UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
DEPARTAMENTO DE CIÊNCIAS DE COMPUTAÇÃO

SCC0219 - Introduction to Web Development

Exercises 13: Node.js

Upload a JavaScript file named USPnumber1.USPnumber2.js with your server implementation.

Attention: If you need to upload a file after the deadline, use late.USPnumber1.USPnumber2.js

1. Read and understand the code in the annex. It explains how to create a node.js web server without a framework.
2. Run the server. You may use node.js on your computer or an online node.js IDE, such as https://codesandbox.io. There is a codesandbox template. You can copy it by signing up (it is free) and using the Fork button. To run a new version, use the Fork button again.
3. Create a new, more modern version of the program that uses import (not require), await/async, let/const (not var), and arrow functions, with a Promise version of `fs.readFile`. For the import command to work, include `"type"="module"` in the `package.json` file.
4. Create a new HTML file, different from `index.html`, and serve it from the folder `example`.
5. Modify the server to create a new route, called `/random`, that returns HTML with a random number between 0 and 1.
6. Modify the server to create a new route, called `/random?max=3`, that returns HTML with a random number between 0 and the value of the `max` variable.

Annex: index.js

```
// We start importing the necessary modules for the server.
// The `http` module is a built-in Node.js module for creating
// networking capabilities like HTTP servers and clients.
var http = require("http");

// The `fs` (file system) module is a built-in Node.js module that
// provides an API to interact with the file system.
var fs = require("fs");

// The `path` module is a built-in Node.js module that provides
// utilities for working with file and directory paths.
// It can be used for extracting file extensions, joining paths,
// resolving paths, etc.
var path = require("path");

// `http.createServer()` is a method to create a new instance of an
// HTTP server.
// It takes a callback function, which is invoked every time the
// server receives a request.
http.createServer(function (request, response) {
    // We're logging the requested URL to the console.
    console.log("request ", request.url);

    // Preparing the file path for the requested resource.
    // The `.` at the start refers to the current directory, so we're
    // looking for files in the same directory as this script.
    var filePath = "." + request.url;
    if (filePath == "./") {
        // If the requested path is just `/`, we change the file path
        // to `./index.html`.
        // This is a common practice since `index.html` usually
serves
        // as the homepage in many web servers.
        filePath = "./index.html";
    }
```

```
// Extracting the extension from the requested file's path.
// We're converting it to lowercase to avoid case-sensitivity
// issues.
var extname = String(path.extname(filePath)).toLowerCase();

// This is a lookup object for MIME types based on file
// extensions.
// A MIME type is a standard that indicates the nature and format
// of a document, file, or assortment of bytes.
var mimeTypes = {
    ".html": "text/html",
    ".js": "text/javascript",
    ".css": "text/css",
    ".json": "application/json",
    ".png": "image/png",
    ".jpg": "image/jpg",
    ".gif": "image/gif",
    ".svg": "image/svg+xml",
    ".wav": "audio/wav",
    ".mp4": "video/mp4",
    ".woff": "application/font-woff",
    ".ttf": "application/font-ttf",
    ".eot": "application/vnd.ms-fontobject",
    ".otf": "application/font-otf",
    ".wasm": "application/wasm"
};

// Finding the correct MIME type for the requested file.
// If the extension is not in our lookup table, we default to
// "application/octet-stream".
var contentType = mimeTypes[extname]||"application/octet-stream";

// Reading the requested file from disk using the `fs.readFile`
// function.
// This is an asynchronous operation, so we provide a callback
// function that will be executed once the file is read.
fs.readFile(filePath, function (error, content) {
    // If there's an error while reading the file:
    if (error) {
```

```
            // If the error is "ENOENT" (Error NO ENTry), it means
the
            // file was not found.
            if (error.code == "ENOENT") {
                // In that case, we read a custom 404 error page.
                fs.readFile("./404.html", function (error, content) {
                    // We set the response status code to 404 and the
                    // content type to "text/html".
                    response.writeHead(404,
                                       {"Content-Type": "text/html"
});
                    // We then end the response, sending the content
                    // of our 404 page.
                    response.end(content, "utf-8");
                });
            } else {
                // For any other error, we set the response status
                // code to 500 (Internal Server Error).
                response.writeHead(500);
                // We then end the response, sending an error
message.
                response.end(
                    "Sorry, check with the site admin for error: " +
                    error.code +
                    " ..\n"
                );
            }
        } else {
            // If the file was successfully read (there's no error),
            // we set the response status code to 200 (OK) and the
            // content type correctly.
            response.writeHead(200, { "Content-Type": contentType });
            // We then end the response, sending the content of the
            // file we just read.
            response.end(content, "utf-8");
        }
    });
})
// We then make our server listen for incoming requests.
// The number 8125 is the port number. This could be any number
// between 1024 and 49151, but 8125 is chosen arbitrarily here.
```

```
.listen(8125);

// Finally, we log to the console that the server is running and
// listening for requests.
// The IP address 127.0.0.1 is the loopback address, which means the
// host computer itself.
console.log("Server running at http://127.0.0.1:8125/");
```

Annex: index.html

```
<html>
    <body>
        <h1>Hello</h1>
    </body>
</html>
```

Good Work!