

# NodeJS

WPROWADZENIE

# Kaj Białas

SENIOR FRONT-END DEVELOPER



@KAJ.BIALAS



@KAJ.BIALAS



@KajBialas



KA.BIALASJ@DEV-INDUSTRY.COM





# Przydatne materiały

**<https://nodejs.org/>** - oficjalny serwis Node JS

# Wprowadzenie

CZYM JEST NODE JS?



HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

#BlackLivesMatter

The 2021 Node.js User Survey is open now

Download for macOS (x64)

**14.15.4 LTS**  
Recommended For Most Users

**15.6.0 Current**  
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).



**Node JS** to wieloplatformowe środowisko  
uruchomieniowe javascript oparte o silnik  
**Chrome V8**

działa po **stronie serwera**, ale nie jest **serwerem**

**nie jest** frameworkiem web'owym

**Node JS** możemy uruchomić zarówno na **serwerze** jak i **lokalnej maszynie**

**Node JS** jest niezależny od systemu operacyjnego

**Node JS** aktywnie wykorzystuje:

**architekturę zdarzeń** oraz nieblokujące **operacje wejścia - wyjścia**

**npm** (node package manager) - największą bibliotekę rozszerzeń  
**open-source** na świecie



**Node JS** to to oprogramowanie stworzone w **C++**  
z zagnieżdżonym silnikiem **V8**.

**Przeglądarki webowe** wykorzystują:

obiekt **window**

**DOM**

**Node JS** wykorzystuje:

system **plików**

tworzenie **serwera**



tworzenie **REST API**

praca z **bazą danych**

**Node JS jest rozwiązaniem open source.**

# Instalacja

PIERWSZE URUCHOMIENIE NODE JS

[HOME](#) | [ABOUT](#) | [DOWNLOADS](#) | [DOCS](#) | [GET INVOLVED](#) | [SECURITY](#) | [CERTIFICATION](#) | [NEWS](#)

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

#BlackLivesMatter

The 2021 Node.js User Survey is open now

## Download for macOS (x64)


**14.15.4 LTS**  
Recommended For Most Users


**15.6.0 Current**  
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#) [Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).





HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

#BlackLivesMatter

The 2021 Node.js User Survey is open now

Download for macOS (x64)

**14.15.4 LTS**  
Recommended For Most Users

**15.6.0 Current**  
Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

```
node -v
```

- sprawdzenie zainstalowanej wersji **node**

```
node
```

- uruchomienie konsoli **node**

```
node server.js
```

- uruchomienie skryptu **js** w środowisku **node**

# Issue #0

Zainstaluj **Node JS**, lokalnie na swoim komputerze ( polecenie `node -v` - powinno wskazać wersję) .

Sprawdź działanie node js, pisząc podstawowy program w **konsoli node**.

Utwórz plik `server.js` oraz utwórz w nim **program dodający** 2 liczby.

Program dodający dwie liczby w przypadku uruchomienia go z jednym **argumentem** powinien wyświetlić zadaną wartość.

**Webserver** NODE JS

```
1 const http = require('http');
2
3 const server = http.createServer((req, res) => {
4   res.writeHead(200, {
5     'Content-Type': 'text/html'
6   });
7   res.end(`<div>Example content</div>`);
8 });
9
10 server.listen(5000, '127.0.0.1', () => console.log('Server is running at 5000'));
```



# Issue #1

Utwórz webserver w **Node JS**, który uruchomi się na porcie **8000**

Wyświetl w postaci strony internetowej, efekt działania programu z **Issue0**.

Program dodający dwie liczby w przypadku uruchomienia go z jednym **argumentem** powinien wyświetlić zadaną wartość.

# Konfiguracja projektu

NODE JS

```
npm init
```

- inicjalizacja projektu npm

```
npm install nazwaPakietu --save
```

- instalacja nowej biblioteki

```
npm install colors --save
```

- instalacja biblioteki **colors**

```
const colors = require('colors');  
console.log('hello'.green);
```

- użycie biblioteki colors w skrypcie **Node JS**

# Issue #2

Utwórz nowy projekt **NPM**, zawierający aplikację **Node JS**

Projekt powinien zawierać serwer WWW i serwować stronę, zawierającą aktualną datę. Data powinna być wyświetlona w formacie: **YYYY-MM-DDD**

Wykorzystaj bibliotekę **date-fns** lub **momentjs**

Server **Node JS**, powinien uruchamiać się po wybraniu komendy:

```
npm run start
```



```
npm install nodemon -g
```

- globalna instalacja pakietu nodemon

```
nodemon server.js
```

- uruchomienie skryptu przy użyciu nodemon

# Issue #3

Wykonaj globalną instalację pakietu **nodemon**

Uruchom dotychczasowy serwer w trybie watch, i zaobserwuj poprawność wykonanej operacji

Server **Node JS**, powinien uruchamiać się po wybraniu komendy:  
z wykorzystaniem pakietu **nodemon**

```
npm run start:watch
```

**Moduły** NODE JS

## Metoda **module.exports** / **require**

plik **add.js**

```
1 const add = (a, b) => a + b;  
2  
3 module.exports = add;
```

plik **index.js**

```
1 const add = require( './add' );  
2  
3 console.log( add( 4, 5 ) );
```

## Metoda **export** / **import**

plik **add.js**

```
1 export const add = (a, b) => a + b;
```

plik **index.js**

```
1 import { add } from './add';  
2  
3 console.log(add(6, 7));
```



**NodeJS** nie wspiera **export** / **import** z EcmaScript 6

**Istnieją** możliwości rozwiązania tego problemu, np. biblioteka **esm**

## Biblioteka **esm**

instalacja biblioteki

```
npm i esm
```

uruchomienie skryptu

```
nodemon -r esm index.mjs
```

```
node -r esm index.mjs
```

# Issue #4

Utwórz moduły **add** (dodający dwie liczby) oraz **sub** (odejmujący dwie liczby)

Wykorzystaj znane Ci metody tworzenia modułów.

# System plików

NODE JS

# odczyt

```
1 const fs = require('fs')
2
3 fs.readFile('./test.txt', 'utf8', (err, data) => {
4   if (err) {
5     console.error(err)
6     return
7   }
8   console.log(data)
9 })
```

```
1 const fs = require('fs')
2
3 try {
4   const data = fs.readFileSync('./joe/test.txt', 'utf8')
5   console.log(data)
6 } catch (err) {
7   console.error(err)
8 }
```

# Issue #5

Pobierz bazę użytkowników z lokalnego pliku **JSON**.

Wyświetl na stronie listę użytkowników (imię i nazwisko).



# zapis

```
1  const fs = require('fs')
2
3  const content = 'Some content!'
4
5  fs.writeFile('/Users/joe/test.txt', content, err => {
6    if (err) {
7      console.error(err)
8      return
9    }
10  })
```

# Issue #6

Pobierz bazę użytkowników z lokalnego pliku **JSON**.

Utwórz nowy plik **JSON**, zawierający posortowaną listę użytkowników (tylko imiona i nazwiska).

Zamiast wykorzystywać lokalny plik jako źródło, pobierz dane z API.

**Webserver** NODE JS

# Metody **HTTP**

**GET**

- pobranie zasobu wskazanego przez URI, może mieć postać warunkową

**HEAD**

- pobiera informacje o zasobie, stosowane do sprawdzania dostępności zasobu

**PUT**

- przyjęcie danych przesyłanych od klienta do serwera

**POST**

- przyjęcie danych przesyłanych od klienta do serwera

**DELETE**

- żądanie usunięcia zasobu, włączone dla uprawnionych użytkowników

**OPTIONS**

- informacje o opcjach i wymaganiach istniejących w kanale komunikacyjnym

**TRACE**

- diagnostyka, analiza kanału komunikacyjnego

**CONNECT**

- żądanie przeznaczone dla serwerów pośredniczących pełniących funkcje tunelowania

**PATCH**

- aktualizacja części danych

## Podstawowy serwer **HTTP**

```
1 const http = require('http');
2
3 const server = http.createServer((req, res) => {
4   res.writeHead(200, {
5     'Content-Type': 'text/html'
6   });
7   res.end(`<div>Example content</div>`);
8 });
9
10 server.listen(5000, '127.0.0.1', () => console.log('Server is running at 5000'));
```

## Podstawowy serwer **HTTP** - określanie ścieżek

```
1 const http = require('http');
2
3 const server = http.createServer((req, res) => {
4   if (req.url === '/') {
5     res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });
6     res.write('<h1>Strona główna</h1>');
7     res.end();
8   }
9
10 });
11
12 server.listen(5000, '127.0.0.1', () => console.log('Server Listen'));
```



# Issue #7

Zmodyfikuj aplikację, tak żeby zawierała ścieżki: **/**, **/about**, **/users**, **/contact**

Dla innych ścieżek powinna się załadować strona **404**

Wykorzystaj moduły, żeby wyeliminować powtarzający się kod.

## Podstawowy serwer **HTTP** - zwracanie plików **.html**

```
1 const http = require('http');
2 const fs = require('fs');
3 const path = require('path');
4
5 const server = http.createServer((req, res) => {
6   if (req.url === '/') {
7     fs.readFile(path.join(__dirname, "index.html"), (err, page) => {
8       res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });
9       res.end(page)
10     })
11   }
12
13   if (req.url === '/about') {
14     res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });
15     res.write('<h1>O nas</h1>');
16     res.end();
17   }
18 });
19
20 server.listen(5000, '127.0.0.1', () => console.log('Server Listen'));
```

# Issue #8

Wykorzystaj indywidualne templatki **.html** dla każdej z podstron

Templatka **/users**, powinna dynamicznie generować dane.

# Podstawowy serwer **HTTP** - zwracanie danych w formacie **JSON**

```
1 const http = require('http');
2 const fs = require('fs');
3 const path = require('path');
4
5 const USERS = [
6   {
7     id: 1,
8     name: 'Jack Daniels',
9   }
10 ];
11
12 const server = http.createServer((req, res) => {
13   if (req.url === '/') {
14     fs.readFile(path.join(__dirname, "index.html"), 'utf-8', (err, page) => {
15       res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' });
16       res.end(page)
17     })
18   }
19
20   if (req.url === '/api') {
21     res.writeHead(200, { 'Content-Type': 'application/json; charset=utf-8' });
22     res.end(JSON.stringify(USERS));
23   }
24 });
25
26 server.listen(5000, '127.0.0.1', () => console.log('Server Listen'));
```

# Issue #9

Utwórz API endpoint **/api/users**, zwracający listę użytkowników na podstawie lokalnego pliku **JSON**, w uproszczonej formie.

**Express JS** FRAMEWORK NODE JS



## Framework umożliwiający:

tworzenie **funkcji** obsługujących żądania o różnych **metodach HTTP** i skierowanych do różnych ścieżek w URL (tzw. **routing**).

integrację z **różnymi silnikami** do generowania widoków, które są tworzone na podstawie osadzanych danych w **szablonach stron**.

konfigurowania typowych **ustawień aplikacji** webowych jak np. **portu**, lokalizacji szablonów do generowania widoków odpowiedzi.

dodatkowe przetwarzanie żądań w **warstwie pośredniej** (tzw. "middleware"), które może być umieszczone w dowolnym miejscu łańcucha obsługi żądania.

# Podstawowe użycie

## instalacja

```
npm install express --save
```

## plik **index.js**

```
1 const express = require('express')
2 const app = express()
3 const port = 3000
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!')
7 })
8
9 app.listen(port, () => {
10   console.log(`Example app listening at http://localhost:${port}`)
11 })
```

# Issue #10

Utwórz aplikację Express, tak żeby zawierała ścieżki: **/**, **/about**, **/users**, **/contact**

Na tym etapie możesz serwować statyczne dane