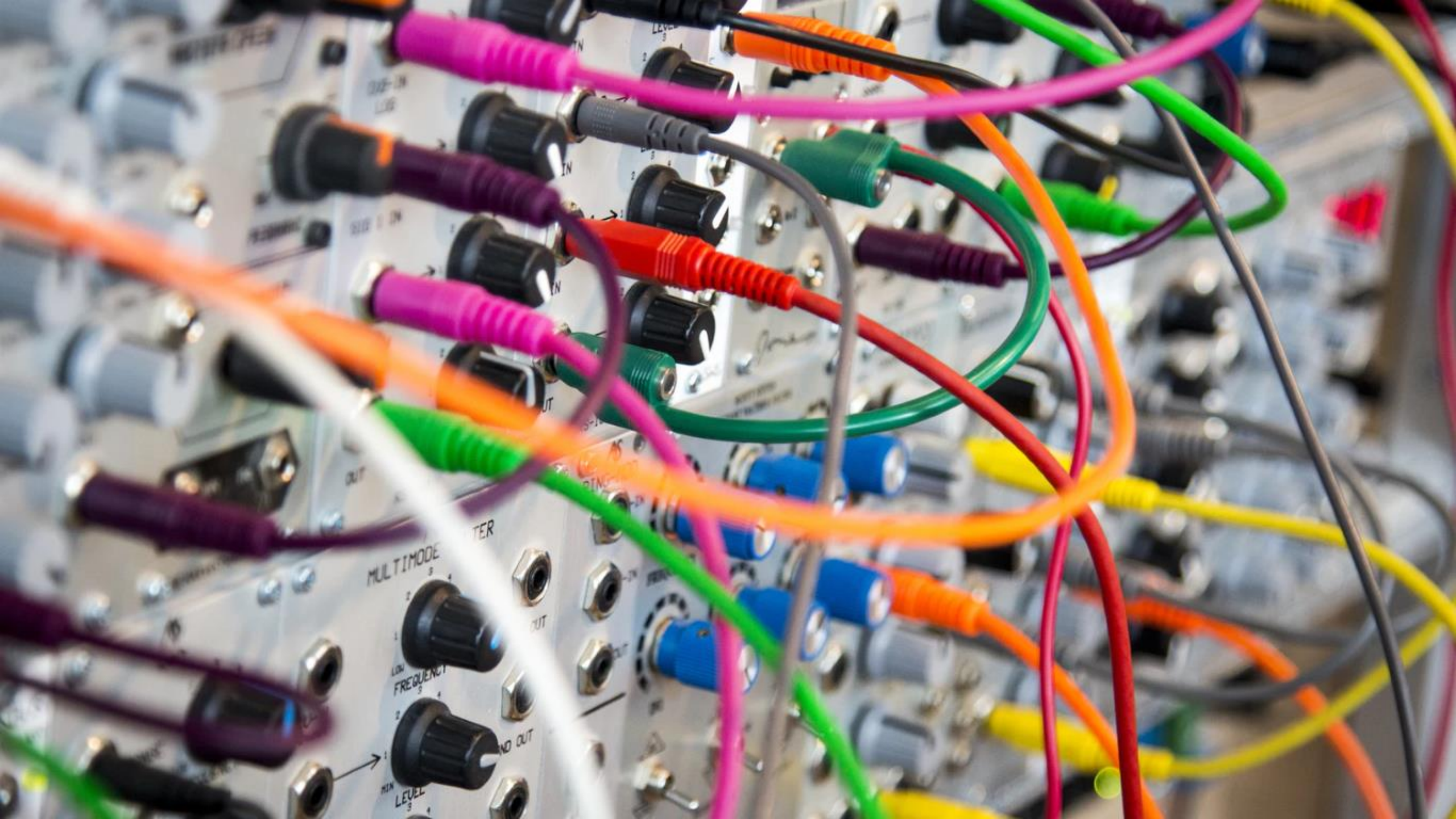
NodeJS

środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

Rok akademicki: 2020/2021, Semestr: 2, Wykład: 6


MULTIMODE
FREQUENCY
LEVEL


0123
4567
89/-

Template engine

Express provides a mechanism that allows you to use static template files in your application.

During program execution, the template engine replaces the variables in the static file with the actual values and transforms the template into an HTML file that is sent to the user.

Template engine

Most popular template engines that work with Express are:

- Jade/Pug
- Mustache (mustache-express)

List of supported template engines:

https://expressjs.com/en/resources/template-engines.html
(some mentioned projects seems to be abandoned)

Render

In order to render the template files it is necessary to:

- install package with template engine
- set the path to the directory where the templates will be located
  e.g.: app.set('views', ,'./views’);
- set the appropriate template engine
  e.g.: app.set('view engine', 'pug')

Jade/Pug

Pug is a high-performance template engine heavily influenced by Haml and implemented with JavaScript for Node.js and browsers.

https://pugjs.org/api/getting-started.html

https://github.com/pugjs/pug

Jade/Pug

```
// npm install pug
const express = require('express');
const app = express();

app.set('view engine', 'pug');
app.set('views', './views');

app.get('/', function (req, res) {
    const scope = { title: 'some title', header: 'heloo!' };
    res.render('index', scope);
});
//...
```

Jade/Pug template

Content of index.pug file:

```
html
        head
                title= title
        body
                h1= header
```

Mustache (mustache-express)

Mustache is a web template system with implementations available for many programming languages.

Mustache is described as a "logic-less" system because it lacks any explicit control flow statements, like if and else conditionals or for loops.

https://mustache.github.io

https://github.com/bryanburgers/node-mustache-express

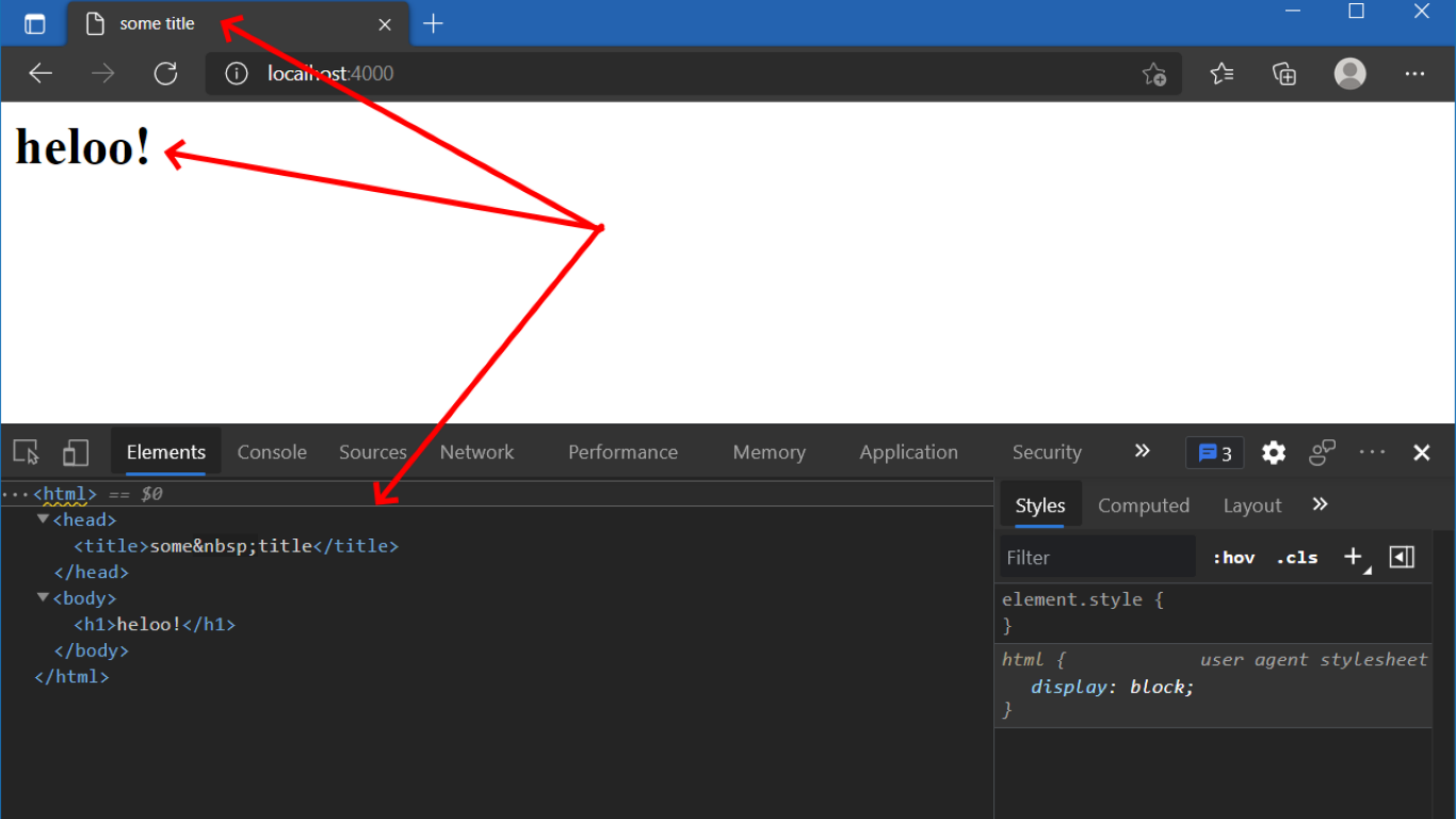
Mustache (mustache-express)

```
// npm install mustache-express

const express = require('express');
const mustacheExpress = require('mustache-express');
const app = express();

app.engine('mustache', mustacheExpress());
app.set('view engine', 'mustache’);
app.set('views', './views');

app.get('/', function (req, res) {
    const scope = { title: 'some title', header: 'heloo!' };
    res.render('index', scope);
});
```

Mustache (mustache-express) template

Content of index.mustache file:

```
<html>
    <head>
        <title>{{ title }}</title>
    </head>
    <body>
        <h1>{{ header }}</h1>
    </body>
</html>
```


some title
localhost:4000
heloo!
Elements
Console
Sources
Network
Performance
Memory
Application
Security
<html> == $0
<head>
<title>some title</title>
</head>
<body>
<h1>heloo!</h1>
</body>
</html>
Styles
Computed
Layout
Filter
:hov .cls
element.style {
}
html { user agent stylesheet
display: block;
}

mongoose

Mongoose

Mongoose is a MongoDB object modeling tool designed to work in an asynchronous environment.

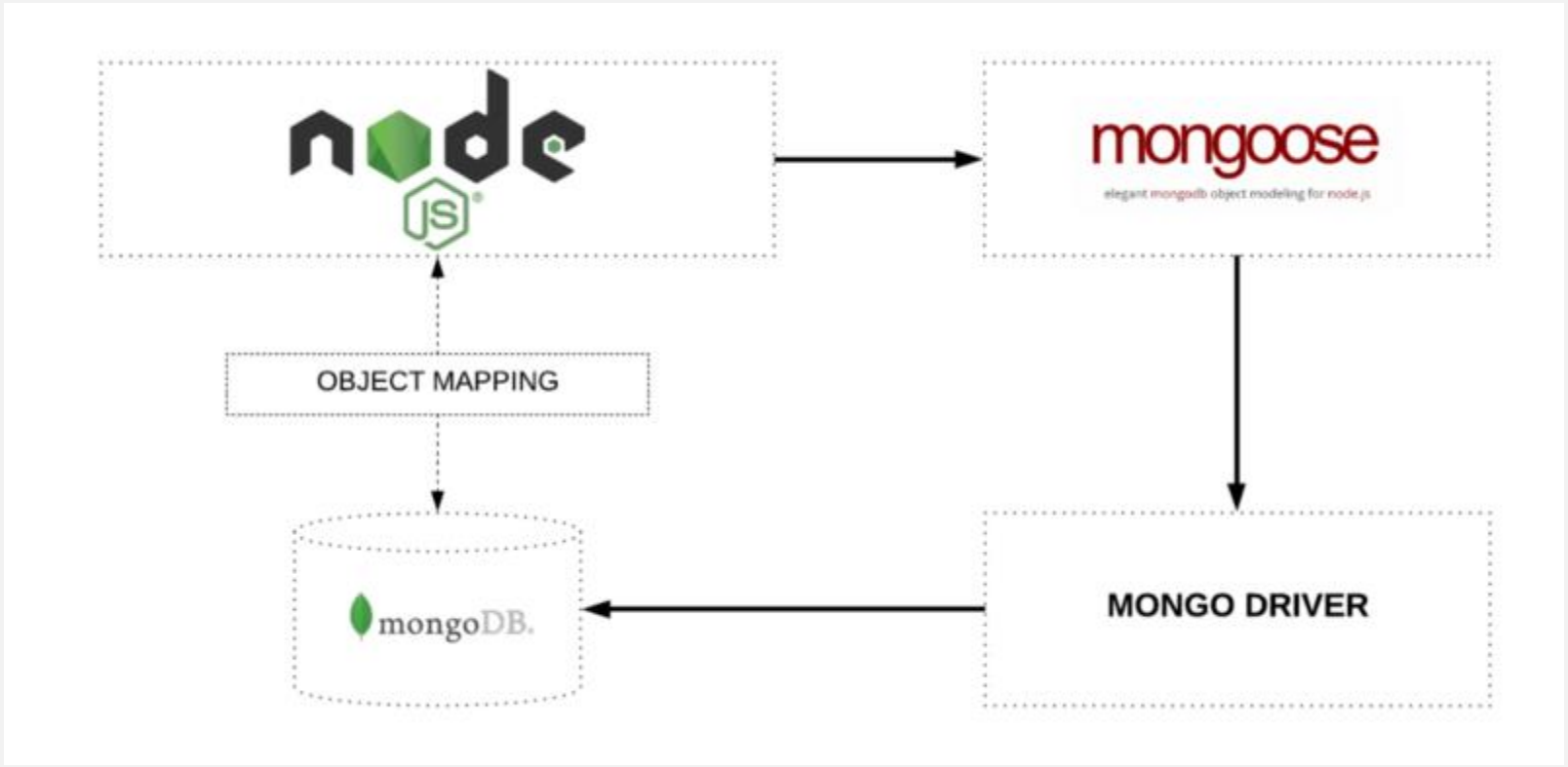
It manages relationships between data, provides schema validation, and is used to translate between objects in code and the representation of those objects in MongoDB.

https://mongoosejs.com

https://github.com/Automattic/mongoose

Mongoose - sample

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

mongoose.connect('connection_string', { useNewUrlParser: true, useUnifiedTopology: true });

const Task = mongoose.model('tasks', {
    name: String,
    completed: type: Boolean
});

(async () => {
    const task = new Task({
        name: 'kupić czekolade',
        completed: false,
    });

    await task.save();
})();
```


node
JS
mongoose
elegant mongodb object modeling for node.js
OBJECT MAPPING
mongoDB.
MONGO DRIVER

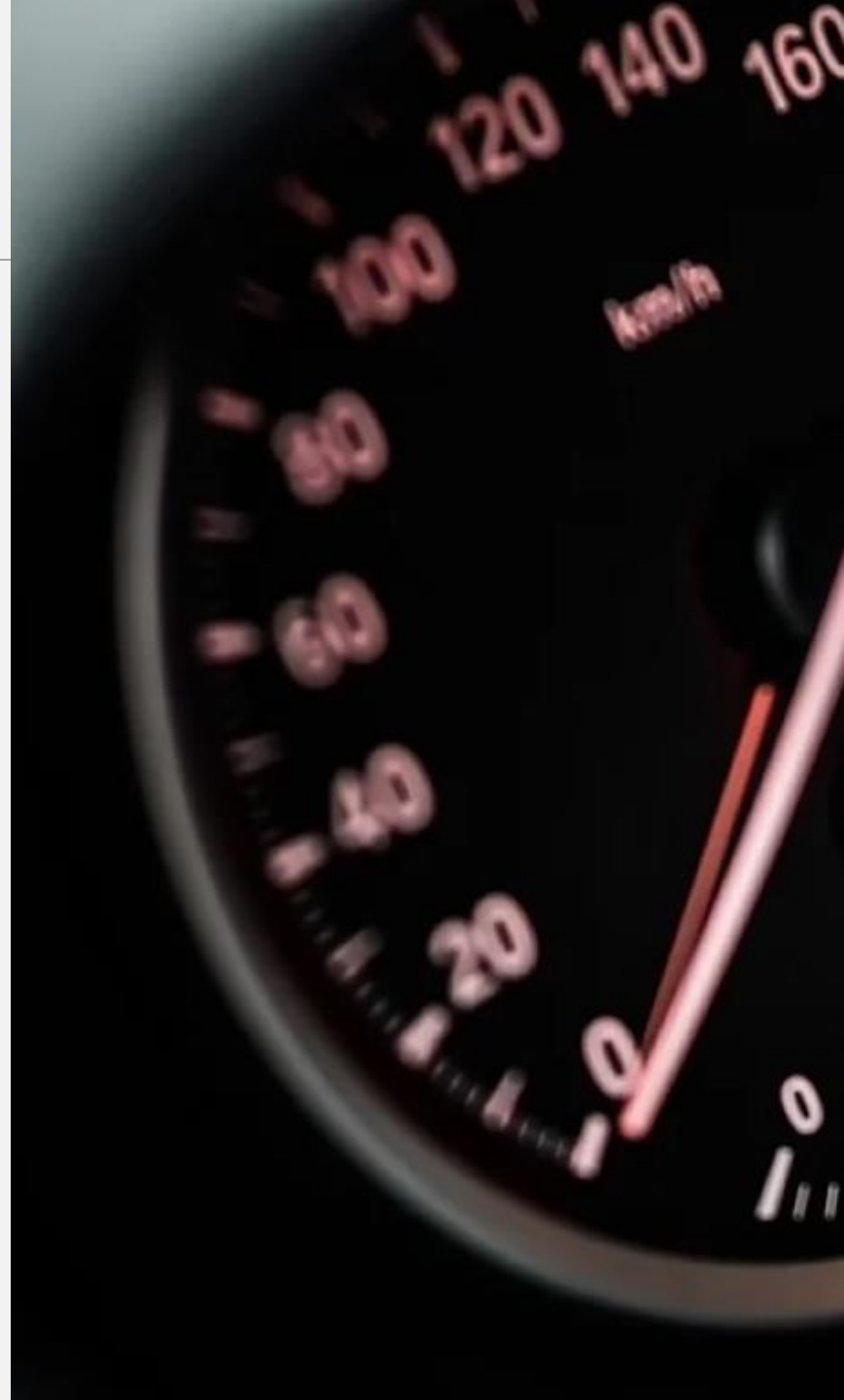
Mongoose features

- Validators (async and sync)
- Defaults
- Getters
- Setters
- Indexes
- Middleware
- Methods definition
- Statics definition
- Plugins
- pseudo-JOINs

Mongoose vs native driver

READS	Native	Mongoose
Throughput	1200 #/sec	583 #/sec
Avg Request	0.83 ms	1.71 ms
WRITES	**Native**	**Mongoose**
Throughput	1128 #/sec	384 #/sec
Avg Request	0.89 ms	2.60 ms

https://blog.jscrambler.com/mongodb-native-driver-vs-mongoose-performance-benchmarks/



Testing API with POSTMAN

Collection runs:

https://learning.postman.com/docs/running-collections/intro-to-collection-runs/

Test scripting:

https://learning.postman.com/docs/writing-scripts/script-references/test-examples/

POSTMAN

Test function

```
pm.test("test case name", function () {
    // test assert
});
```

POSTMAN

Assertions

```
// pm.expect
pm.test("Status code is 200", function () {
    pm.expect(pm.response.code).to.eql(200);
});

// pm.response
pm.test("Status code is 200", function(){
    pm.response.to.have.status(200);
});
```

POSTMAN

NODE.JS
BEST PRACTICES

https://github.com/goldbergyoni/nodebestpractices