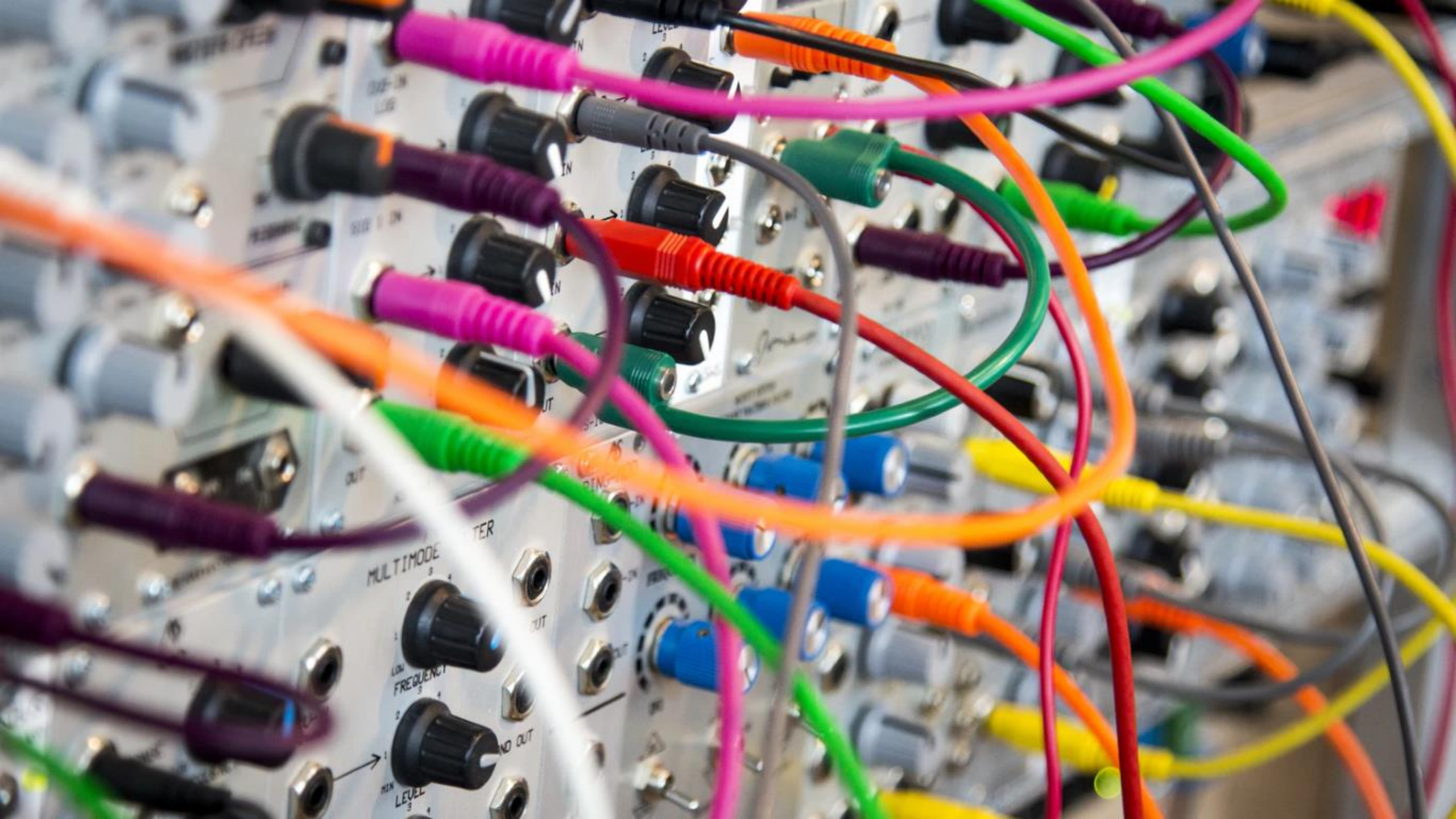
NodeJS

środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

Rok akademicki: 2023/2024, Semestr: 2, Wykład: 6


MULTIMODE
FREQUENCY
LEVEL

mongoose

OBJECT → HTML

```
let title = //...
let h1 = //...
let section = //...

let page =
`<html>
<head>
    <title>TITLE_HERE</title>
</head>
<body>
    <h1>H1_HERE</h1>
    <section>SECTION_HERE</section>
</body>
</html>`;
```

```
page = page.replace('TITLE_HERE', title);
page = page.replace('H1_HERE', h1);
page = page.replace('SECTION_HERE', section);

res.contentType = res.type("text/html");
res.send(page);
```

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’)
  ```

  Or

  ```
  app.engine('mustache', mustacheExpress());
  app.set('view engine', 'mustache');
  ```
- return page using render function
  e.g.:

  ```
  res.render('index', scope);
  // 'index’ - name of template file
  // scope - object with data to render on page
  ```

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
```

```
// content of scope
    const scope = {
        title: 'some title’,
        header: 'heloo!’
    };
```

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)

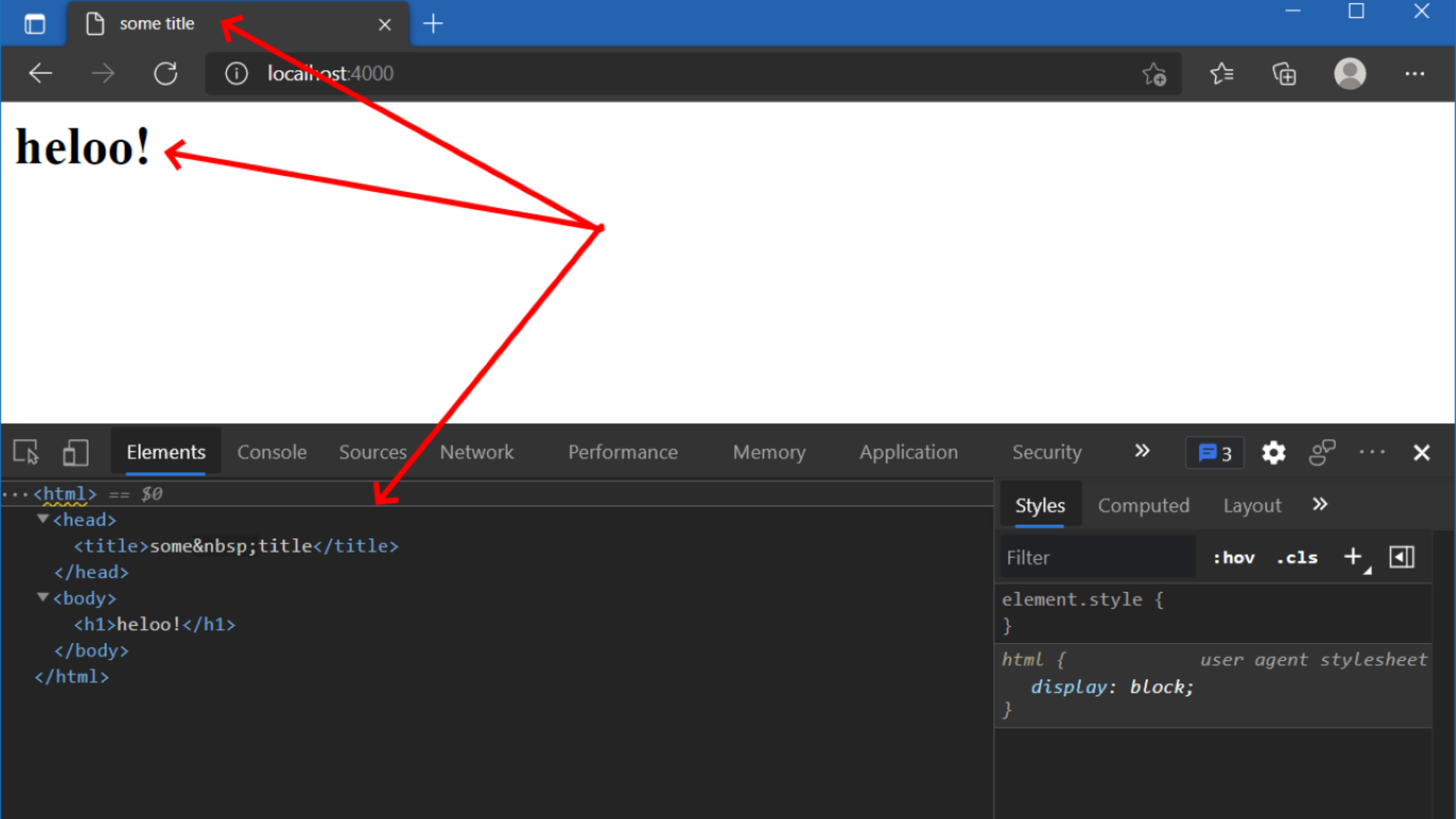
```
// npm install mustache-express

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

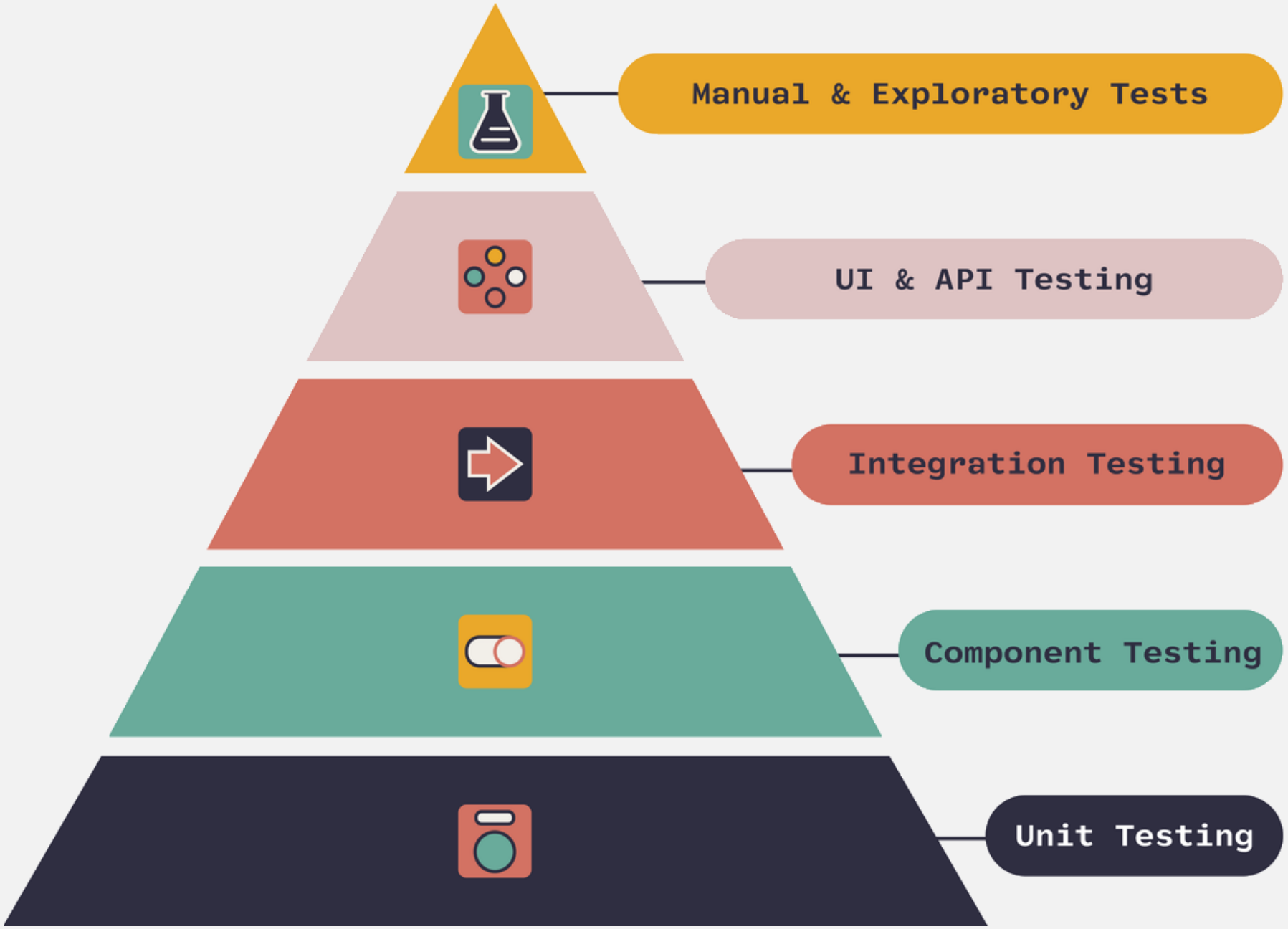
Content of index.mustache file:

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

```
// content of scope

    const scope = {
        title: 'some title’,
        header: 'heloo!’
    };
```


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;
}

Testing is integral part of software development


Manual & Exploratory Tests
UI & API Testing
Integration Testing
Component Testing
Unit Testing

Testing API with POSTMAN

You can use Tests tab in your requests, folders, and collections to write tests that will execute when Postman receives a response from the API you sent the request to.

Add however many tests you need for each request. When you add tests to a folder or Collection, they will execute after each request inside it.

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

POSTMAN

Test function – basics

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

POSTMAN

Assertions

```
// two possible syntax


// 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

Assertions – response body

```
// convert json response into object
const response = pm.response.json();

// response testing
pm.test("Person is Jane", () => {
    const response = pm.response.json();
    pm.expect(response.name).to.eql("Jane");
    pm.expect(response.age).to.eql(23);
  });
```

POSTMAN

Assertions – response metadata

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

// check content-type header
pm.test("Content-Type header is text/plain", () => {
  pm.expect(pm.response.headers.get('Content-Type')).to.eql('text/plain');
});

// check response time
pm.test("Response time is less than 200ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(200);
});
```

POSTMAN

Assertions – custom function

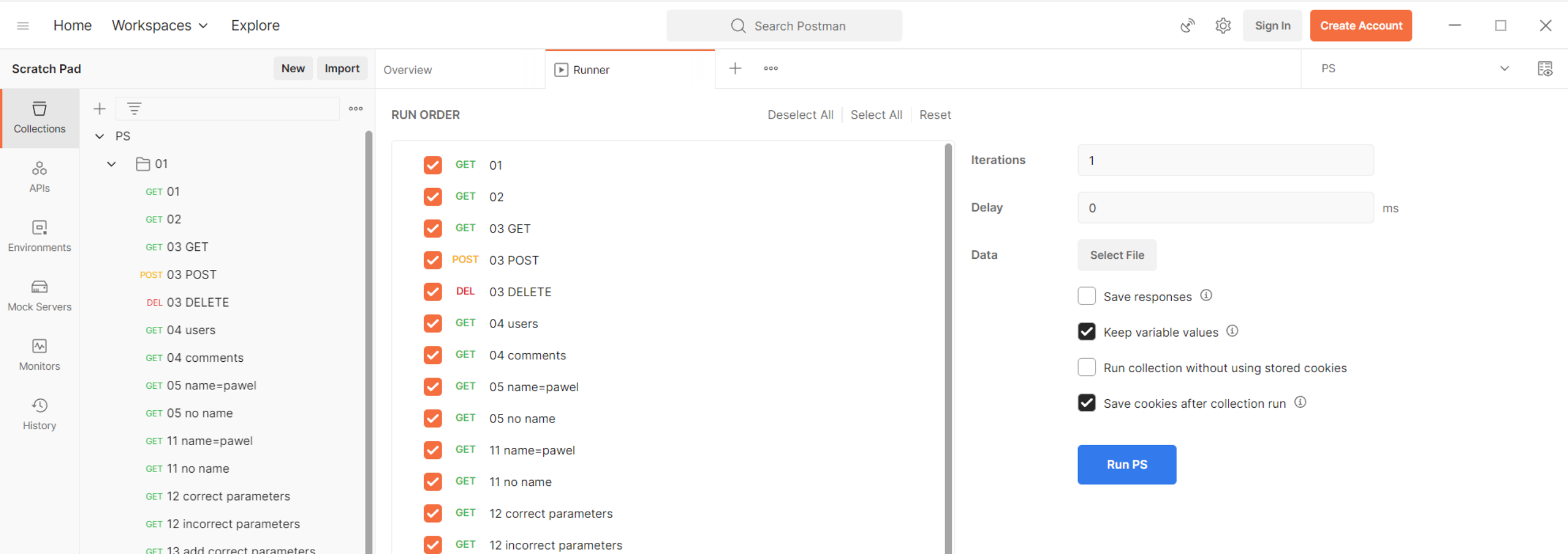
```
// we can call custom functions and evaluate result
function MyCustomFunction(){
    if (...) {
        return true;
    }
    else {
        return false;
    }
}

pm.test("Custom function check", function () {
    pm.expect(MyCustomFunction()).to.be.true;
});
```

POSTMAN

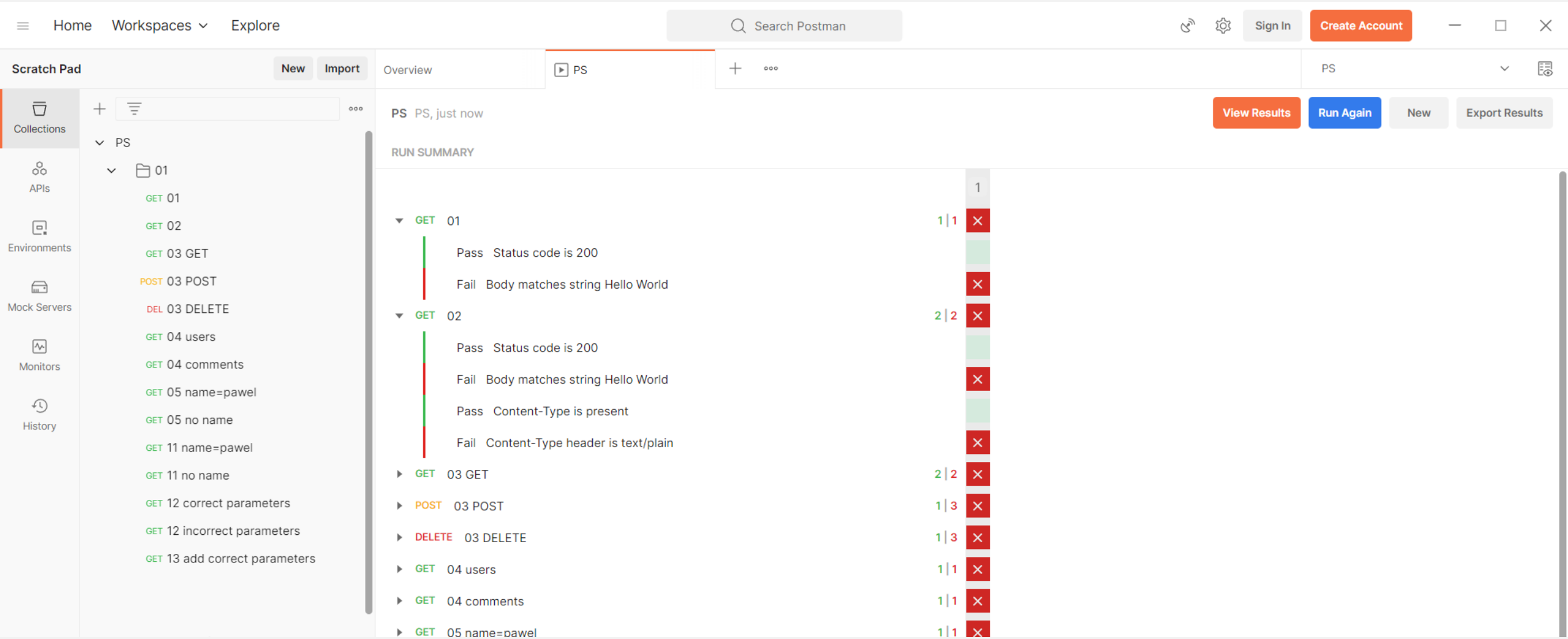
Collection run

Collection run using Postman GUI:

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



Collection run - results



Collection run with CLI - Newman

Newman is a command-line Collection Runner for Postman.

It enables you to run and test a Postman Collection directly from the command line.

You can integrate it with your continuous integration servers and build systems.

https://learning.postman.com/docs/collections/using-newman-cli/command-line-integration-with-newman/

POSTMAN

Collection run with CLI - Newman

Install Newman **globally** on your system, which allows you to run it from anywhere:

> npm install -g newman

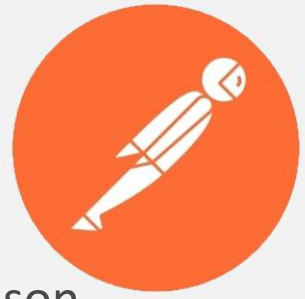
Then export your collection and (environment if necessary).

Run Newman test without environment:

> newman run file_with_postman_collection.json

Run Newman test with environment:

> newman run file_with_postman_collection.json -e file_with_postman_environment.json



POSTMAN

→ get user
GET https://jsonplaceholder.typicode.com/users/{{userId}} [404 Not Found, 971B, 117ms]
1. status is 200
✓ reponse is json
✓ response time is less than 1000ms
2. retrieved user has same Id as Id from enviroment variables
3. geo data is set
4. TypeError in test-script

	executed	failed
iterations	1	0
requests	1	0
test-scripts	1	1
prerequest-scripts	0	0
assertions	5	3
total run duration: 208ms		
total data received: 2B (approx)		
average response time: 117ms [min: 117ms, max: 117ms, s.d.: 0µs]		

#	failure	detail
1.	AssertionError	status is 200 expected response to have status code 200 but got 404 at assertion:0 in test-script inside "get user"
2.	AssertionError	retrieved user has same Id as Id from enviroment variables expected undefined to deeply equal NaN at assertion:3 in test-script inside "get user"
3.	AssertionError	geo data is set expected undefined to be an object at assertion:4 in test-script inside "get user"
4.	TypeError	Cannot read properties of undefined (reading 'geo') at test-script inside "get user"

POSTMAN

Node:

https://nodeweekly.com
https://github.com/goldbergyoni/nodebestpractices
https://github.com/lirantal/nodejs-cli-apps-best-practices
https://gist.github.com/paulfranco/9f88a2879b7b7d88de5d1921aef2093b

Job interviews:

https://github.com/a8hok/NodeJS-Interview
https://github.com/lydiahallie/javascript-questions