Today’s Agenda

• Showcase: Spotify Search
• NodeJS:
  – Setup
  – Express Generator
  – Middleware
• Break Out
• Quiz
Evaluation & Feedback

Please help us improve the tutorials and assignments by filling out this survey:

http://goo.gl/forms/DTdGh4TYaC
NodeJS - About

• What is it?
  “Node.js® is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications” (official website, nodejs.org)

• Node apps are JavaScript files!

• Why do we want it?
  – non-blocking I/O
  – highly scalable
  – web-apps can act as standalone web-server
  – largest ecosystem open source libraries
NodeJS – Who is using it?

https://github.com/nodejs/node-v0.x-archive/wiki/Projects,-Applications,-and-Companies-Using-Node
WordPress.com: Making the Switch

Introducing The New WordPress.com

A single interface to manage all your WordPress.com or Jetpack-enabled sites, built with the latest web technologies and used by millions of people — and now it's open source.

Installing NodeJS

• Windows:
  – Download installer and run it
    https://nodejs.org/en/download/
  – make sure to install npm during the installation (default)

• OS X
  – Option 1: download and install package from nodejs.org
    https://nodejs.org/en/download/
  – Option 2: Homebrew
    brew install node
    http://shapeshed.com/setting-up-nodejs-and-npm-on-mac-osx/

• Linux:
  – please follow the instructions provided here:
    https://github.com/nodejs/node-v0.x-archive/wiki/Installing-Node.js-via-package-manager

• Find out if the installation was successful. Type in a terminal:
  – node \-v
  – npm \-v
Starting a node app from the Command Line

• On your own machine:
  $ node <path_to_file>

• On a CIP pool computer:
  $ nodejs <path_to_file>
Hello World!

```javascript
var http = require('http');
var port = 8976;
var host = '127.0.0.1';

var server = http.createServer(function (request, response) {
  response.writeHead(200, {'Content-Type': 'text/plain'});
  response.end('Hello World\n');
});

server.listen(port, host);
console.log('Server running at http://'+host+':'+port+'/');
```

Then in a the terminal type:

```
$ node helloworld.js
```

```
Hello World
```

`examples/helloworld.js`
Node Package Manager

• Node is highly modular and extensible with “packages”

• npm allows you to easily handle packages from the command line and declare them as dependencies

• Most important operations:
  – Global package installation
    npm install --g PACKAGE [PACKAGE2 PACKAGE3 ...]
  – Local package installation (only for the app)
    npm install PACKAGE [PACKAGE2 PACKAGE3 ...]
  – Local package installation & saving to dependencies list:
    npm install --save PACKAGE [PACKAGE2 ...]
Package script – package.json

```
{
  "name": "examples",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodejs app.js"
  },
  "author": "",
  "license": "MIT",
  "dependencies": {
    "body-parser": "^1.14.1"
  }
}
```

$ npm install
installs all dependencies that are listed in the package script into the node_modules directory
Require

• To import a module, you can use require():
  ```javascript
  var http = require('http');
  ```

• Mind the difference:
  ```javascript
  require('module');
  require('./module');
  ```
  - Without path specification, node tries to import from the node_modules directory (dependencies managed by npm)
  - With path specification:
    * Your own modules
    * Need to call this: module.exports = something;
Hands On: Create your first node app

• Open a bash and type
  `mkdir tutorial07 && cd tutorial07
  npm init`

  follow the wizard (always hitting the return key is okay).

• Now you’ll have your own package script package.json to
which you can add your dependencies
Express – a web application framework

• One of the most popular NodeJS frameworks.
• Characteristics:
  – minimalistic
  – easy to use API
  – many utility methods and middleware functionalities
  – thin layer on top of NodeJS
• Side notes:
  – responsible for the letter E in the MEAN stack

• Find the documentation here: http://expressjs.com/
Basic Express App

```javascript
var express = require('express');
var app = express();

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

var server = app.listen(3000, function () {
    var host = server.address().address;
    var port = server.address().port;
    console.log('app listening at http://%s:%s', host, port)
});
```
Express generator

- Goal: automatically generate the basic structure of an express app that includes views, routes, common dependencies
- Requirements: Install the generator globally:
  $ npm install -g express-generator
  $ express MMNExpressApp
- Documentation: 
  http://expressjs.com/starter/generator.html
- You still have to install the dependencies manually:
  $ cd MMNExpressApp && npm install
Express Generator: bin/www

• The server process is started here (executable file)
  
  $ node bin/www

• Port:

  ```javascript
  var port = normalizePort(process.env.PORT || '3000');
  ```
  
  – normalizePort is also defined in this file. It simply tries to parse the port to an integer.
  
  – process.env.PORT: accesses the environment variable “PORT”
  
  – If there is no such environment variable, the default ‘3000’ is used.

• Common error handlers for EACCES and EADDRINUSE
Express Generator: app.js

• All non-core modules for express are imported here, e.g.
  – `var bodyParser = require('body-parser');`
    Middleware that parses HTTP message bodies and stores the result in
    req.body (POST) or req.query (GET)
  – `var logger = require('morgan');`
    HTTP Request logger (generates console output)

• Middleware and basic routing setup
  
  `var routes = require('./routes/index');`
  `app.use('/', routes);`
Express Generator: routes/*.js (1)

• Routing is the definition of end points (URIs)
• With a node app, routing is a little more elaborate than with PHP scripts.
  – With PHP/Apache: scripts in different folders, accessible immediately https://localhost/folder/subfolder/subsubfolder/script.php
  – Express: more details how to handle paths and routing
• Route ≈ Path ≈ Mountpoint ≈ Endpoint
• Example:
  ```javascript
  app.get('/a-route/sub-route', function(req,res){
    res.send('You are on a sub-route. ')
  });
  ```

http://expressjs.com/starter/basic-routing.html
Express Generator: routes/*.js (2)

• Express includes a lightweight router module
  `var router = express.Router();`

• Defining a route that handles GET requests:
  `router.get('/getRoute', function(req, res) {`
  `  res.send('hello world');`
  `});`

• Defining a route to handle POST requests:
  `router.post('/postRoute', function(req, res){`
  `  res.send('hello world, from post!');`
  `});`

• To use the route in the app, the router object needs to be exported:
  `module.exports = router;`

http://expressjs.com/guide/routing.html
Express Generator: views/*.jade

- Views are templates that are compiled by the webapp
- The default rendering engine is Jade
- routes/index.js shows how a template is rendered:

```javascript
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});
```

- The rendering parameters are passed as JavaScript object
Middleware

- A middleware is a function that sits between the request and the response (“in the middle”)
- Does something with the request. Typical tasks:
  - parse the HTTP message body to a JSON object (body-parser)
  - parse cookies (cookie-parser)
  - authenticate the user and (dis)allow a request
- Usually more than one middleware per route (middleware chain)

```javascript
app.use('/', function(req, res, next){
  // do something with req.object
  // then either send the response or call:
  next();
});
```

- Express router = middleware!
Serving static files and directories

• Express has a built-in middleware to serve directories, e.g. for HTML, CSS, JavaScript, or image files: `express.static`

• Example usage in `app.js` to serve a directory named “public”: `app.use(express.static('public'))`;

• Content of the directory is accessible from the server root: `http://localhost:3000/images/kitten.jpg`
  `http://localhost:3000/hello.html`

• You can also specify a mount-path: `app.use('/kittens', express.static('images/kittens'))`;

• The path is relative to the node process, so do this: `app.use(express.static(__dirname + '/public'))`;

http://expressjs.com/starter/static-files.html
Break Out: Your First Node App

• Use the express generator to initialize a node app

• Play around with it and try the following:
  – add a new route /mmn
    • define it first in routes/index.js
    • then try to re-wire the route as new file /routes/mmn.js
  – respond with a dynamically generated JSON object that contains the current date-time and UNIX timestamp:
    \begin{verbatim}
    {
        millis: 1448731402557
    }
    \end{verbatim}

• Timeframe: 40 Minutes
Good to know...

• Never add the “node_modules” directory to version control (Git). It is enough to declare the dependencies in package.json

• The EADRRINUSE error:
  – The port is already used by another application
  – Try a different port (high number) or
  – Quit the application that’s using the port, e.g.
    $ killall node
Some more links

- http://nodeschool.io/
- http://nodeguide.com/beginner.html
- http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop/
- http://docs.nodejitsu.com/articles/getting-started/what-is-require
- http://docs.nodejitsu.com/articles/getting-started/npm/what-is-npm
Round-up Quiz

1. Name one difference between the NodeJS and Apache
2. What does require(...) do?
3. How do you generate a package script?
4. How do you conveniently save a dependency on a module into the package script?
5. What does the body-parser middleware do?
6. What is a middleware?
Thanks!

What are your questions?
Sources

• http://www.talentbuddy.co/blog/building-with-node-js-at-ebay/
• http://www.talentbuddy.co/blog/building-with-node-js-at-netflix/
• http://venturebeat.com/2011/08/16/linkedin-node/