and

express

Aaron Bartell abartell@krengeltech.com

LITMIS.COM - Open Source for i

Agenda

- Javascript for noobs
- What is Express.js?
- Installing Express.js
- Writing Express.js app

A Little Javascript for noobs

The `require` feature in Node.js is similar to RPG's `/copy` - bring in outside functionality. See below "**Answers**" link because this is a busy topic.

```
1.  var ibmi = require('/home/aaron/git/nodejs_playing/ibmi');
2.  var user = ibmi.current_user();
```

The `exports` syntax declares the functions to make available, like `export` on an RPG sub procedure.

ibmi.js

```
1.  function current_user(){
2.    return rtvjoba('USER');
3.  }
4.
5.  exports.current_user = current_user;
```

http://stackoverflow.com/questions/16521471/relation-between-commonjs-amd-and-requirejs - Answers

noobs continued...

Javascript **callbacks** are like RPG **procedure pointers...**

```
1.  function my_callback(data) {
2.    console.log('data: ' + data);
3.  }
4.
5.  function hi(callback) {
6.    callback('get it?');
7.  }
8.
9.  hi(my_callback);
```

callbackhell.com - Learn how to write Javascript callbacks
javascriptissexy.com/understand-javascript-callback-functions-and-use-them - Callback functions

noobs continued...

Anonymous function assigned to a variable...

```
1.  var my_callback = function(data) {
2.    console.log('data: ' + data);
3.  };
4.
5.  function hi(callback) {
6.    callback('get it?');
7.  }
8.
9.  hi(my_callback);
```

w3schools.com/js/js_function_definition.asp - Anonymous functions

noobs continued...

Inline anonymous function...

```
1.  function hi(callback) {
2.    callback('get it?');
3.  }
4.
5.  hi(function(data) {
6.    console.log('data: ' + data);
7.  });
```

w3schools.com/js/js_function_definition.asp - Anonymous functions

What is Node.js? Make me care.

Server-side Javascript, uses Google's V8

Biggest benefit: One language for both client *(browser)* and server *(IBM i)*.

BIG TIME SAVER!!

Ported and supported by IBM.

Home: http://bit.ly/nodejs_ibmi

Non-blocking I/O. Not unique to Node.js, though they made it popular.

w3schools.com/js - Javascript, getting started
developers.google.com/v8 - Google V8
en.wikipedia.org/wiki/MIT_License - MIT license

What is Express.js?

express

You *could* write everything from scratch. **But why?** Frameworks get you to goals faster!

"Fast, unopinionated, minimalist web framework for Node.js"

Features:

- Routing
- Middleware
- Error handling
- Debugging

Most popular Node.js web framework!

expressjs.com - home
github.com/strongloop/express - repo
npmjs.com/package/express - package

Config IBM i Environment

Do yourself a favor and DON'T use `CALL QP2TERM`, use SSH client instead.

Run the following in a PASE shell:

```
$ export PATH=/QOpenSys/QIBM/ProdData/Node/bin:$PATH
$ export LIBPATH=/QOpenSys/QIBM/ProdData/Node/bin:$LIBPATH
```

Now you can invoke node from anywhere in the IFS.

```
$ node -v
v0.10.29
```

Put the above exports into `node_env.sh` and then easily invoke it *(note the period!)*

```
$ . node_env.sh
```

bitbucket.org/litmis/nodejs/wiki/environment - IBM i Node.js Environment config

installation

```
$ mkdir expressjs_app && cd expressjs_app
$ npm install express
express@4.12.0 node_modules/express
├── utils-merge@1.0.0
├── methods@1.1.1
├── fresh@0.2.4
├── merge-descriptors@0.0.2
├── cookie-signature@1.0.6
├── escape-html@1.0.1
├── range-parser@1.0.2
├── cookie@0.1.2
├── finalhandler@0.3.3
├── vary@1.0.0
├── content-type@1.0.1
├── parseurl@1.3.0
├── content-disposition@0.5.0
├── serve-static@1.9.1
├── path-to-regexp@0.1.3
├── depd@1.0.0
├── on-finished@2.2.0 (ee-first@1.1.0)
├── qs@2.3.3
├── debug@2.1.1 (ms@0.6.2)
├── proxy-addr@1.0.6 (forwarded@0.1.0, ipaddr.js@0.1.8)
├── etag@1.5.1 (crc@3.2.1)
├── send@0.12.1 (destroy@1.0.3, ms@0.7.0, mime@1.3.4)
├── type-is@1.6.0 (media-typer@0.3.0, mime-types@2.0.9)
└── accepts@1.2.4 (negotiator@0.5.1, mime-types@2.0.9)
```

Dependencies

first app

/home/aaron/expressjs_app/**app.js**

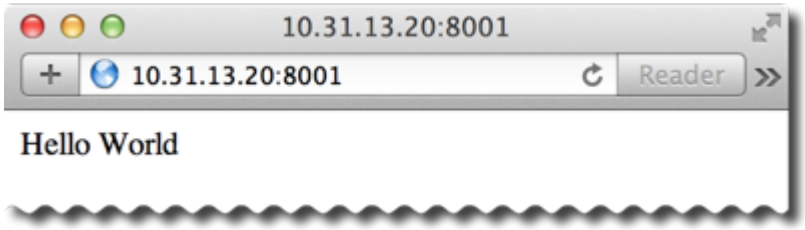
```
1.  var express = require('express')
2.  var app = express()
3.
4.  app.get('/', function (req, res) {
5.    res.send('Hello World')
6.  })
7.
8.  app.listen(8001)
```

1: Write it

2: Start it

```
$ node app.js
```

3: Display it



without Expressjs *(comparison)*

Manually handle request routing

```
1.  var http = require('http');
2.  http.createServer(function (req, res) {
3.    res.writeHead(200, {'Content-Type': 'text/plain'});
4.    res.end('Hello World\n');
5.  }).listen(1337, '127.0.0.1');
```

Manually write headers

"Hello World" not good comparison of Expressjs advantages.

Routing

...determines what code should execute when given a particular path.

Syntax: `app.METHOD(path, [callback...], callback)`

```
1.  app.get('/users', function (req, res) {
2.    //Return list of users in database
3.  })
4.
5.  app.post('/user', function (req, res) {
6.    //Write user row to database
7.  })
8.
9.  app.put('/user/:id', function (req, res) {
10.   //Update user row in database
11. })
12.
13. app.delete('/user/:id', function (req, res)
    {
14.  //Delete user row in database
15. })
```

expressjs.com/guide/routing.html - Tutorial
expressjs.com/4x/api.html#router - Docs

Static Files (ExpressJS built-in middleware)

Server file system:

```
expressjs_app
 |-public
    |-images
    |-css
    |-js
```

Uses middleware `express.static`

app.js:

```
1.  app.use(express.static('public'));
```

Now the following paths are available:

http://domain.com**/images/logo.jpg**
http://domain.com**/css/style.css**
http://domain.com**/js/app.js**
http://domain.com**/index.html**

expressjs.com/starter/static-files.html - Docs
stackoverflow.com/questions/9967887/node-js-itself-or-nginx-frontend-for-serving-static-files - Discussion

package.json

holds meta-data about application

```
$ npm init
```

Creates file `package.json`

```
1.  {
2.    "name": "expressjs_app",
3.    "version": "0.0.0",
4.    "description": "",
5.    "main": "app.js",
6.    "dependencies": {
7.      "express": "^4.12.0"
8.    },
9.    "devDependencies": {},
10.   "author": "Aaron Bartell",
11.   "license": "ISC"
12. }
```

Installs these modules when `npm install` is run.

docs.npmjs.com/cli/init - package.json creation
docs.npmjs.com/files/package.json - Docs
browsenpm.org/package.json - Easier docs

Template Engine


```
$ npm install jade --save
jade@1.9.2 node_modules/jade
├── character-parser@1.2.1
├── commander@2.6.0
├── void-elements@2.0.1
├── mkdirp@0.5.0 (minimist@0.0.8)
├── with@4.0.1 (acorn-globals@1.0.2, acorn@0.11.0)
├── transformers@2.1.0 (promise@2.0.0, css@1.0.8, uglify-js@2.2.5)
└── constantinople@3.0.1 (acorn-globals@1.0.2)
```

package.json

```
{
  "name": "expressjs_app",
. . .
  "dependencies": {
    "express": "^4.12.0",
    "jade": "^1.9.2"
  },
  . . .
```

jade-lang.com - Home

Template Engine

jade
Node Template Engine

```
$ mkdir views
```

Directory to store all views.

views/index.jade

```
1.  html
2.    head
3.      title!= title
4.    body
5.      h1!= message
```

Jade syntax

app.js

```
1.  app.set('views', __dirname + '/views');
2.  app.set('view engine', 'jade')
3.
4.  app.get('/', function (req, res) {
5.    res.render('index', { title: 'Hey', message: 'Hello there!'});
6.  })
```

expressjs.com/guide/using-template-engines.html - Tutorial

Template Engine (database result set)

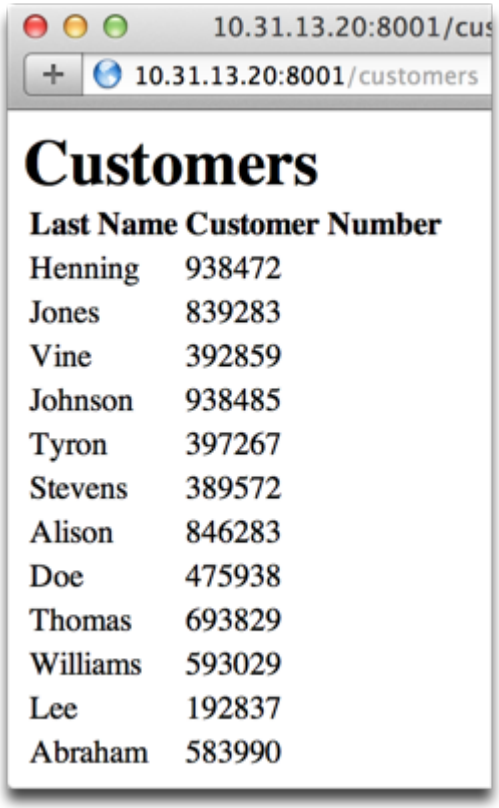
app.js

```
1.  var db = require('/QOpenSys/QIBM/ProdData/Node/os400/db2i/lib/db2')
2.  db.init()
3.  db.conn("*LOCAL")
4.   . . .
5.  app.get('/customers', function (req, res) {
6.    db.exec("SELECT LSTNAM, CUSNUM FROM QIWS.QCUSTCDT", function(results) {
7.      res.render('customers', { title: 'Customers', results: results})
8.    })
9.  })
```

views/customers.jade

```
1.  h1=title
2.    table
3.      thead
4.        tr
5.          th Last Name
6.          th Customer Number
7.      tbody
8.        - each row in results
9.          tr
10.           td=row.LSTNAM
11.           td=row.CUSNUM
```

jade-lang.com/reference/iteration - Iteration



Template Engine (link to another page)

views/customers.jade

```
tbody
  - each row in results
    tr
      td=row.LSTNAM
      td: a(href='/customer/#{row.CUSNUM}')=row.CUSNUM
```

app.js

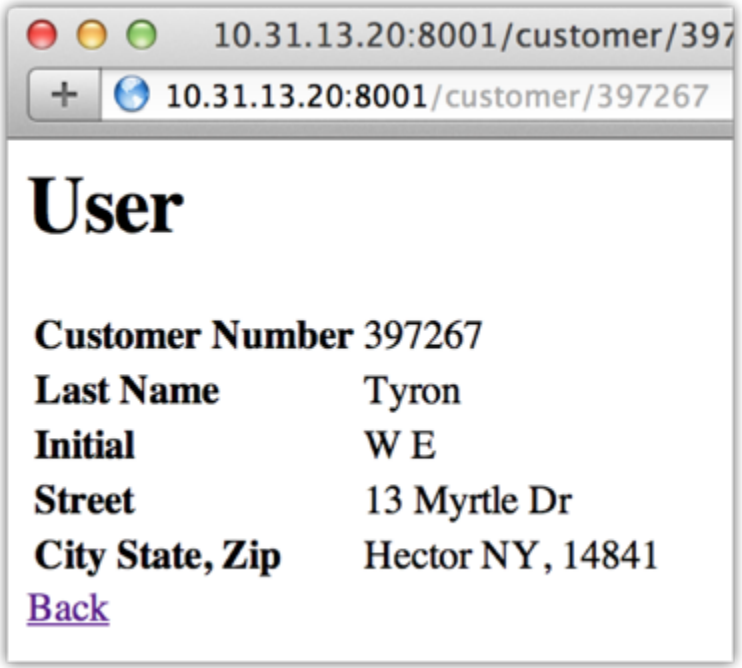
```
app.get('/customer/:id', function (req, res) {
  var sql = "SELECT * FROM QIWS.QCUSTCDT WHERE CUSNUM=" + req.params.id;
  db.exec(sql, function(result) {
    res.render('user', { title: 'User', result: result[0]});
  })
})
```

jade-lang.com/reference/attributes - Attributes

Template Engine (link to another page)

views/user.jade

```
1.  h1=title
2.  table
3.    tr
4.      td: b Customer Number
5.      td=result.CUSNUM
6.    tr
7.      td: b Last Name
8.      td=result.LSTNAM
9.    tr
10.     td: b Initial
11.     td=result.INIT
12.   tr
13.     td: b Street
14.     td=result.STREET
15.   tr
16.     td: b City State, Zip
17.     td= result.CITY + " " +
18.         result.STATE + ", " +
19.         result.ZIPCOD
20.
21.  a(href='/customers') Back
```



Middleware

"An Express application is essentially a series of middleware calls."

Middleware is a **function with access to the request object** (`req`)**, the response object** (`res`).

Middleware can:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware in the stack.

Syntax

```
1.  app.use(... middleware function call ...);
```

expressjs.com/guide/using-middleware.html - Tutorial
blog.safaribooksonline.com/2014/03/10/express-js-middleware-demystified - Tutorial

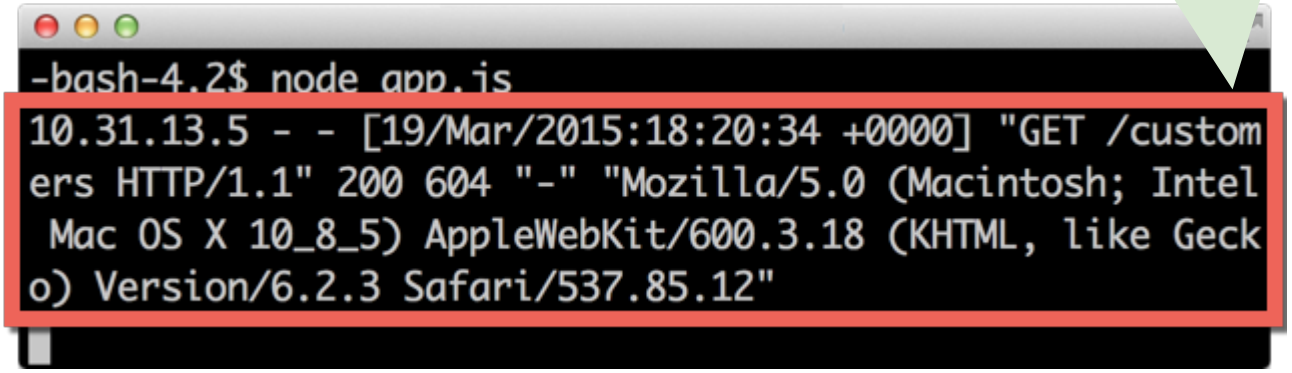
Middleware example

morgan

```
$ npm install morgan --save
```

```
1.  var express = require('express')
2.  var app = express()
3.  var morgan = require('morgan')
4.
5.  app.use(morgan('combined'))
```

Logs now show in console



expressjs.com/resources/middleware.html - Common middleware

Errors

four arguments instead of three, `(err, req, res, next)`

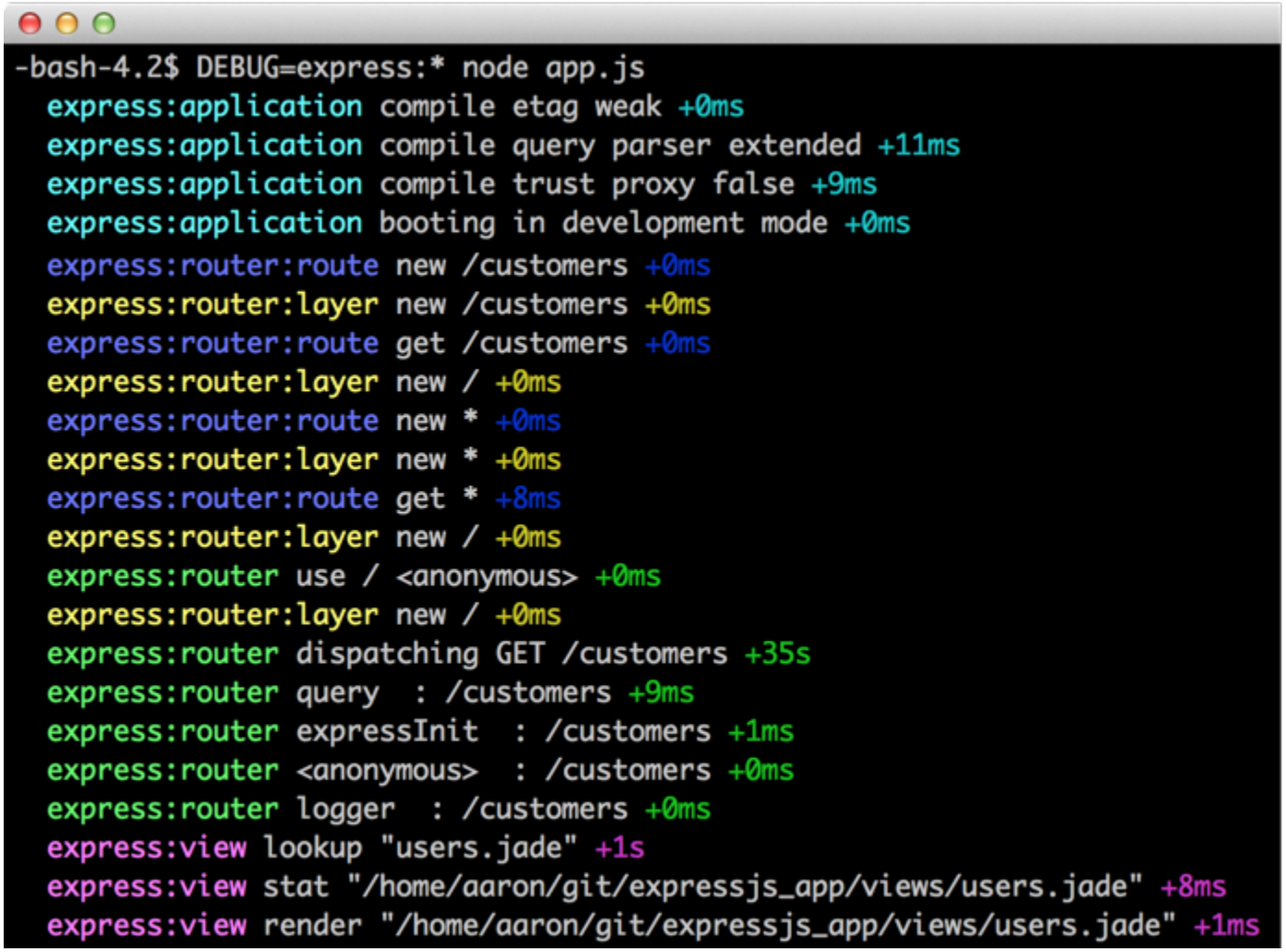
```
1.  app.get('/', function (req, res) {
2.    res.send('Hello World')
3.  })
4.
5.  app.get('*', function(req, res, next) {
6.    var err = new Error();
7.    err.status = 404;
8.    next(err);
9.  })
10.
11. app.use(function(err, req, res, next) {
12.   if(err.status !== 404) {
13.     return next();
14.   }
15.   res.send(err.message || '** Ponder life while we look into this.
    **');
16. })
```

Falls through to here if no other route matches.

One parm on `next` will invoke the error function *(i. e. four parameters)*

expressjs.com/guide/error-handling.html - Tutorial
blog.safaribooksonline.com/2014/03/12/error-handling-express-js-applications - Tutorial

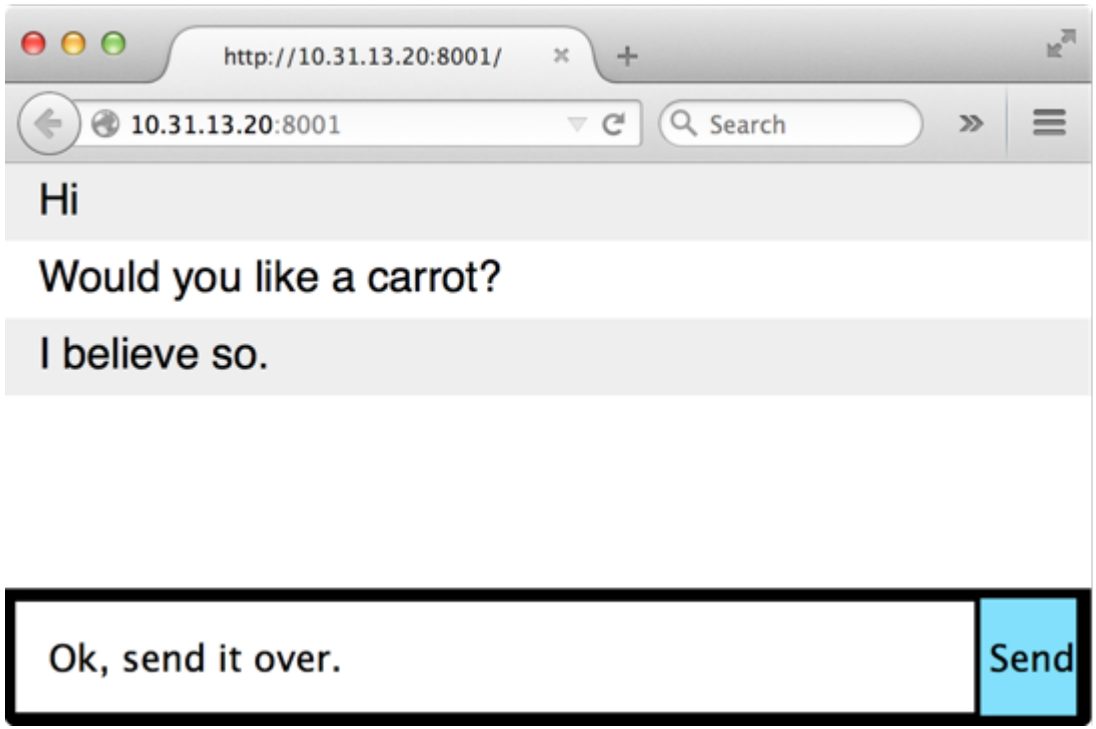
Debugging



expressjs.com/guide/debugging.html - Tutorial

IBM i Chat

- Runs on **IBM i**
- Written in **Node.js**
- Uses HTML5's **WebSockets** to communicate
- **socket.io** Javascript library primary means.



server.js

```
1.  var app  = require('express')();
2.  var http = require('http').Server(app);
3.  var io   = require('socket.io')(http);
4.
5.  app.get('/', function(req, res){
6.    res.sendfile('index.html');
7.  });
8.
9.  io.on('connection', function(server){
10.   server.on('disconnect', function(){
11.     console.log('user disconnected');
12.   });
13.   server.on('chat message', function(msg){
14.     io.emit('chat message', msg);
15.   });
16. });
17.
18. http.listen(8001, function(){
19.   console.log('listening on *:8001');
20. });
```

index.html

```
<html>
  <body>
    <ul id="messages"></ul>
    <form action="">
      <input id="m" autocomplete="off" /><button>Send</button>
    </form>
    <script src="https://cdn.socket.io/socket.io-1.2.0.js"></script>
    <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
    <script>
      var client = io();
      $('form').submit(function(){
        client.emit('chat message', $('#m').val());
        $('#m').val('');
        return false;
      });
      client.on('chat message', function(msg){
        $('#messages').append($('<li>').text(msg));
      });
    </script>
  </body>
</html>
```

order of events

1: `$ node server.js`

2: Initial browser request to http://domain.com

3: `server.js` sends back index.html

4: `index.html` "runs" in the browser, initializing the javascript

5: Client-side `io()` method negotiates Websocket connection with `server.js`

6: Websocket connection established

7: User types message, hits enter.

8: `client.emit()` API is fired which communicates over websocket to `server.js`.

9: `server.js` receives in data and `server.on('chat message'…)` is fired.

10: `io.emit('chat message', msg)` communicates to all clients with `client.on('chat message'…)` specified.

We Have Reached The End!

Aaron Bartell

abartell@krengeltech.com - @aaronbartell

KRENGELTECH

LITMIS.COM - Open Source for i