Lesson 7:
Node.js - continued

Lesson goal

This lesson continues the Node.js topic covering the Express framework fundamentals, asynchronous programming of I/O operations and building views with EJS templates.

The topics directly addressed include:

- Building a redirection (status code 302) response;
- Consuming form data using raw Node.js functionality:

```
const body = [];         // tablica dla napływających danych

req.on('data', (chunk) => {
  body.push(chunk);  // zasilenie tablicy
});

req.on('end', () => {

   const parsedBodfy = Buffer.concat(body).toString();

   const message = parsedBody.split('=')[1];

});
```

- Writing file data asynchronously and constructing response in a callback:

```
fs.writeFile('message.txt', dane, err => {

   res.statusCode = 302;

   res.setHeader('Location', '/');

   return res.end();

});
```

- Distinguishing Node.js Core Modules, local code modules and third party modules installed with NPM;
- Partitioning application by using Express Router;
- Constructing a simple code 404 response;
- Serving static HTML pages and static resources (e.g. CSS);
- Using the **path** module to handle resources in local file system;
- Retrieving request POST data using the body-parser package;
- Using EJS templates and passing data items to them;
- Using nodemon to streamline the development (no manual restarts needed).

Important details to investigate

- Synchronous vs. asynchronous variants of file handling methods (esp. writeFile())
- The notion of *middleware* in Express framework
- Chunk encoding in HTTP data transmission

- The concept of server-side templates – their specific features and benefits
- The historical distinction between CGI-family and SSI-family styles of server-side programming
- The interlacing of pure HTML and JS includes in EJS templates; including static content with the if() { … } clause.

External resources in English

- (see previous module)

Assignment

1. Create (without using Express) a simple functionality consisting of a single-field form (using POST method) and a target page that would consume the submitted data by showing it in the HTML response
2. Create a modification of the above, which instead of printing the submitted data stores it in a file and creates a redirection response pointing to the form page
3. Build a page with a form consisting of several fields. Implement a target page for it that would construct a view using EJS including properly arranged data sent from that form. The page should use a simple external CSS.