Lesson 6:
Node.js - introduction

Lesson goal

This lesson introduces the notion of Web application and the basics of server-side programming of a Web application using a programming framework. The notion of framework is closely related to the concept of design pattern. Design pattern describes a and names solution of a common problem in application construction. A framework then provides the language constructs, structure and readily available functionality allowing to implement a given pattern or even enforcing following it. Our main example of design pattern in Web application will be MVC (model-view-controller), which we are going to realize in the following lessons.

In the practical part of the lesson we are going to begin programming in Node.js . Although it is possible to use one of the cloud-based environment, you would probably prefer install the environment locally: https://nodejs.org/en/download/ Some Node.js aware IDE would be also advisable (for example – Visual Studio Code or JetBrains WebStorm).

In this module we are only going to make some initial steps in development of a simple HTTP server – for now without additional frameworks (like Express) or application structuring.

For that purpose consider the following template:

Poniżej pokazano szkielet implementacji prostego serwera HTTP, wykorzystującego do pobierania żądań i konstruowania odpowiedzi moduł `http`.

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

function serveRequest(req, res){

      console.log(req);

}

const server = http.createServer(serveRequest);

server.listen(3000, localhost);
```

It contains the loading of the http module, the creation of server process (including the callback function to serve incoming requests) and entering the listening mode on the port and address specified (localhost in this case).

In its current shape it only prints the content of the req object to show, what information the programmer can use when serving the request. You would probably need to eventually replace it with a response constructing code, where the following methods would be useful:

- `res.setHeader()`
- `res.write()`
- `res.end()`.

See https://nodejs.org/api/http.html to learn the details of that functionality.

Important details to investigate

- Any server-side dynamic webpage technology is essentially a mean of programmatically generating arbitrary HTTP responses based on the request received and the context of processing; note that this may mean the whole HTML document, but also the selected HTTP headers if needed.
- The overall result of dynamic website invocation is a combination of elements programmed explicitly by application developer, the static fragments and the elements generated by a common implementation of a given framework.
- The principles behind the MVC pattern (the short tutorial mentioned below allows to understand its elements: during gradual development of the example you encounter errors that indicate, what elements of the pattern already worked properly, and what is the next element in the chain of processing that needs to be implemented); the role of the routes definition;
- The concepts behind the PRG (Post-Redirect-Get) pattern
- The Node.js architecture and resulting the style of programming (understanding the Event Loop)
- Node.js environment (the global `object` and its content) and Node Core Modules (including fs and http)

External resources in English

- https://www.w3schools.com/nodejs/ - a simple introduction from W3Schools
- https://www.youtube.com/watch?v=RLtyhwFtXQA – a popular tutorial video on Node.js – a more in-depth; contains some material that will be useful for the subsequent lessons (NPM, Express, EJS, JSON…)

Assignment

Implement, using the Node.js, a simplistic web server that would accept (through appropriate URL segments or its query string part) three arguments:

- first operand (number)
- second operand (number)
- operation name (add, subtract, multiply, divide)

and produce a HTML output containing the result of that calculation.