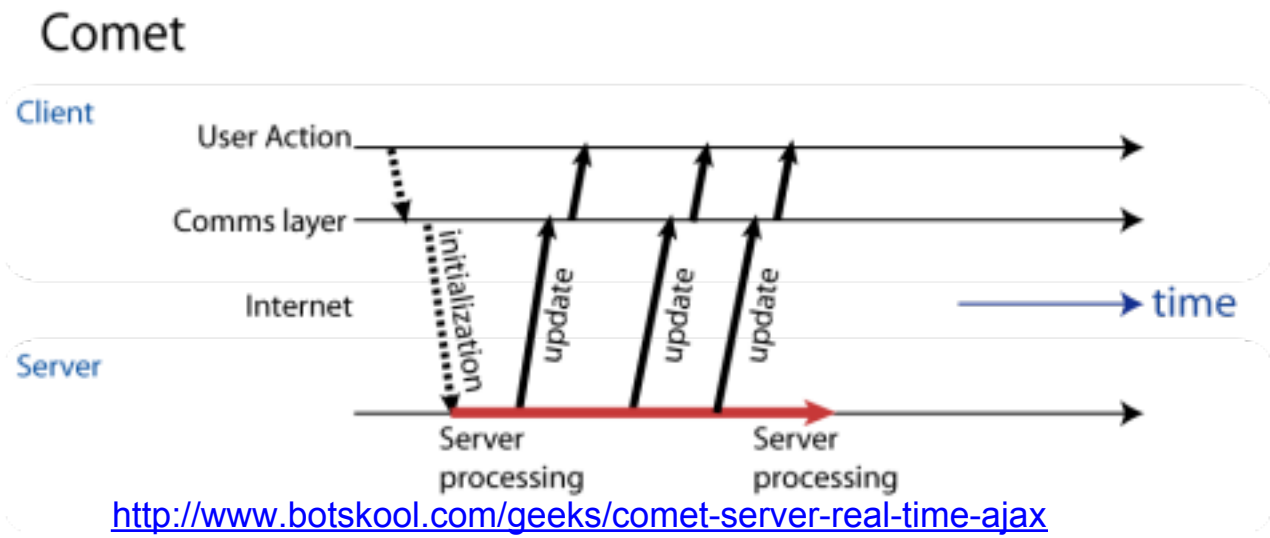
"pubsub"

comet/websockets statt ajax

veröffentlichen und abonnieren

jenseits von ajax & request+response





http://www.botskool.com/geeks/comet-server-real-time-ajax

lösungsansatz (pubsub)

- Pubsub-Server (denkt: Chatserver) => parallele "stehende" Verbindung nach Drupal Response

  Pubsub-Server hat HTTP/"comet" oder Websocket-Transport/Endpoint und Client/Browser-JS Library
- Die Shoutbox ist ein "Kanal" im PubSub-Server (denkt: "Raum/Channel" im Chatserver), man abonniert "Updates"

- Auth: jeder, der die Seite anguckt, kann abonnieren (subscribe); angemeldete benutzer (d.h. Drupal Session + uid) können “in ihm” publizieren (publish+subscribe).

- Der Kanal sollte ein “Gedächtnis” der 10 letzten Nachrichten haben (denkt: "backscroll")

pubsub I

erstmal ohne drupal ...

index.php

```
<html>
  <head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"></script>
    <script type="text/javascript" src="http://<?=$_SERVER['HTTP_HOST'];?>:9000/faye.js"></script>
    <script type="text/javascript"><?='var FAYE_HOST = "'.$_SERVER['HTTP_HOST'].'";';?></script>
    <script src="./shoutbox.js"></script>
  </head>
  <body>
    <div id="shoutbox">
      <h2>Shoutbox</h2>
      <ul></ul>
      <form action="#">
        <input type="text" placeholder="Name" class="name" />
        <input type="text" placeholder="Message" class="message" />
        <input type="submit" value="SHOUT!" />
      </form>
    </div>
  </body>
</html>
```

shoutbox.js

```
(function($) {
  $(document).ready( function() {
    function escape(text) {
      return $('<div/>').text(text).html();
    }
    // subscribe
    var client = new Faye.Client('http://' + FAYE_HOST + ':9000/faye');
    client.subscribe('/shoutbox', function(data) {
      $('#shoutbox ul').prepend( '<li><strong>' + escape(data.name) + '</strong>: ' + escape(data.messa
      if ($("#shoutbox li").length > 10) {
        $('#shoutbox li:last').remove();
      }
    });
    // publish
    $('#shoutbox form').submit( function() {
      client.publish('/shoutbox', {
        name: $('#shoutbox input.name').val(),
        message: $('#shoutbox .message').val()
      });
      $('#shoutbox .message').val('');
      return false;
    });
  });
}(jQuery));
```

server.js

```
var http = require('http'),
    faye = require('faye');
var bayeux = new faye.NodeAdapter({mount: '/faye', timeout: 45});
// HTTP Server
var server = http.createServer();
bayeux.attach(server);
server.listen(9000);
```

pubsub

<demo/>

drupal + nodejs - integration

Optionen:

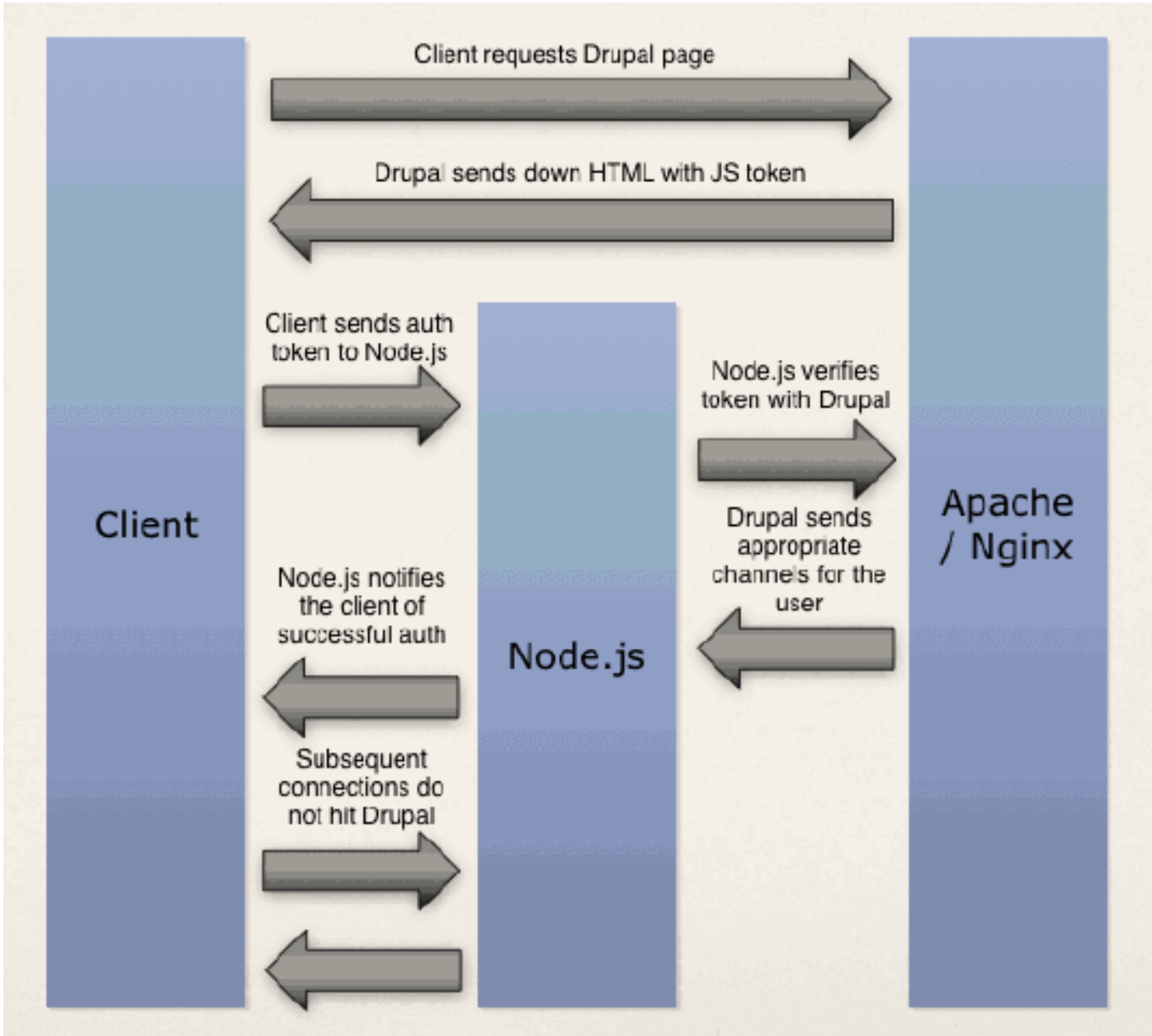
- nodejs.module ("D6"/D7) - "beta"-ish, 1.0 kommt!
- dnode_faye (D7, PHP5.3+) - sandbox - very "alpha"-ish

Was machen die Module?

- Authentifizierung / Autorisierung
- Drupal sessionId <-> (pubsub-)clientId Assozierung
- Messaging, "RPC", Notifizierung ...

Architektonische Unterschiede, aber im Prinzip sehr ähnlich.

Drupal <-> Nodejs



@tizzo/@beejeebus' drupalcon presentation - http://london2011.drupal.org/conference/sessions/nodejs-and-drupal

drupal + nodejs - dnode_faye_chatterbox.module

```php
<?php
/** Implements hook_permission. ...*/
function dnode_faye_chatterbox_permission() {...}

function dnode_faye_chatterbox_block_info() {...}

function dnode_faye_chatterbox_block_view($delta) {
  switch ($delta) {
    case 'chatterbox';
      $block['content'] = '<div id="dnode-faye-chatterbox-wrapper"><ul id="dnode-faye-chatterbox"></ul>';

      drupal_add_js( drupal_get_path('module', 'dnode_faye_chatterbox') . '/dnode_faye_chatterbox.js' );
      if (user_access('dnode_faye_chatterbox post to chatterbox')) {
        global $user;
        $block['subject'] = t('CHATTERBOX');
        dnode_faye_add_subscription_and_publication('/dnode_faye_chatterbox', TRUE);
        $block['content'] .= '<form><input type="text" id="dnode-faye-chatterbox-text" />
                  <input type="submit" id="dnode-faye-chatterbox-submit" value="CHAT"/></form>';
        drupal_add_js(array('dnode_faye_chatterbox' => array('name' => $user->name)), 'setting');
      } else {
        drupal_set_message( t('You cannot write in the chatterbox - but you may bear witness!') );
        dnode_faye_add_subscription('/dnode_faye_chatterbox');
        $block['subject'] = t('CHATTERBOX - READ-ONLY');
      }
      $block['content'] .= '</div>';
      return $block;

  }
}
```

drupal + nodejs - chatterbox.js

```
(function($, Drupal){
  Drupal.behaviors.dnode_faye_chatterbox = {
    attach: function (context, settings) {
      // let's set up the most
      $('#dnode-faye-chatterbox-wrapper').once( function() {
        $('form', $(this)).submit(function() {
          var v = $('#dnode-faye-chatterbox-text').val();
          $('#dnode-faye-chatterbox-text').val('');
          if (v) {
            Drupal.Faye.client.publish('/dnode_faye_chatterbox', {
              'body': v,
              'name': settings.dnode_faye_chatterbox.name,
              'callback': 'dnode_faye_chatterbox'
            });
          }
          return false;
        });
        Drupal.Faye.addObserver('dnode_faye_chatterbox', function(channel, message) {
          if ( message.callback == 'dnode_faye_chatterbox' ) {
            var n = $("#dnode-faye-chatterbox li").length;
            if (n > 10) {
              $('#dnode-faye-chatterbox li:last').remove();
            }
            if (message.name) {
              var c = Drupal.checkPlain( message.name );
              c = '<strong> ' + c + '</strong>: ';
            } else {
              c = '';
            }
            $('<li>' + c + Drupal.checkPlain(message.body) + '</li>').prependTo( $('#dnode-faye-chatterbox') );
          }
        });
      });
    }
  };
})(jQuery, Drupal);
```

drupal + nodejs - chatterbox.js PUBLISH

```
$('form', $(this)).submit(function(e) {
  var v = $('#dnode-faye-chatterbox-text').val();
  if (v) {
    Drupal.Faye.client.publish('/dnode_faye_chatterbox', {
      'body': v,
      'name': settings.dnode_faye_chatterbox.name,
      'callback': 'dnode_faye_chatterbox'
    });
  }
  $('#dnode-faye-chatterbox-text').val('');
  return false;
});
```

drupal + nodejs - chatterbox.js UPDATES

```
Drupal.Faye.addObserver('dnode_faye_chatterbox', function(channel, message) {
  if ( message.callback == 'dnode_faye_chatterbox' ) {
    var n = $("#dnode-faye-chatterbox li").length;
    if (n >= 10) {
      $('#dnode-faye-chatterbox li:last').remove();
    }
    var c;
    if (message.name) {
      c = '<strong> ' + Drupal.checkPlain( message.name ) + '</strong>: ';
    } else {
      c = '';
    }
    $('<li>' + c + Drupal.checkPlain(message.body) + '</li>')
          .prependTo( $('#dnode-faye-chatterbox') );
  }
});
```

dnode_faye_chatterbox

<demo/>

drupal + nodejs - zusammenfassung

- Immer noch komplexe Toolchain, aber besser als früher.
- Richtigen Stack für den Anwendungsfall wählen & bestehende Lösungen nutzen
- Höhere Komplexität mit dem Nutzen abwägen.
- Es muß nicht immer AJAX sein, nicht alles muß in und durch Drupal gelöst werden.

danke - fragen?

@frega
fl@flink-solutions.de

der kunde respecct ...

- Es sollen jetzt keine Nachrichten mehr erlaubt sein, die bestimmte Begriffe beinhalten ...
- z.B. "Bieber" oder "Typo3" oder so ...

dnode_faye_chatterbox - server-side javascript (SSJS)

Wir benutzen Faye als pubsub-Server (ähnliches gilt aber für socket.io "middleware" o.ä., die im nodejs.module implementiert werden könnten).

Faye Server-side extensions können

- incoming messages
- outgoing messages

filtern / verändern / abbrechen / validieren usw.

Messages statt $_POST und $_GET

dnode_faye_chatterbox - extensions

```
var chatterboxExtension = {
  // every incoming message passes through this *ONCE*
  'incoming': function(message, callback) {
    if ( (message.channel == '/dnode_faye_chatterbox' ) && (!message.error) ) {
      debug && console.log('MESSAGE - incoming chatterbox', message);
      // filter, if filter is set => demonstrate aborting messages
      if (
        config.filter &&
        message.data.body.toLowerCase().indexOf( config.filter.toLowerCase() ) != -1
      ) {
        debug && console.log('MESSAGE - FILTERED!', message);
        message.error = config.filter + ' must not be in message!';
        return callback(message);
      }
      // everybody SHOUTING UPPERCASE.
      if (config.uppercaseMessages && message.data.body) {...}
      // ensure user IS correct
      if (config.retrieveUsername) {...} else {...}
    } else {
      debug && console.log('MESSAGE - incoming chatterbox', message);
      callback(message);

    }
  },
  // every outgoing message passes through this *ONCE*
  'outgoing': function(message, callback) {...}
}
```

dnode_faye_chatterbox

<demo2/>
extensions (textfilter, uppercase, username)

dnode_faye_chatterbox - SSJS Faye extensions

- Extensions ergänzen Funktionalitäten
  - Input Filtern (filter)
  - Input verändern (UPPERCASE)
  - Input Validieren (user name)

- Weitere Beispiele:
  - "Channelgedächtnis" ist eine Extension ("backlog", message.ext)
  - Authentifizierung ist eine Extension (checkt tokens)