Zaawansowane Programowanie Webowe

Lab 6. Node.js

Cel zajęć:

Poznanie środowiska Node.js wraz z frameworkiem Express oraz wykorzystanie ich do stworzenia aplikacji webowej typu klient-serwer.

Narzędzia:
Node.js (https://nodejs.org/) + menadżer pakietów NPM (do pobrania razem z Nodem - https://nodejs.org/en/download/) + edytor kodu
lub w przeglądarce https://codesandbox.io/

Node.js
Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine.[8]
(V8 is Google's open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js.[9])
Node.js is an open-source, cross-platform, JavaScript runtime environment that executes JavaScript code outside of a browser.
As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

Example code (to execute - save the code as the JS file e.g. `index.js` and run in console `node index.js`, then check `localhost:3000` in a browser):

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

In the following example, many connections can be handled concurrently.

[8] https://nodejs.org/en/about/
[9] https://v8.dev/

Node.js operates on a single-thread event loop, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching. The design of sharing a single thread among all the requests that use the observer pattern is intended for building highly concurrent applications, where any function performing I/O must use a callback. [10]

Node.js comes with a built-in modules. These modules provide methods to i.a.:

- handle the file system (fs),
- handle binary data (buffer),
- make Node.js act as an HTTP server (http),
- handle streaming data (stream).

https://www.w3schools.com/nodejs/ref_modules.asp

Usage:

```
var http = require('http');
http
  .createServer(function(req, res) {
    res.write("Hello World!"); //write a response to the client
    res.end(); //end the response
  })
  .listen(8080)
```

npm - Node Package Manager

npm is the package manager for Node.js. It was created in 2009 as an open source project to help JavaScript developers easily share packaged modules of code.

The npm Registry is a public collection of packages of open-source code for Node.js, front-end web apps, mobile apps, robots, routers, and countless other needs of the JavaScript community.

npm is the command line client that allows developers to install and publish those packages.
https://www.npmjs.com/

Another popular npm client is yarn - https://yarnpkg.com/lang/en/

Example package on npm:

- lodash - https://www.npmjs.com/package/lodash - JavaScript utility library that has methods for working with arrays, numbers, strings, objects etc.

Usual flow of creating new Node.js project:

1. `mkdir project`
2. `cd project`
3. `npm init`
    a. this command will create a package.json file

[10] https://en.wikipedia.org/wiki/Node.js

b. `package.json` holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies.
c. example `package.json`:

```
{
  "name": "node-sandbox",
  "version": "1.0.0",
  "description": "Simple Node Sandbox",
  "main": "index.js",
  "scripts": {},
  "dependencies": {},
  "devDependencies": {}
}
```

4. `touch index.js` - create file and fill it with JS code
5. `node index.js`
6. to install some npm package use: `npm install name-of-the-package`
   Installed package will be added to package.json to dependencies object.
   All the packages are installed into the `node_modules` directory. You should never add this directory to git repository. `package.json` and command `npm install` always allow to create and fill `node_modules` directory with needed for the project packages.

Using npm package - lodash example:
$ `npm install lodash`
In code:

```
const _ = require('lodash');
const object = { 'a': 1, 'b': '2', 'c': 3 };

_.omit(object, ['a', 'c']); // result will be { 'b': '2' }
```

Express

Express[11] is an open-source web application framework for Node.js. It is designed for building web applications and APIs.
`npm install express`
https://www.npmjs.com/package/express
Example:

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening on port ${port}!`))
```

Basic routing:

[11] http://expressjs.com/

Respond with Hello World! on the homepage:

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

Respond to POST request on the root route (/), the application's home page:

```
app.post('/', function (req, res) {
  res.send('Got a POST request')
})
```

Respond to a PUT request to the /user route:

```
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user')
})
```

Respond to a DELETE request to the /user route:

```
app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user')
})
```

More: http://expressjs.com/en/starter/hello-world.html

Zadanie

Stwórz aplikację webową prezentującą wyniki wyszukiwania z wykorzystaniem API GitHuba. Aplikacja ma składać się z serwera stworzonego w Node.js z użyciem Expressa oraz frontendu stworzonego z wykorzystaniem EJS (Embedded JavaScript Templates) https://ejs.co/

Serwer:

- plik server.js
- na adresie '/' serwuje wyrenderowany plik index.ejs
- zawiera endpoint do obsługi zapytania POST od klienta z formularza - wysyła zapytanie do API GitHuba z wpisaną przez użytkownika frazą np. https://api.github.com/search/users?q=tom
  Endpoint zwraca wyrenderowany szablon z wynikami wyszukiwania. W przypadku braku rezultatów - wyświetlamy klientowi informację o błędzie.
- Przy wysyłaniu response dla zapytania użyj następującej konstrukcji:
  - `res.render('index', {data: ######, error: ######})`

Frontend:

- plik index.ejs
- zawiera formularz z inputem do wpisania zapytania i przyciskiem Szukaj
- akcja formularz to POST do serwera

- szablon zawiera miejsce, w którym wyrenderowane zostaną zwrócone z API wyniki wyszukiwania (użyj pętli `for` do wypisania wszystkich rezultatów, każdy użytkownik powinien być wyrenderowany jako osobna tabela zawierająca wszystkie informacje zwrócone z API)

Ostyluj stronę używając osobnego pliku CSS.

[12pkt]