# Środowisko uruchomieniowe 'Node.js'

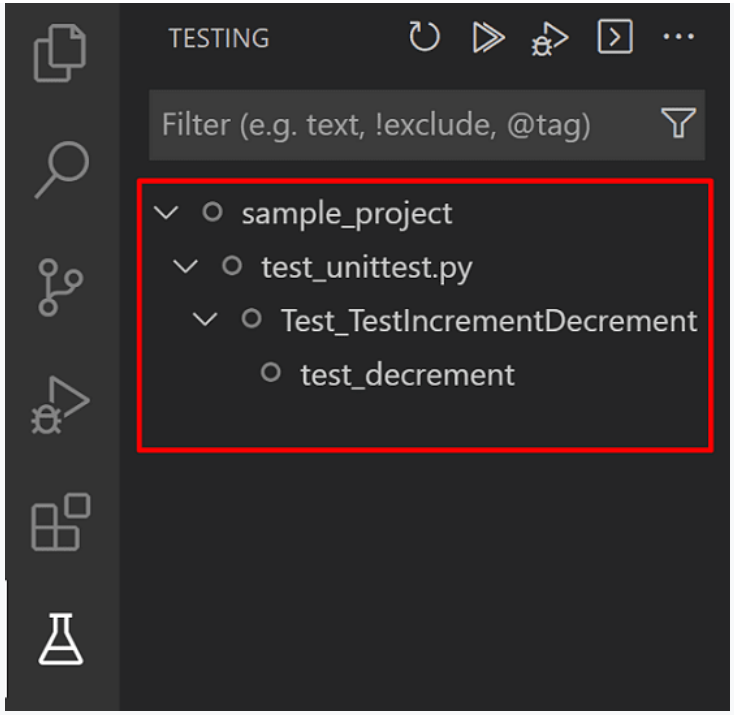
## 1. Moduły typu 'CommonJS' oraz 'ES6'

1. Skonfiguruj edytor *Visual Studio Code*
2. Utwórz, w katalogu 'cw4', skrypt (główny) o nazwie 'index.js' zawierający klasę *Operation*. Składniki klasy:
   - Dwie własności: *x* oraz *y*
   - Dwuargumentowy konstruktor przypisujący ww. własnościom dane początkowe
   - Metoda `sum()`, która oblicza, a następnie zwraca x+y
3. Umieść w ww. skrypcie fragment kodu wypisujący wynik sumowania liczb dla ustalonych (przez Ciebie) wartości *x*, *y*
4. Uruchom skrypt za pomocą komendy `node index`, `node .` lub naciśnij `Ctrl+Alt+R`
5. Sprawdź, czy jest możliwe utworzenie obiektu przed definicją klasy
6. Przenieś definicję klasy do osobnego pliku o nazwie 'module.js' — w skrypcie głównym ma zostać tylko kod wypisujący wynik sumowania liczb
7. W pierwszej linii pliku 'index.js' umieść `const module = require('./module');`, a następnie spowoduj, aby skrypt nadal działał, tzn. po wykonaniu komendy `node index` wypisywał wynik operacji x+y
8. Stworzony (powyżej) moduł to moduł typu "CommonJS". Skonwertuj go (ręcznie) do modułu typu "ES6", a następnie użyj tego modułu w skrypcie głównym — być może będzie to wymagać zmiany rozszerzenia nazwy pliku
9. Przeczytaj fragment artykułu poświęcony poleceniu `npx`
10. Utwórz podkatalog 'cw4/test'
11. Utwórz w nim plik o nazwie 'test1.js' i następującej zawartości:

```
/*
  Mocha allows you to use any assertion library you wish. In this example, we are using the built-in module called 'Assert'.
  If you prefer the 'Chai' library (https://www.chaijs.com/) then you have to install it yourself: 'npm install chai --save-dev',
  and then you need to uncomment the lines below.
*/

//---------------------------------------
// Mocha tests with CommonJS style imports
//---------------------------------------

// var expect = require('chai').expect;
var assert = require('assert');
var module = require('../module');

describe('The sum() method', function () {
  it('Returns 4 for 2+2', function () {
    var op = new module.Operation(2, 2);
    assert.strictEqual(op.sum(), 4)
    // expect(op.sum()).to.equal(4);
  });
  it('Returns 0 for -2+2', function () {
    var op = new module.Operation(-2, 2);
    assert.strictEqual(op.sum(), 0)
```

```
    // expect(op.sum()).to.equal(0);
  });
});

//-----------------------------------
// Mocha tests with ES6 style imports
//-----------------------------------

/*
- You must install the 'esm' module (https://www.npmjs.com/package/esm) — npm install esm --save-dev
- You must run tests as follows: npx mocha --require esm
Source: https://stackoverflow.com/questions/57004631/mocha-tests-with-es6-style-imports

import { Operation } from "../module";
import assert from 'assert';

describe('The sum() method', function () {
  it('Returns 4 for 2+2', function () {
    var op = new Operation(2, 2);
    assert.strictEqual(op.sum(), 4)
  });
  it('Returns 0 for -2+2', function () {
    var op = new Operation(-2, 2);
    assert.strictEqual(op.sum(), 0)
  });
});
*/
```

12. Zainstaluj wtyczkę "ES6 Mocha Snippets" dla *Visual Studio Code* — wykonaj komendę: `code --install-extension spoonscen.es6-mocha-snippets`
13. Obejrzyj jej demo, a następnie wypróbuj niektóre z oferowanych, przez nią, snippetów
14. Z poziomu katalogu 'cw4', uruchom test za pomocą komendy `npx mocha`
    - Powyższa komenda spowoduje zainstalowanie (w podkatalogu wymienionym w "How can I clear the central cache for `npx`?"), a następnie uruchomienie programu *mocha*
    - Jeżeli komenda `npx` nie jest dostępna, to:
      1. Zainstaluj (samodzielnie) framework Mocha: `npm install mocha --save-dev` — w tym przypadku program *mocha* zostanie zainstalowany w podkatalogu './node_modules/mocha/bin'
      2. Uruchom test za pomocą komendy `node ./node_modules/mocha/bin/mocha`
15. Zainstaluj wtyczkę "Mocha sidebar" dla *Visual Studio Code* — wykonaj komendę: `code --install-extension maty.vscode-mocha-sidebar`




16. Obejrzyj jej demo, a następnie uruchom, za jej pomocą, test z poziomu *Visual Studio Code* Przykład uruchomienia testu dla języka Python
17. Zmodyfikuj skrypt 'index.js' — wartości parametrów *x* oraz *y* mają być przekazywane z poziomu linii komend — `node index 2 7`
18. Wzbogać kod źródłowy modułu (klasa *Operation*, metoda `sum()` oraz konstruktor) o komentarze dokumentacyjne programu JSDoc
19. Wygeneruj dokumentację korzystając z komendy `npx jsdoc module.js`

# 2. Moduł 'fs'

1. 
    1. Napisz skrypt, który korzystając z modułu fs, dla podanego w linii komend napisu, wyświetla informację czy reprezentuje on nazwę (istniejącego) pliku, czy katalogu. Jeżeli jest to plik, to korzystając z funkcji w wersji **synchronicznej** (`fs.*Sync(...)`), ma on wypisać jego zawartość

        ```
        $ ./script.js plik.txt
        'plik.txt' jest plikiem, a jego zawartością jest:
              Ala ma kota
        $ ./script.js /etc
        '/etc' jest katalogiem
        ```

    2. Utwórz plik 'test/test2.js' zawierający test weryfikujący poprawność działania tej implementacji

# 3. Moduł 'http'

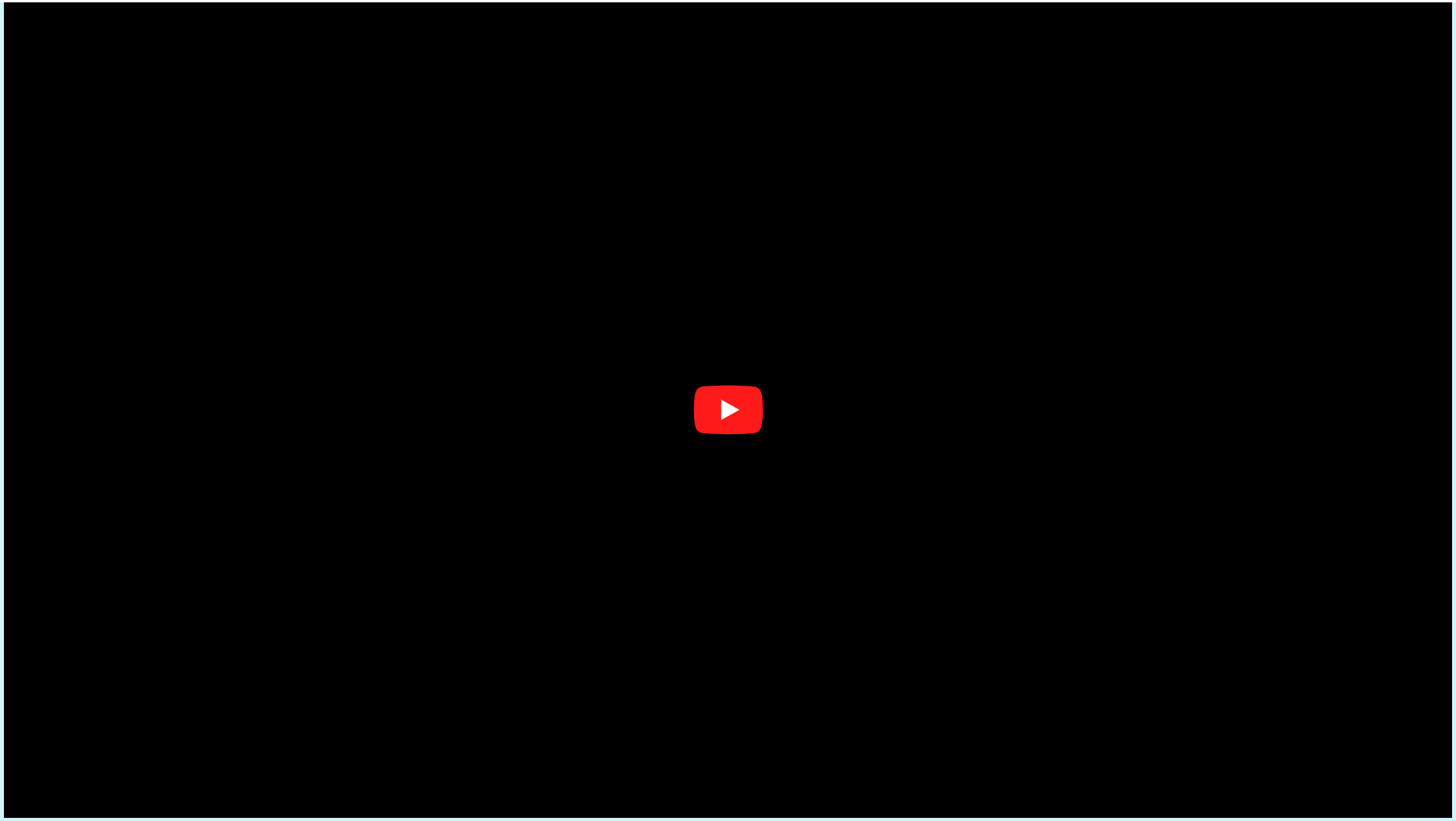
1. Utwórz, w katalogu 'cw4', plik 'server.js' o następującej zawartości:

    ```
    /**
           * Handles incoming requests.
    ```

```
        *
        * @param {IncomingMessage} request - Input stream – contains data received from the browser, e.g. encoded contents of HTML form fields.
        * @param {ServerResponse} response - Output stream – put in it data that you want to send back to the browser.
        * The answer sent by this stream must consist of two parts: the header and the body.
        * <ul>
        *  <li>The header contains, among others, information about the type (MIME) of data contained in the body.
        *  <li>The body contains the correct data, e.g. a form definition.
        * </ul>
       */
function requestListener(request, response) {
       console.log("-------------------------------------");
       console.log("The relative URL of the current request: " + request.url + "\n");
       var url = new URL(request.url, `http://${request.headers.host}`); // Create the URL object
       if (url.pathname == '/submit') { // Processing the form content, if the relative URL is '/submit'
              /* ************************************************ */
              console.log("Creating a response header");
              // Creating an answer header – we inform the browser that the body of the answer will be plain text
              response.writeHead(200, { "Content-Type": "text/plain; charset=utf-8" });
              /* ************************************************ */
              console.log("Creating a response body");
              if (request.method == 'GET') // If the GET method was used to send data to the server
                     // Place given data (here: 'Hello <name>') in the body of the answer
                     response.write(`Hello ${url.searchParams.get('name')}`); // "url.searchParams.get('name')" contains the contents of the field (form) named 'name'
              else // If other method was used to send data to the server
                     response.write(`This application does not support the ${request.method} method`);
              /* ************************************************ */
              console.log("Sending the response");
              response.end(); // The end of the response – send it to the browser
       }
       else { // Generating the form
              /* ************************************************ */
              console.log("Creating a response header")
              // Creating a response header – we inform the browser that the body of the response will be HTML text
              response.writeHead(200, { "Content-Type": "text/html; charset=utf-8" });
              /* ************************************************ */
              console.log("Creating a response body");
              // and now we put an HTML form in the body of the answer
              response.write(`<form method="GET" action="/submit">
                                   <label for="name">Give your name</label>
                                   <input name="name">
                                   <br>
                                   <input type="submit">
                                   <input type="reset">
                            </form>`);
              /* ************************************************ */
              console.log("Sending the response");
              response.end();  // The end of the response – send it to the browser
       }
}

/* ********************************************** */
```

```
/* Main block
/* ********************************************** */
var http = require("http");

var server = http.createServer(requestListener); // The 'requestListener' function is defined above
server.listen(8000);
console.log("The server was started on port 8000");
console.log("To stop the server, press 'CTRL + C'");
```

2. Uruchom skrypt za pomocą komendy `node server`
    - Jeżeli chcesz, aby po zmodyfikowaniu zawartości pliku źródłowego, aplikacja samodzielnie się restartowała, to uruchom ją następująco: `npx nodemon server`
    - W powyższym przykładzie dane (z formularza) są przekazywane metodą GET (`<form method="GET" ...>`). W przypadku przesyłania danych poufnych należy użyć POST — patrz GET a POST. Aby powyższy skrypt obsługiwał tę metodę przesyłu, należy go uzupełnić o dodatkowy fragment kodu
    - W naszym przykładowym kodzie, do wypisywania komunikatów diagnostycznych użyto `console.log()` — lepszym rozwiązaniem jest użycie modułu debug



3. Wpisz w przeglądarce adres http://localhost:8000/, podaj, w formularzu HTML, swoje imię i obejrzyj wyniki generowane przez skrypt
4. Sprawdź zachowanie się skryptu dla następujących adresów:
   - http://localhost:8000/submit?name=Jan
   - http://localhost:8000/submit?nazwisko=Kowalski
   - http://localhost:8000/dowolnyTekst

   Zamiast adresu symbolicznego `localhost`, np. http://localhost:8000/, możesz użyć adresu <nazwa>.lvh.me (przykład http://www.lvh.me:8000/)

5. Zainstaluj pakiet SuperTest — `npm install supertest --save-dev`

6. Utwórz plik 'test/test3.js' o następującej zawartości:

```
//Source:  https://codeforgeek.com/unit-testing-nodejs-application-using-mocha/
var supertest = require("supertest");

// This agent refers to PORT where program is runninng.
var server = supertest.agent("http://localhost:8000");

// UNIT test begin
describe('GET /submit?name=John', function () {
      it('respond with "Hello John"', function (done) {
            server
                  .get('/submit?name=John')
                  .expect('Content-Type', /text\/plain/)
                  .expect(200, "Hello John", done);
      });
});
```

7. Uruchom test. **Uwaga:** aplikacja musi już działać — `node server`
8. Przeczytaj artykuł poświęcony narzędziu ZAP
9. Zainstaluj ten program
10. Utwórz zmodyfikowaną wersję skryptu z sekcji "Moduł 'fs'":
    - Skrypt używa funkcji modułu fs, ale **tylko i wyłącznie**, w wersji **asynchronicznej** (~~`fs.*Sync(...)`~~)

      > Proszę korzystać z API używającego funkcji zwrotnych (ang. Callback), a nie obietnic — te ostatnie będą omawiane, dopiero, na jednym z kolejnych wykładów

    - Dane są przekazywane z formularza HTML, a nie z linii komend — zamiast aplikacji konsolowej należy utworzyć aplikację webową
    - Wszelakie dane wynikowe (treść pliku, informacja czy dany napis reprezentuje nazwę pliku, czy katalogu) mają się pojawiać w przeglądarce WWW, a nie w konsoli
    - Zmodyfikuj treść pliku 'test/test3.js' — test ma weryfikować poprawność działania bieżącej implementacji

## Dla ambitnych

1. Korzystając z programu Apache Bench zbadaj wydajność dwóch aplikacji internetowych. Pierwsza z nich czyta zawartość pliku tekstowego, korzystając z funkcji modułu 'fs' w wersji synchronicznej. Druga, używa wersji asynchronicznej tych funkcji.
   - Pobierz kod źródłowy ww. aplikacji z repozytorium
   - Uruchom wersję synchroniczną — wykonaj komendę `node sync.js`
   - Za pomocą polecenia `ab -n 1000 -c 1000 -vhr http://localhost:8081/` zbadaj szybkość działania wersji synchronicznej
   - Uruchom wersję asynchroniczną — wykonaj komendę `node async.js`
   - Za pomocą polecenia `ab -n 1000 -c 1000 -vhr http://localhost:8000/` zbadaj szybkość działania wersji asynchronicznej

11. Napisz skrypt, który na podstawie danych zawartych w polu formularza, przetwarza zawartość pliku lub katalogu