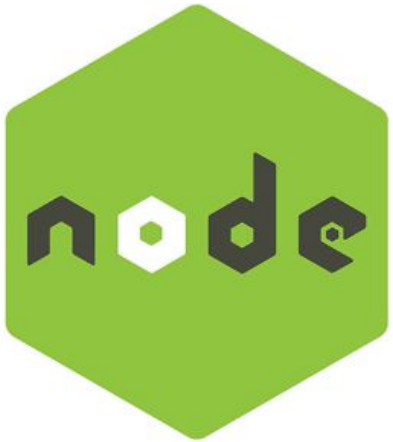
UNIVERSITY OF SZEGED
Department of Software Engineering
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS



Practice 6
NodeJS + Express

Topics

- NodeJS basics
- ExpressJS - REST endpoint creation
- Simple example, REST calls with Postman

NodeJS

- JavaScript runtime environment for running server tools and web applications
- Event-driven architecture, asynchron capability, callback-based execution
- Every file is handled as a single module, the modules can create connections between each other, call async functions from each other

```
const passport = require("passport");
        passport.initialize()
```

- Easy to scale, very popular thanks to the wide usage of JavaScript

NodeJS server

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end("Hello From NodeJS");
}).listen(9615);
```

- The lines above create and run a simple NodeJS server

```
node filename.js
```

- It will listen to requests on port 9615, and answers with a simple request handling function
- A good foundation, with a large amount of frameworks available for creating webapplications and servers

NPM – Node Package Manager

- We've met with this already
- The package manager of NodeJS, gets installed with it
- Most important commands and elements

```
npm init
```

Creates the package.json file in the actual working directory

```
package.json
```

Contains the project's name, developer, dependencies, optionally even some simple scripts for running the project

```
npm install
```

Reads every dependency from package.json and installs them in a directory called node_modules (after that, we can reference them in NodeJS scipts with the require function)

NPM 2.

```
npm install module_name --save
```

Downloads the module_name module into node_modules, also modifies the package.json file by inserting this new dependency

```
npm install module_name -g
```

Downloads the module, and makes it globally available, even outside from the current working directory (this is how we install compilation tools, generator modules, command line interfaces)

```
npm start
```

The package.json is capable of containing simple scripts, this will run the script named start (typically used to start up the project contained in the directory)

ExpressJS

- The node_modules directory is usually referenced in the .gitignore file – there's no reason to upload it to git, since an npm install will always download every dependency
- Usually the first step when developing a NodeJS based server: `npm install express --save`
- Express is a NodeJS based framework, which can be used to create REST endpoints, worfkflows for handling requests
- With NodeJS it is the backend of the MEAN development stack

MongoDB **E**xpress **A**ngular **N**odeJS

ExpressJS app development

```
//server.js
const express = require("express");
var app = express();
app.use('/example', require('./routes')) //we create the example
route, it's concrete routes and supported operations are in the
routes module
app.listen(5000, () => {
    console.log('The server is running');
}); //The app will listen on port 5000

// routes.js
var express = require("express");
var router = express.Router(); // with Route, we can create
portable, easy-to-use route definitions
router.route('/greeting').get(function (req, res, next) {
    return res.status(200).end('Ez egy csoda');
}); //the greeting route is created, ready for http get
module.exports = router; //the created router is exported (this
will be in place of the require call in our main module)
```

REST architecture

- Client-server based, cacheable, stateless, ...
- It defines how a webapp should handle it's resources, and resource operations (the original goal was to create a method for the interoperability of webapplications)

- "Representative State Transfer"
- What does it mean in practice? We have a url, representing a resource on our server:
  - http://mylibrary.com/books/fiftyshades
  - On this resource we can do various operations, each of these will return a response containg the result of the operation (in an XML or JSON format)
  - These operations with http are: get, post, put, delete, ...
  - The official usage is that post creates a resource, put updates it, delete well … deletes it – but of course, these can be defined in the way we need it
  - With ExpressJS these are very easy to define:

ExpressJS - REST

```
router.route('/books/:title').get(function (req, res)
{ ... });
router.route('/books/:title').post(function (req, res)
{ ... });
router.route('/books/:title').put(function (req, res)
{ ... });
router.route('/books/:title').delete(function (req, res)
{ ... });
```

This is how complex it is to create REST endpoints to our resources...

Another question that we really don't want just any user to have complete power over our resources – we'll talk more about that next week...