

Node.js

Node.js

środowisko uruchomieniowe programów napisanych w Java Script
(back-end JavaScript runtime environment) - serwer

działa poza przeglądarką
(back-end)

open-source server

napisany w C++

cross platform
(działa na wielu platformach)

połączenia wykonuje asynchronicznie
(nie czeka na skończenie wykonania
połączenia aby rozpocząć kolejne połączenie,
- jak w php, języku synchronicznym)

korzysta z V8 (open-source Java
Script engine developed by
Google), który został napisany w
C++, działa synchronicznie

korzysta z wielowątkowości V8

sterowany zdarzeniami
(event-driven architecture)

Node.js

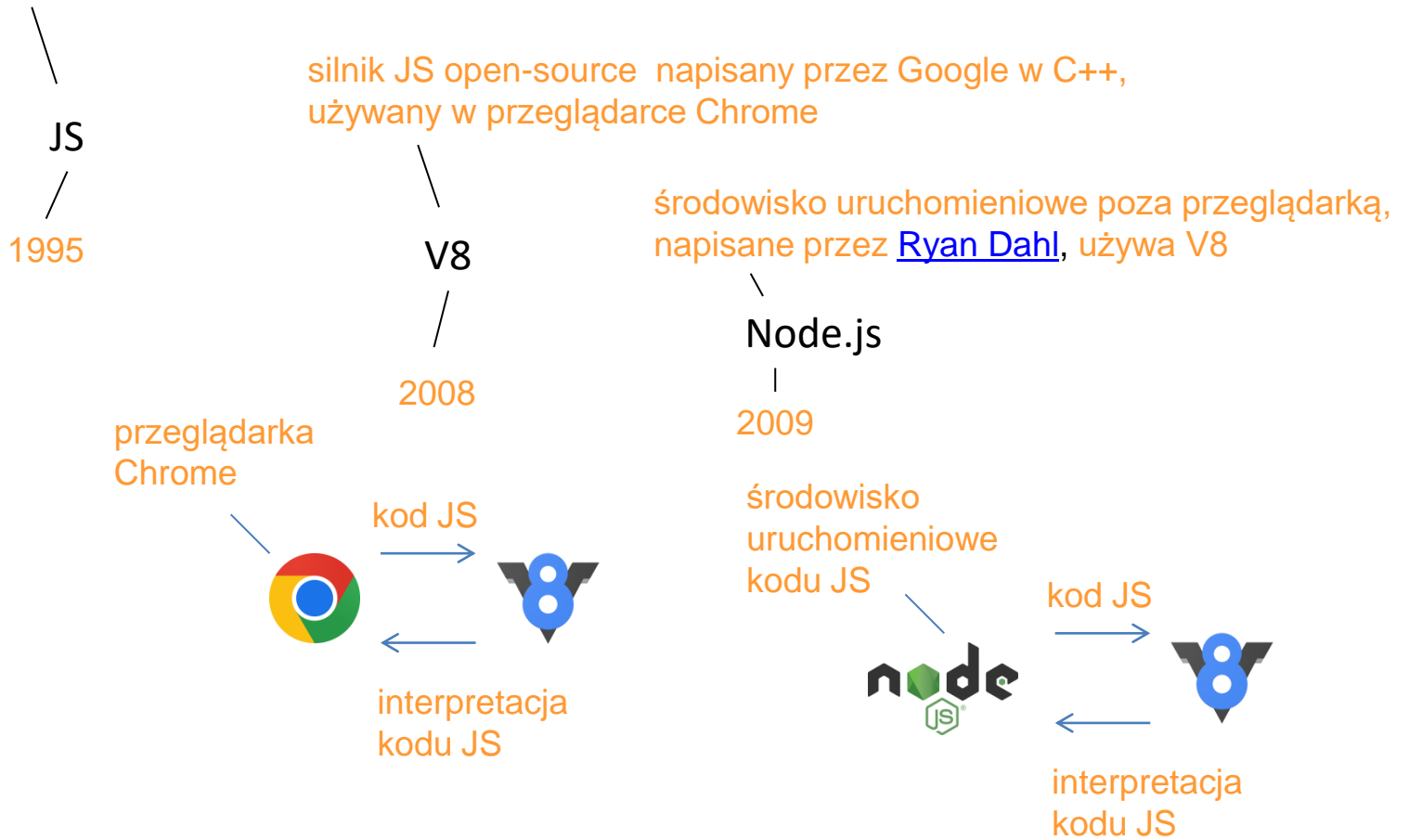
<https://www.youtube.com/watch?v=Txu3DhbDLpk>

<https://ichi.pro/pl/wprowadzenie-do-node-js-przewodnik-dla-poczatkujacych-po-node-js-i-npm-260213882550782>

npm (Node Package Manager) jest domyślnym
managerem pakietów (The World's Largest Software
Registry (Library))

Node.js

JS powstał w celu wprowadzenia interaktywności na stronach www



Node.js

The screenshot shows a web browser window with the URL <https://www.w3schools.com/nodejs/default.asp>. The page features a dark navigation bar with links for HTML, CSS, JAVASCRIPT, SQL, PYTHON, JAVA, PHP, BOOTSTRAP, and HOW TO. A sidebar on the left lists various Node.js topics, with 'Node.js HOME' highlighted in green. The main content area has a large heading 'Node.js Tutorial', a green button labeled '< Home', and a light green background with the text: 'Node.js is an open source server environment. Node.js allows you to run JavaScript on the server.' Below this is another green button labeled 'Start learning Node.js now »'.

<https://www.w3schools.com/nodejs/default.asp>

Node.js



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

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

Download for Windows (x64)

16.14.0 LTS

Recommended For Most Users

17.7.1 Current

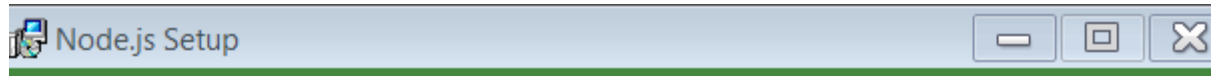
Latest Features

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

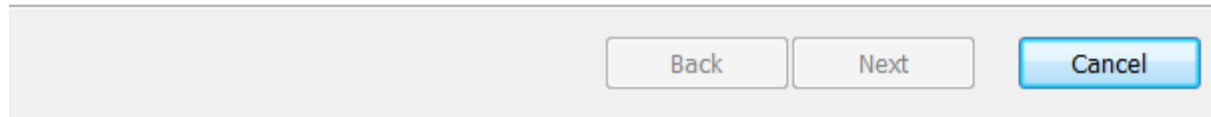
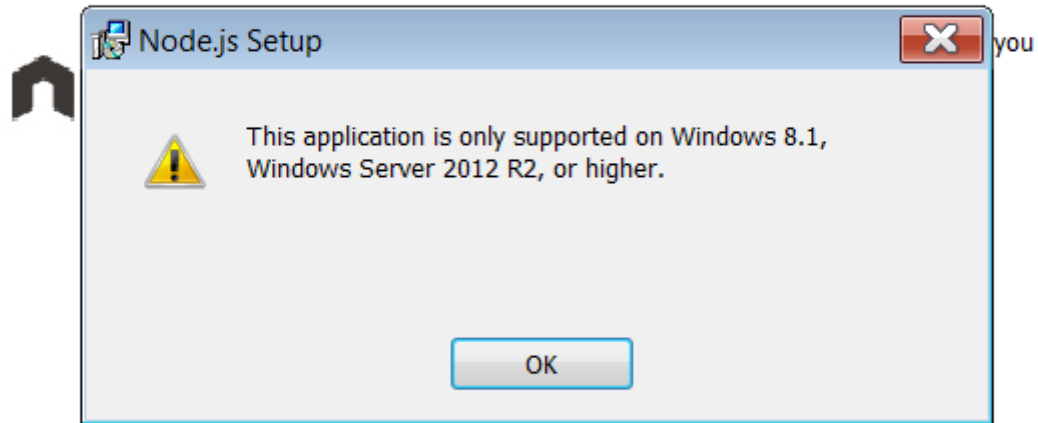
Or have a look at the [Long Term Support \(LTS\) schedule](#)

<https://nodejs.org/en/>

Node.js



Welcome to the Node.js Setup Wizard



[Zainstaluj Node.js na Win7](#) (wersja 13.14.0)

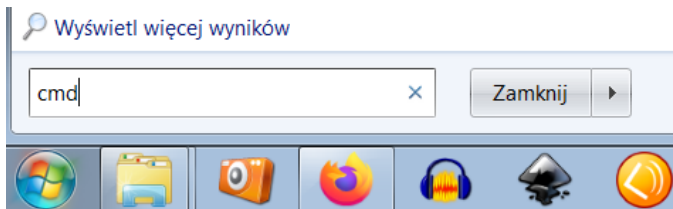
Node.js

```
Wiersz polecenia - node

C:\Users>node -v
v16.18.0

C:\Users>node
Welcome to Node.js v16.18.0.
Type ".help" for more information.
> .help
.break      Sometimes you get stuck, this gets you out
.clear      Alias for .break
.editor     Enter editor mode
.exit       Exit the REPL
.help       Print this help message
.load       Load JS from a file into the REPL session
.save       Save all evaluated commands in this REPL session to a file

Press Ctrl+C to abort current expression, Ctrl+D to exit the REPL
> 2*2
4
> let a = 5;
undefined
> a
5
> console.log(a + 10)
15
undefined
>
```



Po zainstalowaniu Node.js możemy korzystać z wiersza poleceń interaktywnego terminala REPL (read, eval, print, loop)

>node // włączenie terminala

>2*2

Polecenia nie są związane z przeglądarką (np. brak obiektu window)

Node.js

```
Wiersz polecenia - node
C:\Users>node
Welcome to Node.js v16.18.0.
Type ".help" for more information.
>
AbortController      AbortSignal          AggregateError       Array                 ArrayBuffer
Atomics              BigInt               BigInt64Array        BigInt64Array        Boolean
Buffer               DataView            Date                 Error                 EvalError
Event                EventTarget          FinalizationRegistry Float32Array          Float64Array
Function             Infinity            Int16Array           Int32Array            Int8Array
Intl                 JSON                 Map                  Math                  MessageChannel
MessageEvent         MessagePort          NaN                  Number                Object
Promise              Proxy                RangeError           ReferenceError        Reflect
RegExp               Set                  SharedArrayBuffer   String                Symbol
SyntaxError          TextDecoder          TextEncoder          TypeError             URIError
URL                  URLSearchParams     Uint16Array          Uint32Array           Uint8Array
Uint8ClampedArray   WeakMap              WeakRef              assert                async_hooks
_                    _error              child_process        clearImmediate       clearInterval
btoa                 buffer               console               constants             crypto
clearTimeout         decodeURI            decodeURIComponent  dgram                 diagnostics_channel
domain               encodeURI            encodeURIComponent  escape                 dns
events               fs                   global                 globalThis            eval
http2                https                inspector             isFinite              http
module               net                  os                     parseFloat            isNaN
path                 perf_hooks           performance           process                parseInt
querystring          queueMicrotask       readline              repl                   punycode
setImmediate         setInterval          setTimeout            stream                 require
sys                  timers                tls                    trace_events          tty
undefined            unescape             url                    util                    v8
```

po dwukrotnym naciśnięciu klawisza TAB dostajemy listę funkcji i narzędzi dostępnych z poziomu konsoli node

Node.js

```
C:\Windows\system32\cmd.exe - node
Microsoft Windows [Wersja 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Marek>node
Welcome to Node.js v13.14.0.
Type ".help" for more information.
> global
<ref *1> Object [global] {
  global: [Circular *1],
  clearInterval: [Function: clearInterval],
  clearTimeout: [Function: clearTimeout],
  setInterval: [Function: setInterval],
  setTimeout: [Function: setTimeout] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  },
  queueMicrotask: [Function: queueMicrotask],
  clearImmediate: [Function: clearImmediate],
  setImmediate: [Function: setImmediate] {
    [Symbol(nodejs.util.promisify.custom)]: [Function (anonymous)]
  }
}
> window
Uncaught ReferenceError: window is not defined
>
```

obiekt global jest nadrzędny
w stosunku do pozostałych obiektów
(podobnie jak obiekt window
w przeglądarkach)

brak obiektu window w Node.js

Node.js

```
C:\Windows\system32\cmd.exe - node
TEMP: 'C:\\Users\\Marek\\AppData\\Local\\Temp',
TMP: 'C:\\Users\\Marek\\AppData\\Local\\Temp',
USERDOMAIN: 'Komp',
USERNAME: 'Marek',
USERPROFILE: 'C:\\Users\\Marek',
VBOX_INSTALL_PATH: 'C:\\Program Files\\Oracle\\VirtualBox\\',
VS100COMNTOOLS: 'C:\\Program Files (x86)\\Microsoft Visual Studio 10.0\\Comm
\\',
VS120COMNTOOLS: 'E:\\visual_studio\\Common7\\Tools\\',
windir: 'C:\\Windows',
XNAGSShared: 'C:\\Program Files (x86)\\Common Files\\Microsoft Shared\\XNA\\
XNAGSv4: 'C:\\Program Files (x86)\\Microsoft XNA\\XNA Game Studio\\v4.0\\'
},
title: 'C:\\Windows\\system32\\cmd.exe - node',
argv: [ 'C:\\Program Files\\nodejs\\node.exe' ],
execArgv: [],
pid: 592,
ppid: 4680,
execPath: 'C:\\Program Files\\nodejs\\node.exe',
debugPort: 9229,
argv0: 'node',
_preload_modules: [],
[Symbol(kCapture)]: false
```

PID (Process Identifier - node)
PPID (Parent Process Identifier – PID cmd)

obiekt process

Node.js

```
Wiersz polecenia - node
>
C:\Users>
C:\Users>node
Welcome to Node.js v16.18.0.
Type ".help" for more information.
> os.version()
'Windows 10 Pro'
>
> os.arch()
'x64'
>
> os.platform()
'win32'
>
> os.release()
'10.0.19043'
> os.
os.__proto__          os.constructor       os.hasOwnProperty   os.isPrototypeOf
os.propertyIsEnumerable os.toLocaleString    os.toString         os.valueOfus: [Function: cpus], endiannes...

os.EOL                os.arch              os.constants         os.cpus
os.devNull            os.endianness       os.freemem           os.getPriority
os.homedir            os.hostname          os.loadavg           os.machine
os.networkInterfaces  os.platform         os.release           os.setPriority
os.tmpdir             os.totalmem         os.type              os.uptime
os.userInfo           os.version
```

odwołanie do zainstalowanego modułu os
udostępniającej informacje o systemie

Node.js

The screenshot shows the npm website search results for the query 'colors'. The browser address bar shows 'https://www.npmjs.com/search?q=colors'. The page header includes the npm logo, a search bar with 'colors' entered, and buttons for 'Search', 'Sign Up', and 'Sign In'. Below the search bar, it states '4033 packages found' with pagination controls showing page 1 of 202. On the left, there are 'Sort Packages' options: Optimal, Popularity, Quality, and Maintenance. The main content area lists three packages: 'colors' (published 1.4.0, 3 years ago), 'chalk' (published 5.1.2, a month ago), and 'ansi-colors' (published 4.1.3, 6 months ago). Each package entry includes a description, a list of related tags, and a small 'p q m' icon.

<https://www.npmjs.com/>

wyszukiwarka pakietów

Node.js

The image shows a browser window displaying the Node.js website. The website has a dark header with the Node.js logo and navigation links: HOME, ABOUT, DOWNLOADS, DOCS, GET INVOLVED, SECURITY, CERTIFICATION, NEWS. Below the header, it states "Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine." and offers "Download for Windows (x64)" with two options: "16.14.0 LTS Recommended For Most Users" and "17.7.1 Current Latest Features". Below these are links for "Other Downloads | Changelog | API Docs". A link for "Or have a look at the Long Term Support (LTS) schedule" is also present.

Below the website screenshot, the Chrome DevTools console is shown. The console has tabs for Elements, Console, Recorder, Sources, Network, Performance, Memory, Application, Security, and Lighthouse. The Console tab is active, showing the following commands and output:

```
> 2*2  
< 4  
> window.document.querySelectorAll('img')  
< ▶ NodeList(2) [img, img]
```

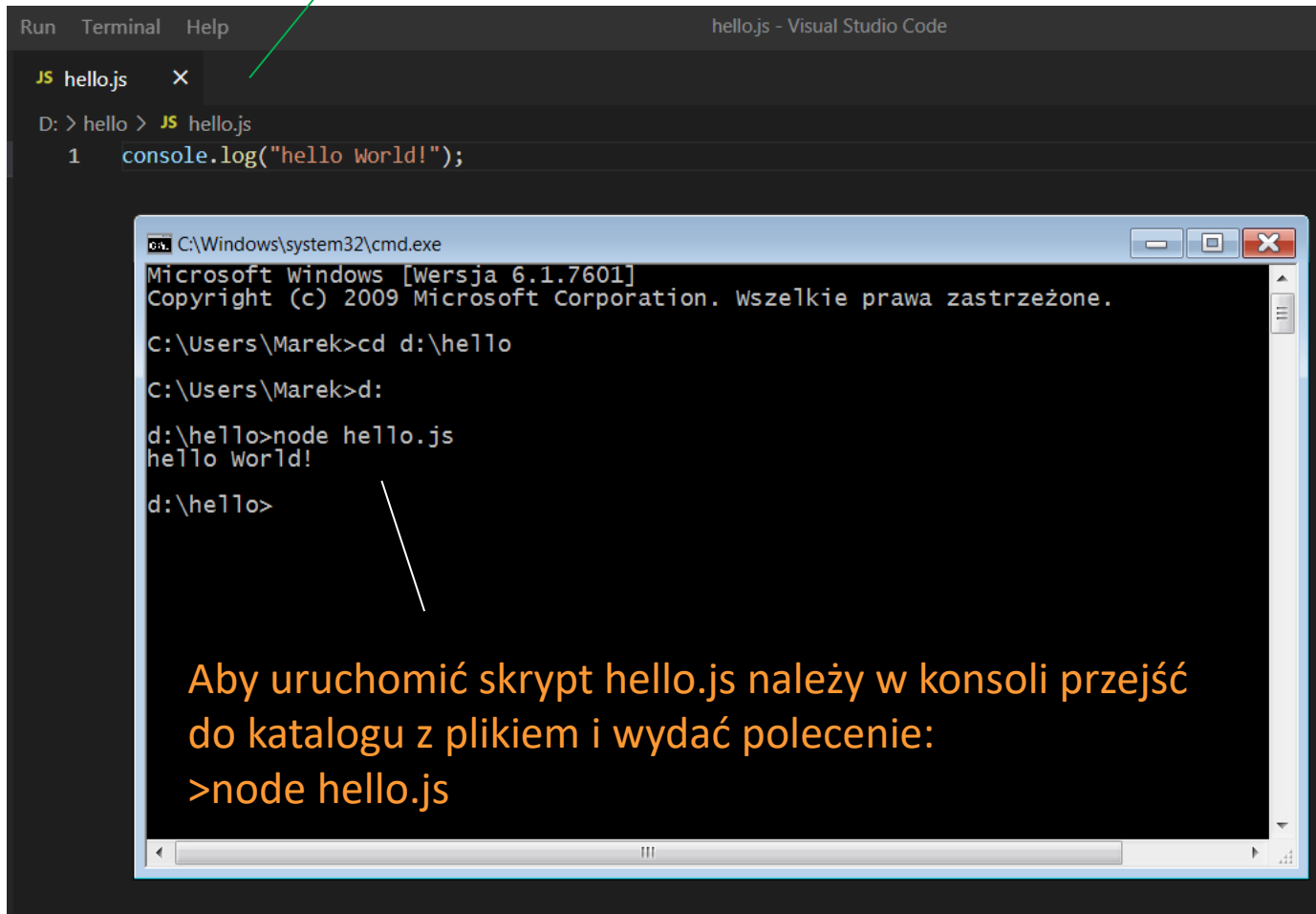
A white arrow points from the text on the right to the console output. At the bottom of the browser window, there are tabs for "Console" and "What's New".

Konsola w Chrome (F12) – można korzystać z poleceń w JS:
>2*2
>window.document.querySelectorAll('img')
Polecenia są związane z przeglądarką (jest obiekt window).

Node.js

Uruchamianie aplikacji Noda – pliku hello.js bez zakładania projektu

Plik hello.js otwarty w edytorze Visual Studio Code



The image shows a Visual Studio Code editor window with a file named 'hello.js' open. The code in the editor is:

```
1 console.log("hello world!");
```

Below the editor is a Windows Command Prompt window. The command prompt shows the following sequence of commands and output:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [wersja 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\Marek>cd d:\hello
C:\Users\Marek>d:
d:\hello>node hello.js
hello world!
d:\hello>
```

Below the command prompt, there is a text box with the following text:

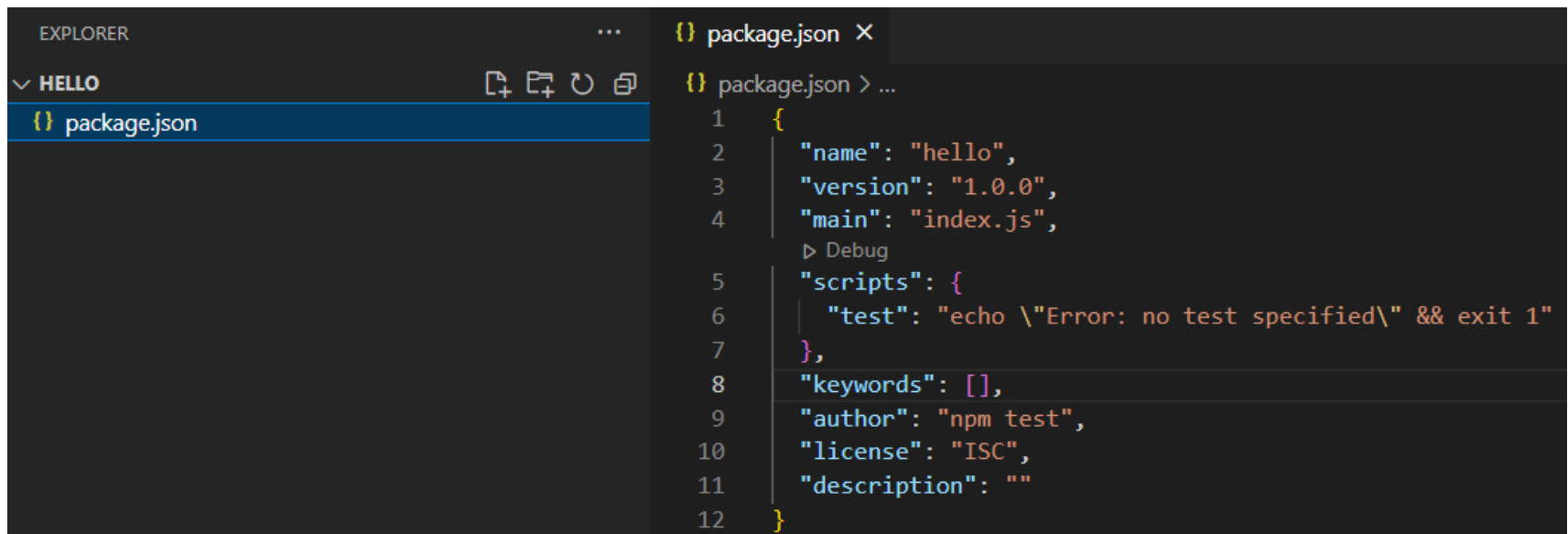
Aby uruchomić skrypt hello.js należy w konsoli przejść do katalogu z plikiem i wydać polecenie:
>node hello.js

Node.js nowy projekt

w Visual Studio Code w katalogu projektu (hello) w terminalu wydajemy polecenie inicjalizujące projekt:

```
npm -y init
```

wygenerowany zostaje plik package.json z konfiguracją projektu



```
EXPLORER
HELLO
  package.json

package.json
1 {
2   "name": "hello",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "test": "echo \"Error: no test specified\" && exit 1"
7   },
8   "keywords": [],
9   "author": "npm test",
10  "license": "ISC",
11  "description": ""
12 }
```

TypeScript

TS TypeScript

Download Docs Handbook Community Playground Tools

TypeScript is **JavaScript**
with syntax for types.

TypeScript is a strongly typed programming language that builds on JavaScript, giving you better tooling at any scale.

Try TypeScript Now
Online or via npm

□□□

<https://www.typescriptlang.org/>

TypeScript



Tutorials ▾

Exercises ▾

Certificates ▾

Services ▾

Search...



< W3.CSS C C++ C# BOOTSTRAP REACT MYSQL JQUERY EXCEL

TypeScript tutorial

TS HOME

TS Introduction

TS Get Started

TS Simple Types

TS Special Types

TS Arrays

TS Tuples

TS Object Types

TS Enums

TS Aliases & Interfaces

TS Union Types

TS Functions

TS Casting

TS Classes

TS Basic Generics

TS Utility Types

TypeScript Tutorial

< Home

TypeScript is JavaScript with added syntax for types.

Start learning TypeScript now »

<https://www.w3schools.com/typescript/>

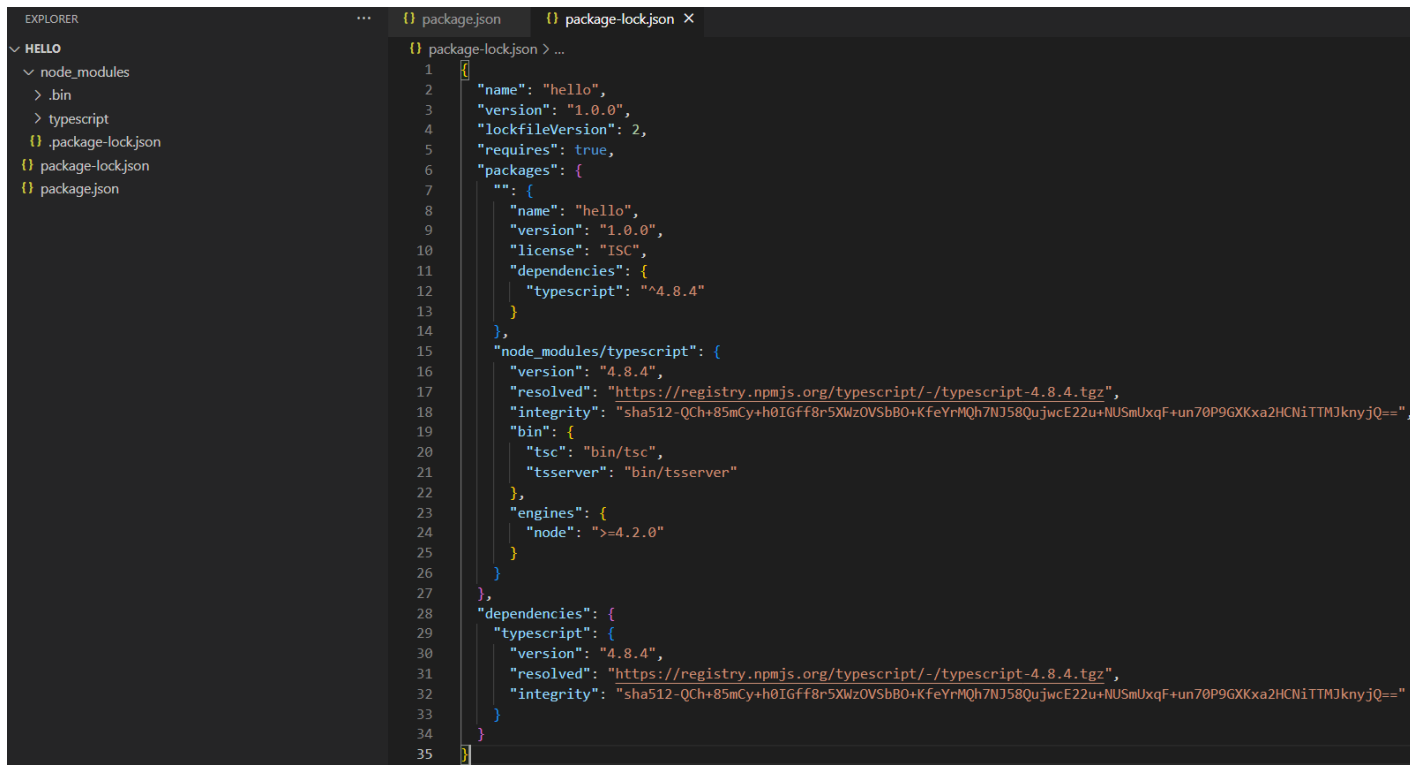
Node.js

instalacja TypeScript

w Visual Studio Code w katalogu projektu (hello) w terminalu wydajemy polecenie instalujące język TypeScript:

```
npm install typescript
```

wygenerowany zostaje plik package-lock.json z informacjami dotyczącymi języka oraz katalog node_modules z plikami obsługującymi TypeScript



```
1 {
2   "name": "hello",
3   "version": "1.0.0",
4   "lockfileVersion": 2,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "hello",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "typescript": "^4.8.4"
13      }
14    },
15    "node_modules/typescript": {
16      "version": "4.8.4",
17      "resolved": "https://registry.npmjs.org/typescript/-/typescript-4.8.4.tgz",
18      "integrity": "sha512-QCh+85mCy+h0IGff8r5XWz0VSbB0+KfeYrMQh7NJ58QujwcE22u+NUSmUxqF+un70P9GXka2HCNiTTMJknyjQ==",
19      "bin": {
20        "tsc": "bin/tsc",
21        "tsserver": "bin/tsserver"
22      },
23      "engines": {
24        "node": ">=4.2.0"
25      }
26    }
27  },
28  "dependencies": {
29    "typescript": {
30      "version": "4.8.4",
31      "resolved": "https://registry.npmjs.org/typescript/-/typescript-4.8.4.tgz",
32      "integrity": "sha512-QCh+85mCy+h0IGff8r5XWz0VSbB0+KfeYrMQh7NJ58QujwcE22u+NUSmUxqF+un70P9GXka2HCNiTTMJknyjQ=="
33    }
34  }
35 }
```

Node.js

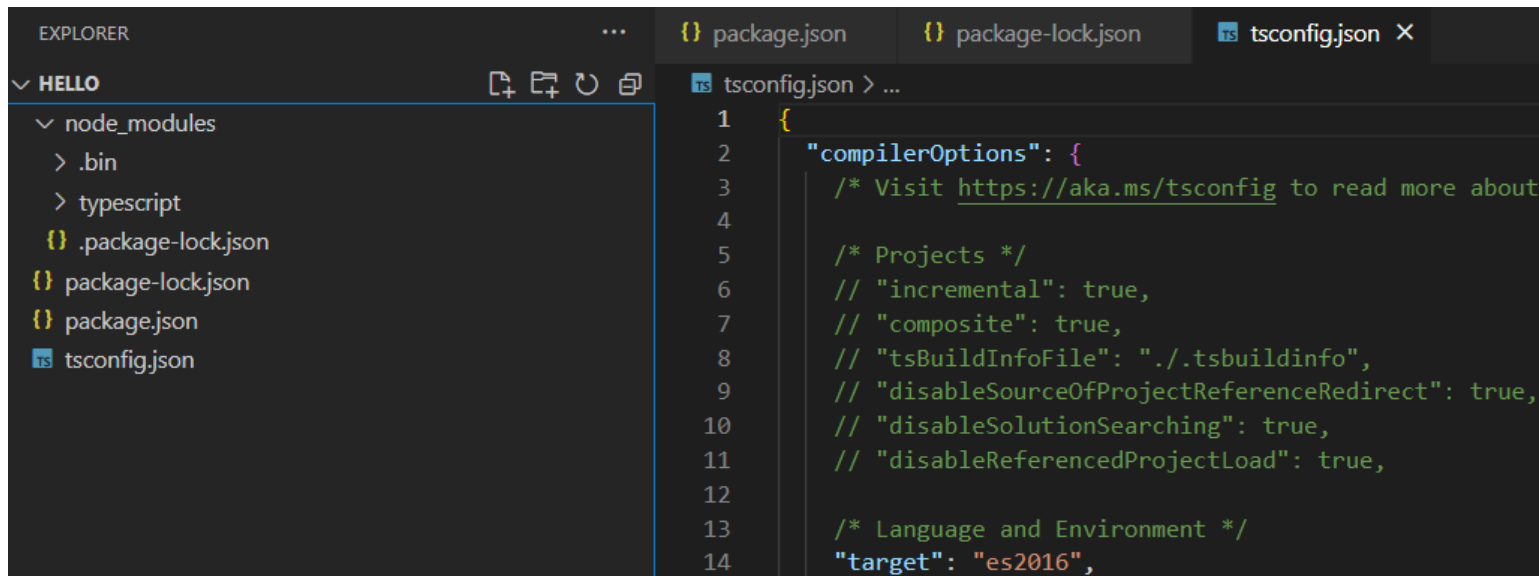
inicjalizacja TypeScript

w Visual Studio Code w katalogu projektu (hello) w terminalu wydajemy polecenie inicjalizujące język TypeScript:

```
hello> npx tsc -init
```

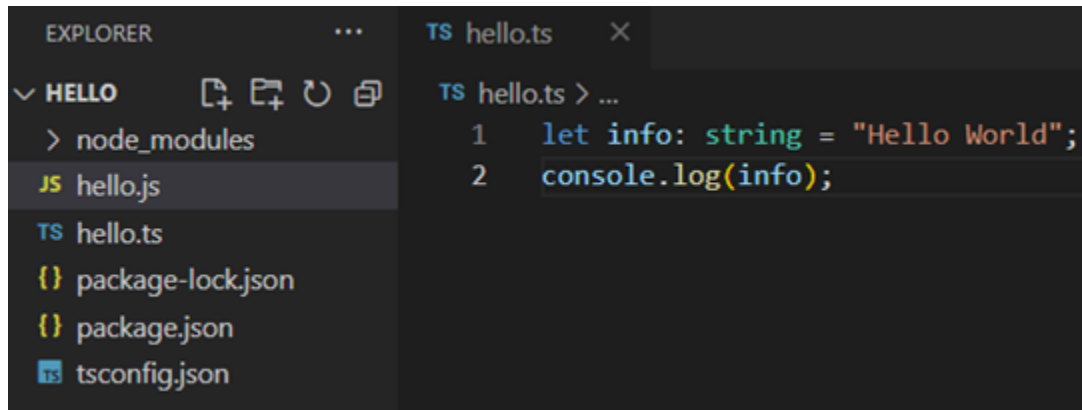
npx: Node Package Execute - uruchomienie modułu z katalogu node_modules

wygenerowany zostaje plik tsconfig.json z konfiguracją kompilatora



```
tsconfig.json > ...
1  {
2    "compilerOptions": {
3      /* Visit https://aka.ms/tsconfig to read more about
4
5      /* Projects */
6      // "incremental": true,
7      // "composite": true,
8      // "tsBuildInfoFile": "./.tsbuildinfo",
9      // "disableSourceOfProjectReferenceRedirect": true,
10     // "disableSolutionSearching": true,
11     // "disableReferencedProjectLoad": true,
12
13     /* Language and Environment */
14     "target": "es2016",
```

Node.js plik hello.ts



```
EXPLORER
HELLO
  > node_modules
  JS hello.js
  TS hello.ts
  {} package-lock.json
  {} package.json
  tsconfig.json

TS hello.ts
TS hello.ts > ...
1 let info: string = "Hello World";
2 console.log(info);
```

tworzymy plik hello.ts w języku TypeScript

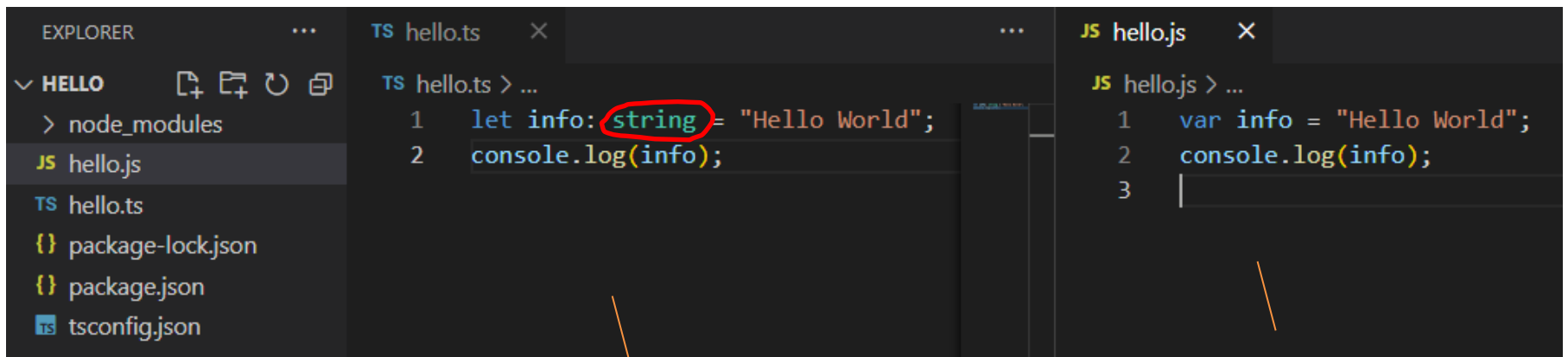
Node.js TypeScript

aby uruchomić plik ts w środowisku node.js należy go przekonwertować do js

w Visual Studio Code w katalogu projektu (hello) w terminalu kompilujemy plik hello.ts - wydajemy polecenie konwertujące plik hello.ts do języka JavaScript:

```
npx tsc hello.ts
```

wygenerowany zostaje plik w języku JavaScript hello.js który można uruchomić w node.js



The screenshot shows the Visual Studio Code interface with three panels. On the left is the Explorer view showing a project named 'HELLO' with files: 'node_modules', 'hello.js', 'hello.ts', 'package-lock.json', 'package.json', and 'tsconfig.json'. The middle panel shows the source TypeScript file 'hello.ts' with two lines of code: '1 let info: string = "Hello World";' and '2 console.log(info);'. The word 'string' in the first line is circled in red. The right panel shows the compiled JavaScript file 'hello.js' with two lines of code: '1 var info = "Hello World";' and '2 console.log(info);'. The third line in the JS panel is empty.

plik źródłowy (jest typ string - statyczny mechanizm typowania)

wygenerowany plik po kompilacji (brak typu string – dynamiczny mechanizm typowania)

Node.js TypeScript

```
PS D:\hello> node hello.js  
Debugger attached.  
Hello World  
Waiting for the debugger to disconnect...
```

aby uruchomić plik hello.js wpisujemy polecenie:

```
node hello.js
```

```
npx tsc --watch
```

polecenie uruchamiające tryb nasłuchu – automatyczną kompilację pliku ts po wykryciu w nim zmiany

Node.js TypeScript

hello.ts

hello.js

```
TS hello.ts > ...
1 let tekst: string = "napis"; // typ tekstowy
2 let decimal: number = 3.14; // typ liczbowy dziesiętny
3 let hex: number = 0xffff; // typ liczb szesnastkowy
4 let octal: number = 0o765; // typ liczbowy ósemkowy
5 let binary: number = 0b1110; // typ liczbowy binarny
6 let logic: boolean = true; // typ logiczny
7
8 let wszystkoJednoCo: any = 23; // typ dynamiczny
9 wszystkoJednoCo = "napis"; // typ string
10 wszystkoJednoCo = false; // typ boolean
11
12 console.log(`wszystkoJednoCo:
13     wartość: ${wszystkoJednoCo}
14     nazwa typu: ${typeof wszystkoJednoCo}`);
15
16 let common: number | string // zmienna może przyjmować
17 // typ tekstowy lub liczbowy
18 let marka: "bmw" | "mercedes" | "ford"; // typ jako zbiór wartości
19 //marka = "opel"; // błąd - brak opła w zdefiniowanych wartościach'
20
21 let auta: string[] = ["bmw", "mercedes", "ford"]; // tablica
22 let krotka: [number, boolean] = [25, true]; // krotka - tablica różnych typów
23 enum colors {red, green, blue}; // enum - wyliczenie
24 let czerwony = colors.red; // korzystanie z enum colors
25 console.log(czerwony);
```

statyczne typowanie w TypeScript

```
JS hello.js > ...
1 var tekst = "napis"; // typ tekstowy
2 var decimal = 3.14; // typ liczbowy dziesiętny
3 var hex = 0xffff; // typ liczb szesnastkowy
4 var octal = 501; // typ liczbowy ósemkowy
5 var binary = 14; // typ liczbowy binarny
6 var logic = true; // typ logiczny
7
8 var wszystkoJednoCo = 23; // typ dynamiczny
9 wszystkoJednoCo = "napis"; // typ string
10 wszystkoJednoCo = false; // typ boolean
11
12 console.log(`wszystkoJednoCo:
13     wartość: ${wszystkoJednoCo}
14     nazwa typu: ${typeof wszystkoJednoCo}`);
15
16 var common; // zmienna może przyjmować
17 // typ tekstowy lub liczbowy
18 var marka; // typ jako zbiór wartości
19 //marka = "opel"; // błąd - brak opła w zdefiniowanych wartościach'
20
21 var auta = ["bmw", "mercedes", "ford"]; // tablica
22 var krotka = [25, true]; // krotka - tablica różnych typów
23 var colors;
24 (function (colors) {
25     colors[colors["red"] = 0] = "red";
26     colors[colors["green"] = 1] = "green";
27     colors[colors["blue"] = 2] = "blue";
28 })(colors || (colors = {}));
29 // enum - wyliczenie
30 var czerwony = colors.red; // korzystanie z enum colors
31 console.log(czerwony);
32
```

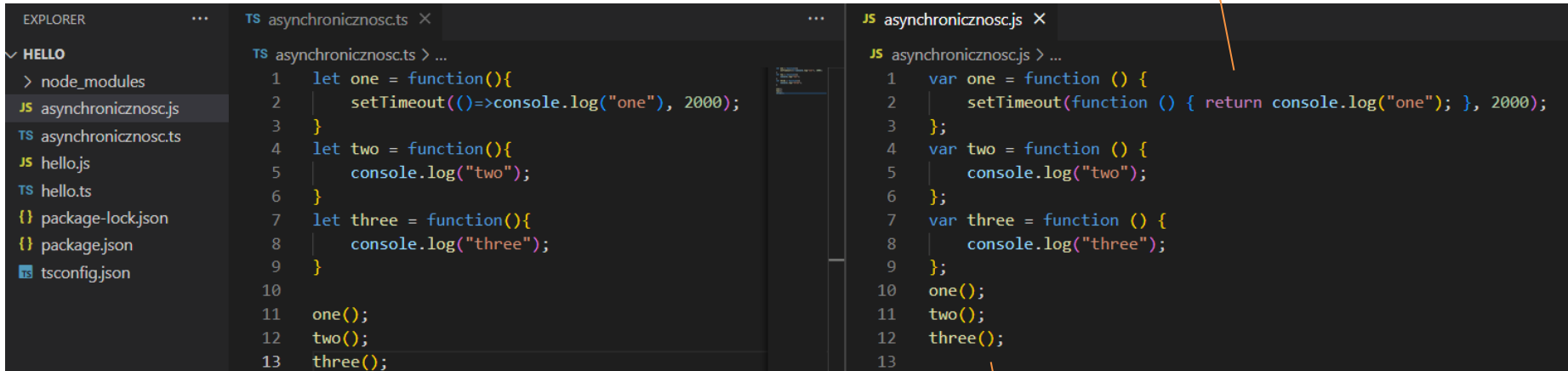
dynamiczne typowanie w JavaScript

```
PS D:\hello> node hello.js
Debugger attached.
wszystkoJednoCo:
  wartość: false
  nazwa typu: boolean
0
```

uruchomienie hello.js

Node.js asynchroniczność

funkcja one ma ustawiony 2s timeout

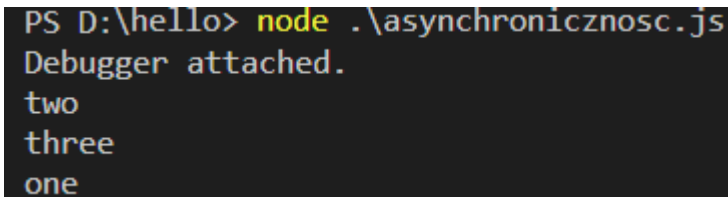


The screenshot shows two editor windows. The left window is titled 'TS asynchronicznosc.ts' and contains the following code:

```
1 let one = function(){
2   setTimeout(()=>console.log("one"), 2000);
3 }
4 let two = function(){
5   console.log("two");
6 }
7 let three = function(){
8   console.log("three");
9 }
10
11 one();
12 two();
13 three();
```

The right window is titled 'JS asynchronicznosc.js' and contains the following code:

```
1 var one = function () {
2   setTimeout(function () { return console.log("one"); }, 2000);
3 };
4 var two = function () {
5   console.log("two");
6 };
7 var three = function () {
8   console.log("three");
9 };
10 one();
11 two();
12 three();
13
```



```
PS D:\hello> node .\asynchronicznosc.js
Debugger attached.
two
three
one
```

funkcje w pliku źródłowym są wywoływane po kolei: one, two, three

podczas uruchomienia skryptu funkcje są wykonywane asynchronicznie: two, three, one (funkcja one z timoutem nie blokuje programu) – node.js ma wbudowany mechanizm, który to umożliwia (pomimo synchroniczności V8)

Node.js

kod, który można wykorzystać w projekcie

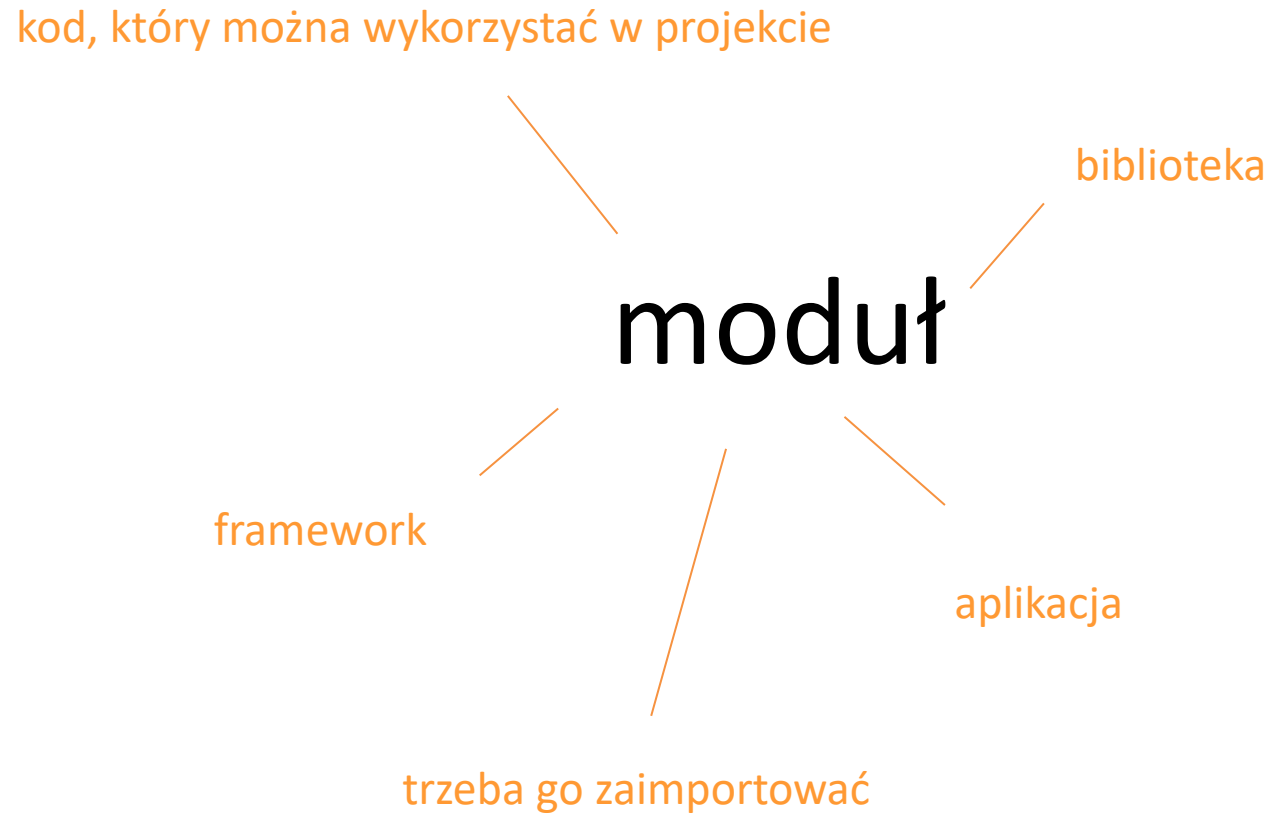
biblioteka

moduł

framework

aplikacja

trzeba go zaimportować



Node.js

Node.js Built-in Modules

[← Previous](#)

Node.js has a set of built-in modules which you can use without any further installation.

Here is a list of the built-in modules of Node.js version 6.10.3:

Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do DNS lookups and name resolution functions
domain	Deprecated. To handle unhandled errors
events	To handle events
fs	To handle the file system
http	To make Node.js act as an HTTP server
https	To make Node.js act as an HTTPS server.
net	To create servers and clients
os	Provides information about the operation system
path	To handle file paths
punycode	Deprecated. A character encoding scheme
querystring	To handle URL query strings
readline	To handle readable streams one line at the time
stream	To handle streaming data
string_decoder	To decode buffer objects into strings
timers	To execute a function after a given number of milliseconds
tls	To implement TLS and SSL protocols

https://www.w3schools.com/nodejs/ref_modules.asp

informacje o systemie operacyjnym

moduł os plik app.js

import modułu

```
EXPLORER  ...  JS app.js  X
└─ MODULOS
   └─ JS app.js
      1  // import modułu os
      2  const os = require('os');
      3
      4  console.log(`
      5      version: ${os.version()}
      6      type: ${os.type()}
      7      uptime: ${os.uptime()}
      8      platform: ${os.platform()}
      9      release: ${os.release()}
     10      hostname: ${os.hostname()}
     11      totalmame: ${os.totalmem()}
     12  `)
```

uruchomienie
skryptu
app.js

```
PS D:\> node app.js
```

```
version: Windows 10 Pro
type: Windows_NT
uptime: 196892
platform: win32
release: 10.0.19043
hostname: DESKTOP-G94GGIJ
totalmame: 8482922496
```

informacje o systemie

https://www.w3schools.com/nodejs/ref_os.asp

moduł os import w pliku ts

```
TS modulOs.ts  
1 // import modułu os  
2 import * as os from 'os';
```

import modułu os w pliku ts

```
npm i @types/node
```

aby środowisko rozpoznało nazwę os
trzeba zainstalować dodatkowy pakiet

This package contains type definitions for Node.js

<https://www.npmjs.com/package/@types/node>

moduł url — analizowanie adresu url

```
EXPLORER  ...  JS app.js  X
v MODULURL
JS app.js
1 // import modułu url
2 const url = require('url');
3
4 // adres do parsowania
5 const address = "http://pracownia.palacmlodziezy.lodz.pl/a/b/c/index.html"
6
7 const addr = url.parse(address, true)
8
9 console.log(`
10 |         |         |         |         |
11 |         |         |         |         |         |
12 |         |         |         |         |         |         |
13 |         |         |         |         |         |         |
14 |         |         |         |         |         |         |
15 |         |         |         |         |         |         |
    `)`
```

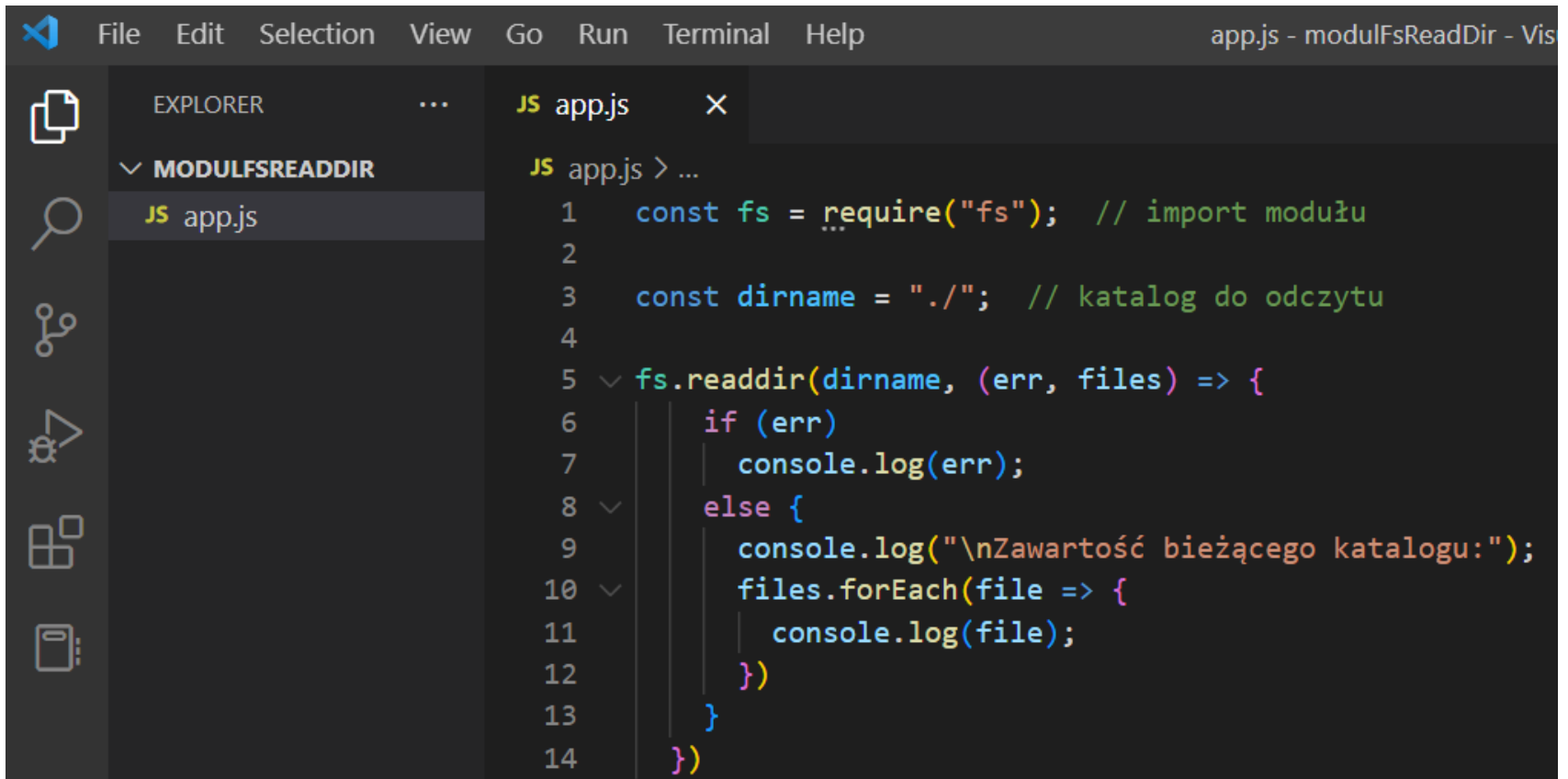
```
PS D:\> node app.js
```

```
host: pracownia.palacmlodziezy.lodz.pl
href: http://pracownia.palacmlodziezy.lodz.pl/a/b/c/index.html
hostname: pracownia.palacmlodziezy.lodz.pl
path: /a/b/c/index.html
protocol: http:
```

moduł fs

dostęp do plików

zawartość katalogu — readdir



```
File Edit Selection View Go Run Terminal Help app.js - modulFsReadDir - Vis
EXPLORER
  MODULFSREADDIR
    JS app.js
  JS app.js
  JS app.js > ...
1  const fs = require("fs"); // import modułu
2
3  const dirname = "."; // katalog do odczytu
4
5  fs.readdir(dirname, (err, files) => {
6    if (err)
7      console.log(err);
8    else {
9      console.log("\nZawartość bieżącego katalogu:");
10     files.forEach(file => {
11       console.log(file);
12     })
13   }
14 })
```

```
PS D:\> node app.js
```

```
Zawartość bieżącego katalogu:
app.js
```

moduł fs readfile

zawartość pliku

dostęp do plików

```
File Edit Selection View Go Run Terminal Help app.js - modulFsReadFile - Visual St
EXPLORER
MODULFSREAD...
JS app.js
plik.txt
JS app.js
JS app.js > ...
1 // import modułu fs
2 var fs = require('fs');
3
4 // czytamy zawartość pliku plik.txt
5 fs.readFile('plik.txt', 'utf8', function(err, data){
6     if (err) {
7         return console.log("błąd");
8     }
9     // wypisujemy w logu zawartość pliku
10    console.log(data);
11 });
```

```
plik.txt
plik.txt
1 tajna informacja
```

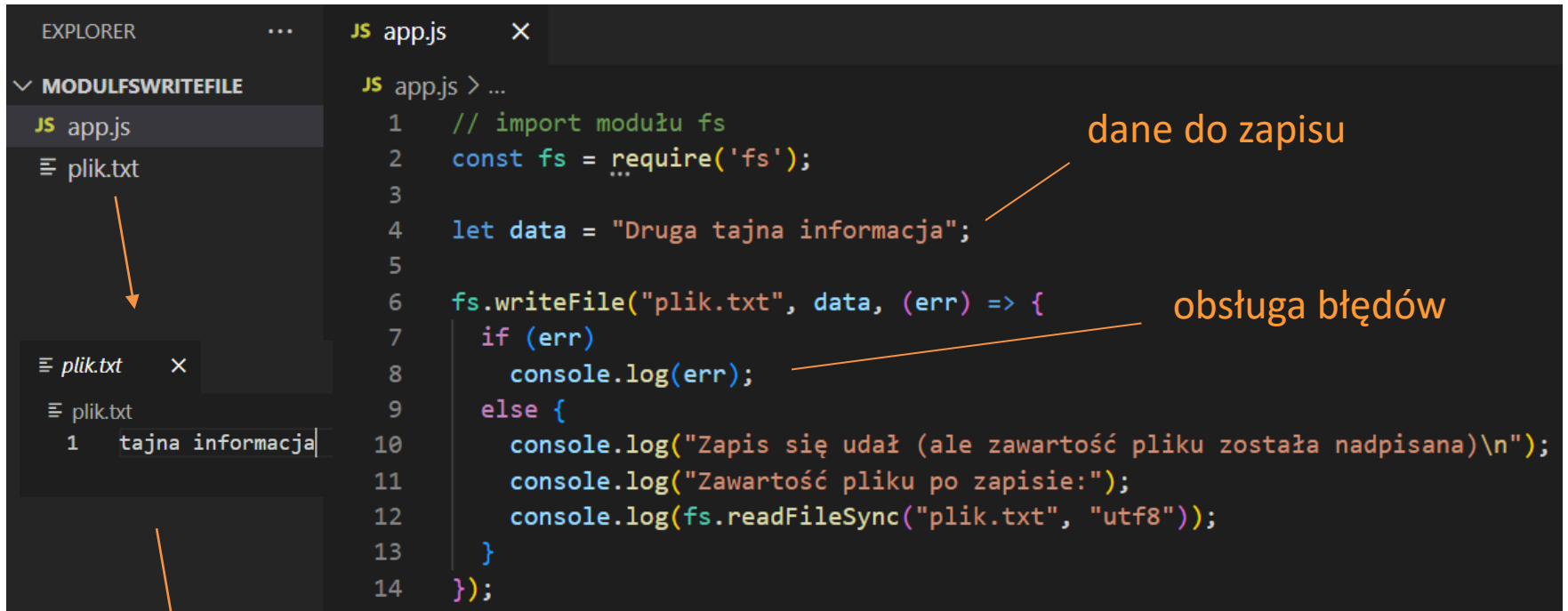
```
PS D:\> node app.js
tajna informacja
```

zawartość pliku plik.txt

moduł fs writefile

dostęp do plików

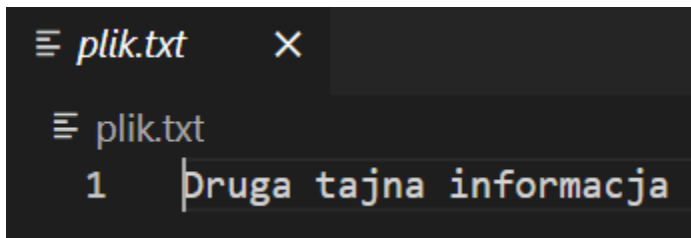
zapis do pliku
(nadpisanie zawartości),
gdy nie będzie pliku o podanej nazwie to go stworzy



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'MODULFSWRITEFILE' with a file 'plik.txt'. An arrow points from the Explorer to the code editor. The code editor shows the following JavaScript code:

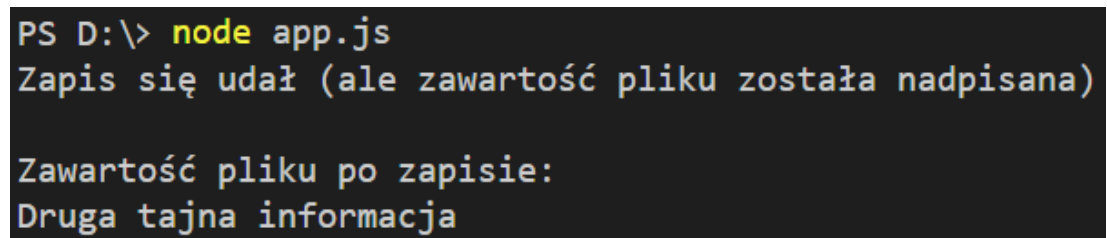
```
1 // import modułu fs
2 const fs = require('fs');
3
4 let data = "Druga tajna informacja";
5
6 fs.writeFile("plik.txt", data, (err) => {
7   if (err)
8     console.log(err);
9   else {
10    console.log("Zapis się udał (ale zawartość pliku została nadpisana)\n");
11    console.log("Zawartość pliku po zapisie:");
12    console.log(fs.readFileSync("plik.txt", "utf8"));
13  }
14 });
```

Annotations in orange text point to specific parts of the code: 'dane do zapisu' points to the `data` variable on line 4, and 'obsługa błędów' points to the `if (err)` block on lines 7-8.



This screenshot shows the file explorer with 'plik.txt' selected. The file content is displayed as:

```
1 druga tajna informacja
```



The terminal window shows the command `node app.js` being executed. The output is:

```
PS D:\> node app.js
Zapis się udał (ale zawartość pliku została nadpisana)
Zawartość pliku po zapisie:
Druga tajna informacja
```

zawartość pliku plik.txt
(nadpisanie zawartości)

moduł fs appendfile

dostęp do plików

zapis do pliku
(dopisanie zawartości)

```
EXPLORER ... JS app.js X
└─ MODULFSAPPENDFILE
   └─ JS app.js
      └─ plik.txt
         └─ plik.txt x
            └─ plik.txt
               1 druga tajna informacja

JS app.js > ...
1 // import modułu fs
2 const fs = require('fs');
3 // dane do zapisu
4 let data = "\nTrzecia tajna informacja";
5
6 fs.appendFile("plik.txt", data, (err) => {
7   if (err)
8     console.log(err);
9   else {
10    console.log("Zapis się udał (zawartość pliku nie została nadpisana)\n");
11    console.log("Zawartość pliku po zapisie:");
12    console.log(fs.readFileSync("plik.txt", "utf8"));
13  }
14 });
```

```
plik.txt x
plik.txt
1 Druga tajna informacja
2 Trzecia tajna informacja
```

```
PS D:\> node app.js
Zapis się udał (zawartość pliku nie została nadpisana)

Zawartość pliku po zapisie:
Druga tajna informacja
Trzecia tajna informacja
```

zawartość pliku plik.txt
(dopisanie nowej informacji)

moduł path

odczytuje, buduje ścieżkę do pliku

```
EXPLORER  ...  JS app.js  X
MODULPATH
JS app.js
1 // import modułu path
2 const path = require('path');
3 // ścieżka do pliku
4 const filePath = '/a/b/main.js';
5 // wyciąga nazwa pliku z podanej ścieżki
6 const filename = path.basename(filePath);
7 // wyciąga nazwę katalogu z podanej ścieżki
8 const dirname = path.dirname(filePath);
9 // dokładnie bieżący katalog do ścieżki
10 const join = path.join(__dirname, filePath);
11 // sprawdza czy podana ścieżka jest względna
12 // (Path From Content Root) (zwraca false)
13 // czy bezwzględna (Absolute Path) (zwraca true)
14 const isAbsolute = path.isAbsolute(join);
15
16 console.log(`
17     |     |     |     |     |
18     |     |     |     |     |     filePath: ${filePath}
19     |     |     |     |     |     filename:  ${filename}
20     |     |     |     |     |     dirname:  ${dirname}
21     |     |     |     |     |     join:     ${join}
22     |     |     |     |     |     isAbsolute:  ${isAbsolute}
`);
```

```
PS D:\> node app.js
```

```
filePath: /a/b/main.js
filename: main.js
dirname: /a/b
join: D:\Node.js\programy\modulPath\a\b\main.js
isAbsolute: true
```

własny moduł polaFigur

kod modułu polaFigur piszemy w pliku index.js w folderze polaFigur (nazwa modułu jest nazwą folderu)



```
EXPLORER  ...  JS index.js  X
└─ MODULWLASNY
  └─ polaFigur
     └─ JS index.js
        └─ JS app.js

polaFigur > JS index.js > ...
1  const wzory = [
2  { figura: "kwadrat" , wzor: "p=a^2" },
3  { figura: "prostokąt" , wzor: "p=a*b" },
4  { figura: "trójkąt" , wzor: "p=a*h/2" },
5  ];
6  // export metod modułu polaFigur
7  module.exports = {
8  ...
9  podajWzory(){
10     console.log("\nWzory na pola figur:\n");
11     for (let i = 0; i < wzory.length; i++) {
12         const item = wzory[i];
13         console.log(item.figura + ": " + item.wzor);
14     }
15     console.log("\n");
16   },
17   obliczPoleKwadratu(a){
18     console.log("pole kadratu o boku " + a + " wynosi " + a*a);
19   }
}
```

tablica obiektów z wzorami na pola

wypisuje wzory na pola

oblicza pole kwadratu o podanym boku

moduł polaFigur ma 2 metody: podajWzory oraz obliczPoleKwadratu

własny moduł polaFigur

```
EXPLORER  ...  JS app.js  X
└─ MODULWLASNY
  └─ polaFigur
     JS index.js
     JS app.js
└─ JS app.js > ...
   1 // import modułu polaFigur
   2 const wzory = require('./polaFigur');
   3
   4 // wywołanie metod modułu polaFigur
   5 wzory.podajWzory();
   6 wzory.obliczPoleKwadratu(5);
```

```
PS D:\modulWlasny> node app.js
```

```
Wzory na pola figur:
```

```
kwadrat:  $p=a^2$ 
```

```
prostokąt:  $p=a*b$ 
```

```
trójkąt:  $p=a*h/2$ 
```

```
pole kwadratu o boku 5 wynosi 25
```

w pliku app.js importujemy własny moduł i korzystamy z niego

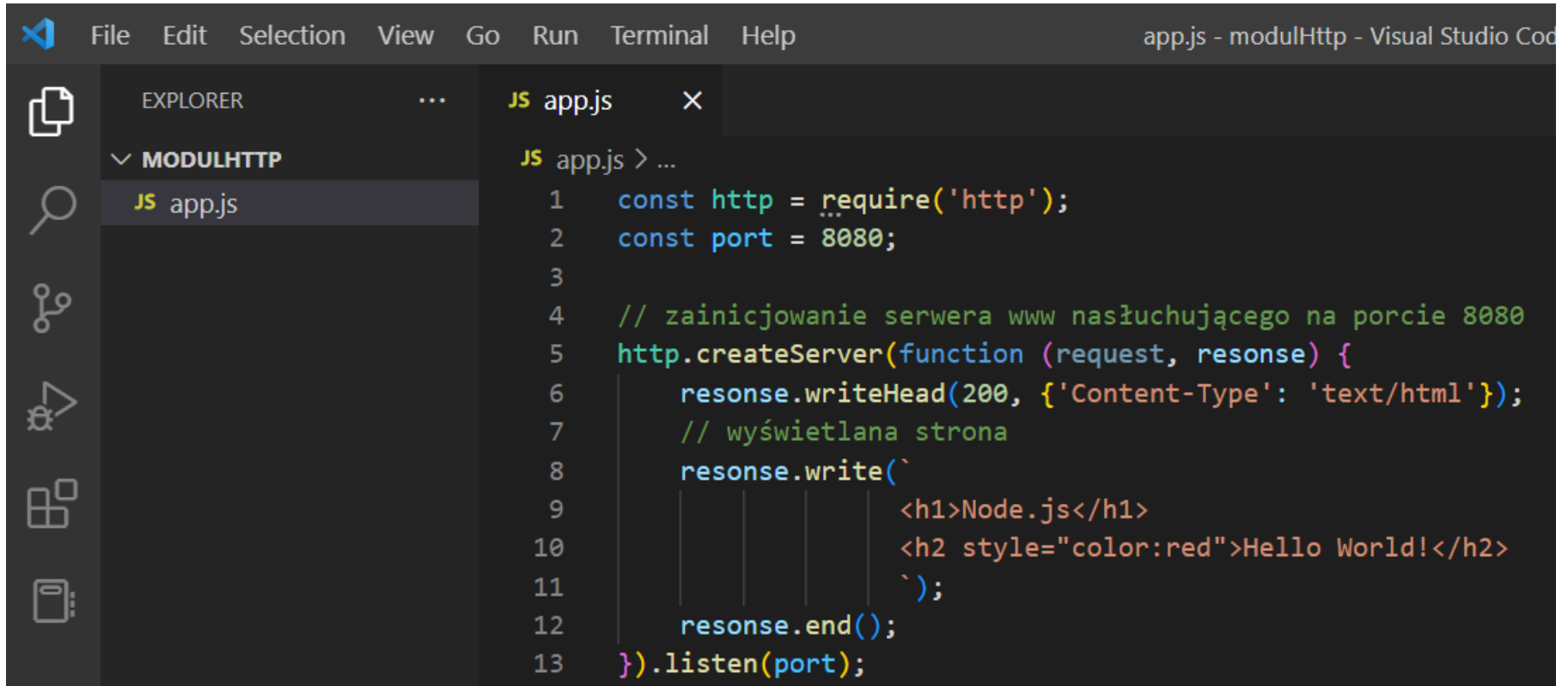
metoda podajWzory wypisuje wzory na pola figur

metoda obliczPoleKwadratu(bok) oblicza pole kwadratu o podanym boku

prosty serwer http

moduł http

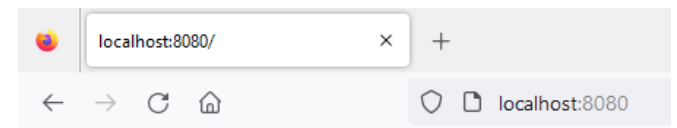
serwer http



```
File Edit Selection View Go Run Terminal Help app.js - modulHttp - Visual Studio Cod
EXPLORER
MODULHTTP
JS app.js
JS app.js > ...
1 const http = require('http');
2 const port = 8080;
3
4 // zainicjowanie serwera www nasłuchującego na porcie 8080
5 http.createServer(function (request, response) {
6     response.writeHead(200, {'Content-Type': 'text/html'});
7     // wyświetlana strona
8     response.write(`
9         <h1>Node.js</h1>
10        <h2 style="color:red">Hello World!</h2>
11    `);
12    response.end();
13 }).listen(port);
```

uruchomienie serwera www

```
node app.js
```



Node.js

Hello World!

https://www.w3schools.com/nodejs/ref_http.asp

moduł https

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

aby uruchomić serwer https należy zaimportować moduł https
(zamiast http)

https://www.w3schools.com/nodejs/ref_https.asp

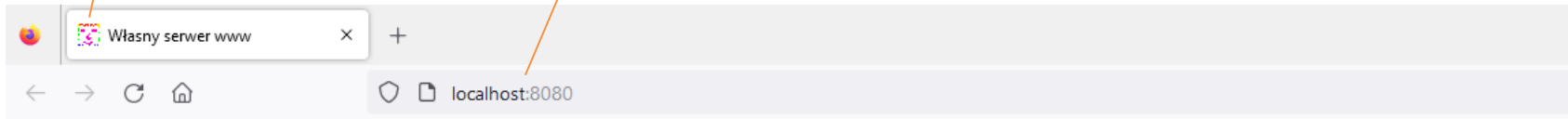
moduł http

serwer http

serwer http wyświetlający pliki strony

<https://favicon.io/favicon.ico>

serwer nasłuchuje lokalnie na porcie 8080



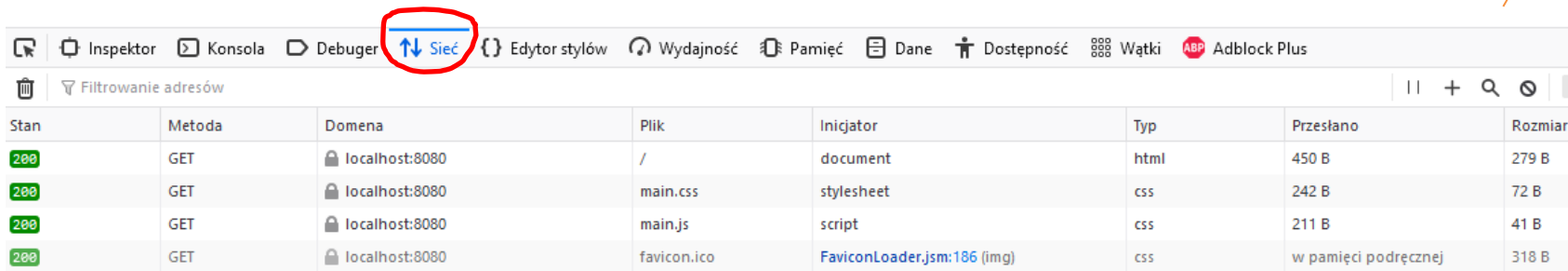
nawet działa

wynik działania skryptu w pliku main.js

Własny serwer www

styl umieszczony w odrębnym pliku main.css

Developer Tools F12



Stan	Metoda	Domena	Plik	Inicjator	Typ	Przesłano	Rozmiar
200	GET	localhost:8080	/	document	html	450 B	279 B
200	GET	localhost:8080	main.css	stylesheet	css	242 B	72 B
200	GET	localhost:8080	main.js	script	css	211 B	41 B
200	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (img)	css	w pamięci podręcznej	318 B

index.html
main.css
main.js
favicon.ico

żądania GET przeglądarki dotyczące poszczególnych plików na które składa się strona

```
PS D:\modulHttp2> node app.js
serwer nasłuchuje pod adresem localhost, na porcie 8080
OK- 200 ./favicon.ico was send to bowser
OK- 200 ./index.html was send to bowser
OK- 200 ./main.css was send to bowser
OK- 200 ./main.js was send to bowser
```

logi serwera

moduł http

```
EXPLORER
MODUL...
JS app.js
★ favicon.ico
<> index.html
# main.css
JS main.js

JS app.js > answer > fs.readFile() callback
1 // import modułów
2 const http = require("http");
3 const path = require("path");
4 const fs = require("fs");
5
6 // zmienne określające adres i port serwera
7 const host = 'localhost';
8 const port = '8080';
9
10 // ścieżki do plików strony
11 const main = "./index.html";
12 const css = "./main.css";
13 const js = "./main.js";
14 const favicon = "./favicon.ico";
15
16 // funkcja odpowiadająca na żądania wysłania poszczególnych plików tworzących stronę
17 function answer(req, res){
18     // przełącznik żądań (request)
19     switch (req.url){
20         // gdy żądanie dotyczy strony głównej wyślemy plik index.html
21         case '/':
22             fs.readFile(main, (error, dane) =>{
23                 if(error){
24                     res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
25                     console.log(`error - 404 page ${main} not found`);
26                 }else{
27                     res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8'});
28                     res.end(dane);
29                     console.log(`OK- 200 ${main} was send to browser`);
30                 }
31             });
32             break;
33         // gdy żądanie dotyczy arkusza css wyślemy plik main.css
34         case '/main.css':
35             fs.readFile(css, (error, dane) =>{
36                 if(error){
37                     res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
38                     console.log(`error - 404 page ${css} not found`);
39                 }else{
40                     res.writeHead(200, { 'Content-Type': 'text/css; charset=utf-8'});
41                     res.end(dane);
42                     console.log(`OK- 200 ${css} was send to browser`);
43                 }
44             });
45             break;
```

1/2 app.js

wysłanie nagłówka

wysłanie pliku index.html

obsługa błędów – gdy nie będzie pliku index.html pojawi się strona z błędem 404

gdy plik istnieje zostanie wysłany nagłówek 200 (OK) oraz plik index.html

moduł http

2/2 app.js

```
46 // gdy żądanie dotyczy skryptu js wysyłamy plik main.js
47 case '/main.js':
48     fs.readFile(js, (error, dane) =>{
49         if(error){
50             res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
51             console.log(`error - 404 page ${js} not found`);
52         }else{
53             res.writeHead(200, { 'Content-Type': 'text/javascript; charset=utf-8'});
54             res.end(dane);
55             console.log(`OK- 200 ${js} was send to browser`);
56         }
57     });
58     break;
59 // gdy żądanie dotyczy pliku favicon wysyłamy plik favicon.ico
60 case '/favicon.ico':
61     fs.readFile(favicon, (error, dane) =>{
62         if(error){
63             res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
64             console.log(`error - 404 page ${favicon} not found`);
65         }else{
66             res.writeHead(200, { 'Content-Type': 'image/x-icon; charset=utf-8'});
67             res.end(dane);
68             console.log(`OK- 200 ${favicon} was send to browser`);
69         }
70     });
71     break;
72 }
73 }
74
75 // tworzymy serwer http
76 const myServer = http.createServer(answer);
77 myServer.listen(port,host, () => console.log(`serwer nasłuchuje pod adresem ${host}, na porcie ${port}`));
```

start serwera

moduł http

```
EXPLORER  ...  <> index.html X
MODULHTTP2
JS app.js
★ favicon.ico
<> index.html
# main.css
JS main.js

<> index.html > ...
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset='utf-8'>
5      <title>Własny serwer www</title>
6      <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
7      <script src='main.js'></script>
8  </head>
9  <body>
10     <h1>Własny serwer www</h1>
11 </body>
12 </html>
```

index.html

```
EXPLORER  ...  # main.css X
MODULHTTP2
JS app.js
★ favicon.ico
<> index.html
# main.css
JS main.js

# main.css > ...
1  h1{
2      background-color: blue;
3      color: white;
4      width: 300px;
5  }
```

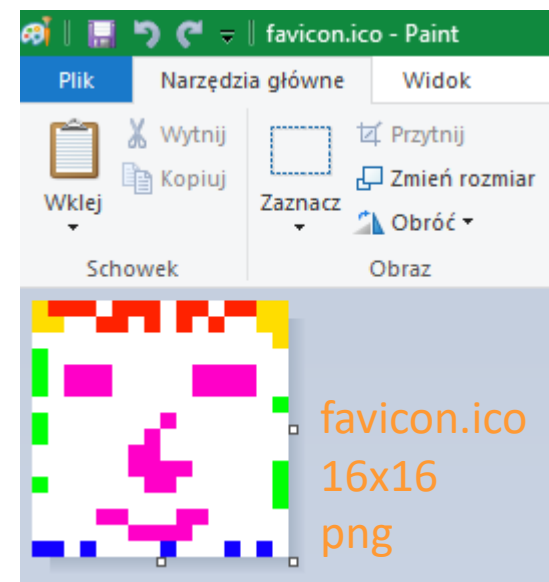
main.css

```
EXPLORER  ...  JS main.js X
MODULHTTP2
JS app.js
★ favicon.ico
<> index.html
# main.css
JS main.js

JS main.js
1  document.write("<p>nawet działa</p>");
2
3
4
5
6
```

main.js

<https://favicon.io/>



prosty serwer http

Express serwer http

framework webowy

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

instalacja frameworka



Hello World Express!

Inspector | Konsola | Debugger | **Sieć** | Edytor stylów | Wydajność | Pamięć | Dane | Dostępność | Wątki | Adblock Plus

Filtrowanie adresów

Stan	Metoda	Domena	Plik	Inicjator	Typ	Przesłano	Rozmiar
304	GET	localhost:8080	/	document	html	w pamięci podręcznej	29 B
404	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (img)	html	w pamięci podręcznej	150 B

serwer http w oparciu o framework Express

Developer
Tools F12

prosty serwer http

Express serwer http

framework

```
EXPLORER
EXPRES...
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json

JS app.js
1 // import frameworka express
2 const express = require('express');
3
4 // wywołanie frameworka
5 const app = express();
6 const port = 8080;
7
8 // obsługa żądania GET strony głównej
9 app.get('/', (req, res) => {
10   res.send('<h1>Hello World Express!</h1>');
11 });
12
13 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
14 app.listen(port, () => {
15   console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
16 });
```

kod serwera http w oparciu o framework Express j

nagłówek jest ustawiany automatycznie w zależności od wysyłanych danych (nie trzeba go opisywać)

pliki konfiguracyjne frameworka Express są generowane po jego instalacji

uruchomienie skryptu

```
PS D:\expressHttp> node app.js
serwer nasłuchuje pod adresem: http://localhost:8080
```

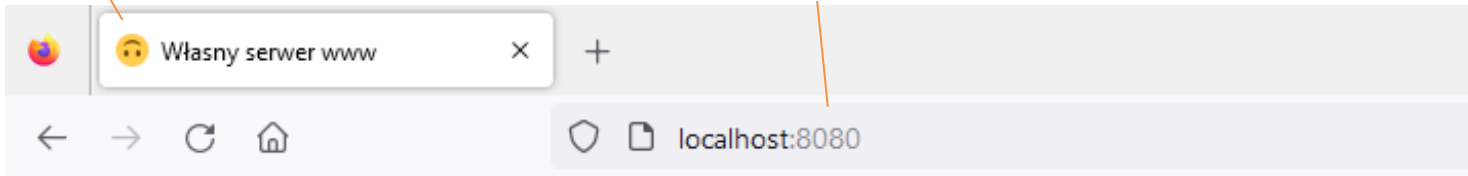
serwer http wyświetlający pliki strony

Express serwer http

framework

favicon.ico <https://favicon.io/>

serwer nasłuchuje lokalnie na porcie 8080



nawet działa

wynik działania skryptu w pliku main.js

**Własny serwer www
oparty na Node.js i
Express**

styl umieszczony w odrębnym pliku main.css

żądania GET przeglądarki dotyczące poszczególnych plików na które składa się strona

A screenshot of the browser's developer tools, specifically the Network tab. The 'Sieć' (Network) icon is circled in red. Below the toolbar is a table of network requests.

Stan	Metoda	Domena	Plik	Inicjator	Typ	Przesłano	Rozmiar
200	GET	localhost:8080	/	document	html	629 B	313 B
200	GET	localhost:8080	main.css	stylesheet	css	397 B	84 B
200	GET	localhost:8080	main.js	script	js	384 B	57 B
200	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (img)	x-icon	15,71 kB	15,41 kB

Developer
Tools F12

index.html

main.css

main.js

favicon.ico

Express serwer http

```
EXPLOSER    ...    JS app.js    X
EXPRESSHTTP2
  > node_modules
  < serwerLogs
    JS index.js
  < strona
    ★ favicon.ico
    < index.html
    # main.css
    JS main.js
  JS app.js
  {} package-lock.json
  {} package.json

JS app.js > ...
1 // import frameworka express
2 const express = require('express');
3 // import własnego modułu wyświetlającego logi
4 const serwerLogs = require('./serwerLogs');
5
6 // wywołanie frameworka
7 const app = express();
8 // port, na którym serwer nasłuchuje
9 const port = 8080;
10
11 // ścieżki do plików strony
12 // muszą być bezwzględne
13 // lub trzeba używać root
14 // w którym podajemy ścieżkę bezwzględną
15 // do folderu z plikami strony np.:
16 // res.sendFile(main, {root: ...})
17 const main = 'index.html';
18 const css = 'main.css';
19 const js = 'main.js';
20 const image = 'favicon.ico';
21
22
23 // obsługa żądania GET strony głównej
24 app.get('/', (req, res) => {
25   // w root jest podana ścieżka bezwzględna do katalogu
26   // z plikami strony
27   res.sendFile(main, { root: __dirname + '\\strona'});
28   // wyświetlenie logów za pomocą własnego modułu
29   serwerLogs.show(main, req);
30 });
31
32 // obsługa żądania GET pliku main.css
33 app.get('/main.css', (req, res) => {
34   res.sendFile(css, { root: __dirname + '\\strona'});
35   serwerLogs.show(css, req);
36 });
37
38 // obsługa żądania GET pliku main.js
39 app.get('/main.js', (req, res) => {
40   res.sendFile(js, { root: __dirname + '\\strona'});
41   serwerLogs.show(js, req);
42 });
43
44 // obsługa żądania GET pliku favicon.ico
45 app.get('/favicon.ico', (req, res) => {
46   res.sendFile(image, { root: __dirname + '\\strona'});
47   serwerLogs.show(image, req);
48 });
49
50 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
51 app.listen(port, () => {
52   console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
53 });
54
55
```

app.js

logi serwera

obsługa żądań GET
poszczególnych plików
strony – wysłanie
żądanej strony
do przeglądarki

Express serwer http

EXPLORER

- EXPRESSHTTP2
 - node_modules
 - serwerLogs
 - JS index.js
 - strona
 - favicon.ico
 - index.html
 - main.css
 - main.js
 - app.js
 - package-lock.json
 - package.json

```
index.html
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8'>
5   <title>Własny serwer www</title>
6   <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
7   <script src='main.js'></script>
8 </head>
9 <body>
10  <h1>Własny serwer www oparty na Node.js i Express</h1>
11 </body>
12 </html>
13
```

EXPLORER

- EXPRES...
 - node_modules
 - serwerLogs
 - JS index.js
 - strona
 - favicon.ico
 - index.html
 - main.css

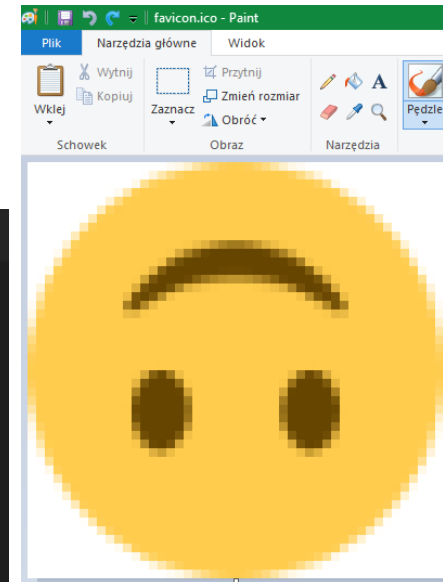
```
main.css
1 h1{
2   background-color: blue;
3   color: white;
4   width: 300px;
5 }
```

EXPLORER

- EXPRES...
 - node_modules
 - serwerLogs
 - JS index.js
 - strona
 - favicon.ico
 - index.html
 - main.css
 - main.js

```
main.js
1 document.write("<p>nawet działa</p>");
2
3
4
5
6
7
8
9
```

<https://favicon.io/>
favicon.ico
48x48



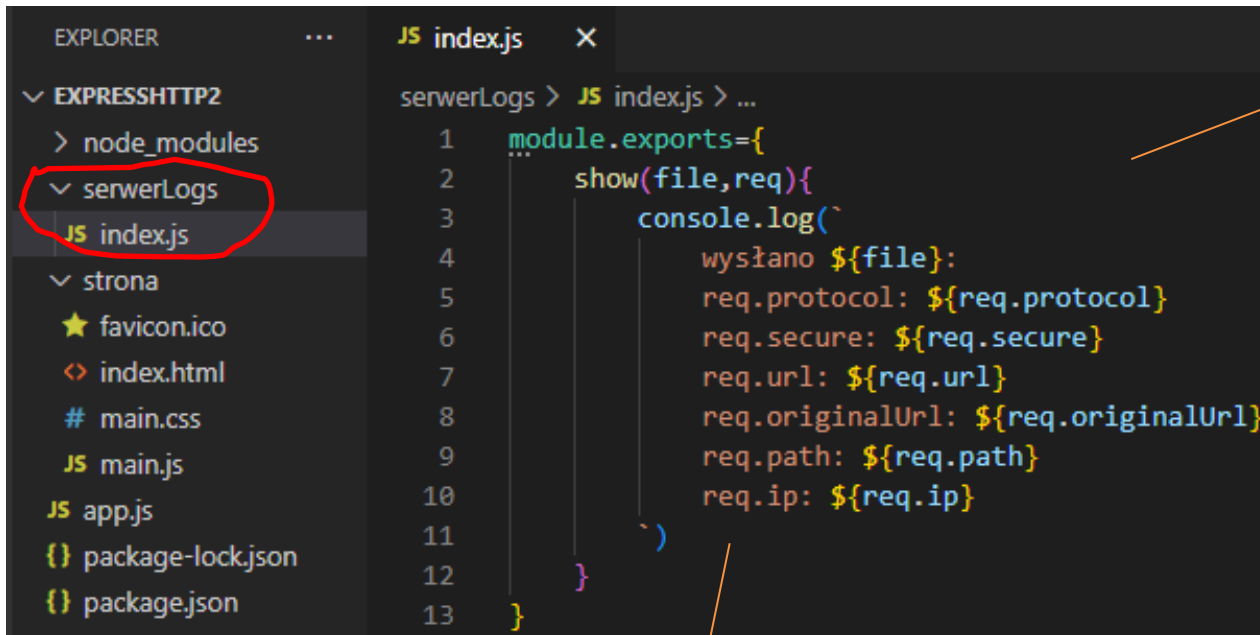
index.html

main.css

main.js

Express serwer http

własny moduł wyświetlający logi serwera znajduje się w katalogu serwerLogs



```
1  module.exports={
2    ...
3    show(file,req){
4      console.log(`
5        wysłano ${file}:
6        req.protocol: ${req.protocol}
7        req.secure: ${req.secure}
8        req.url: ${req.url}
9        req.originalUrl: ${req.originalUrl}
10       req.path: ${req.path}
11       req.ip: ${req.ip}
12     `)
13   }
}
```

kod znajduje się
w pliku index.js

metoda show
wyświetlająca logi
serwera

Express serwer http

logi serwera
wygenerowane za
pomocą własnego
modułu
serverLogs

```
PS D:\arch\Technik Programista\aplikacjeWebowe\node.js\programy\expressHttp2> node app.js  
serwer nasłuchuje pod adresem: http://localhost:8080
```

```
wysłano index.html:  
req.protocol: http  
req.secure: false  
req.url: /  
req.originalUrl: /  
req.path: /  
req.ip: ::ffff:127.0.0.1
```

wysłany plik
protokół
czy użyto https

ścieżka adresu URL, polecenie Node.js

```
wysłano main.css:  
req.protocol: http  
req.secure: false  
req.url: /main.css  
req.originalUrl: /main.css  
req.path: /main.css  
req.ip: ::ffff:127.0.0.1
```

ścieżka adresu
URL, w przypadku
przekierowania
zachowuje
oryginalną ścieżkę

```
wysłano main.js:  
req.protocol: http  
req.secure: false  
req.url: /main.js  
req.originalUrl: /main.js  
req.path: /main.js  
req.ip: ::ffff:127.0.0.1
```

ścieżka adresu
URL, zawiera
ostatnią część
adresu URL

```
wysłano favicon.ico:  
req.protocol: http  
req.secure: false  
req.url: /favicon.ico  
req.originalUrl: /favicon.ico  
req.path: /favicon.ico  
req.ip: ::ffff:127.0.0.1
```

adres IP klienta
(IPv6 oraz IPv4)

Google wyszukiwanie informacji

początek definicji parametrów

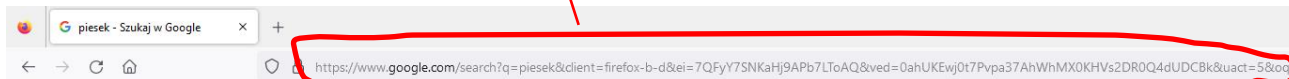
adres URL

nazwa parametru

wartość parametru

separator

https://www.google.com/search?q=piesek&client=firefox-b-d&ei=7QFyY7SNKaHj9APb7LToAQ&ved=0ahUKEwj0t7Pvpa37AhWhMX0KHVs2DR0Q4dUDCBk&uact=5&oq=piesek&gs_lcp=Cgxnd3Mtd2l6LXNlcnAQAziKCAAQsQMqgWEQQzIOCC4QgAAQsQMqgWEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAEELEDEIMBMggiABCxAXCDATILCC4QgAAQsQMqgWEyCwgAEIAEELEDEIMBMgsIABCABBcAxCDATIFCC4QgAAQ6CggAEecQ1gQQsAM6BwgAELADEEM6DAguEMgDELADEEMYAToFCAAQgAAQ6EAgUELEDEIMBEMcBENEDEEM6CAguELEDEIMBOgQIABBD0goILhDHARDRAXBDOgQILhBDOggILhCABBDUAjoKCC4QsQMqgWEQQzoHCC4Q1AIQQzoQCC4QgWEQ1AIQsQMqgAAQQzoHCAAQsQMqgzoLCC4QgAAQsQMqg1AI6DQguELEDEIMBENQCEENKBAhBGABKBAhGGAfQ_A1YzxZgpBtoAnABeACAAY4CiAGTDJIBAZItNpgBAKABAcgBEcABAAdoBBggBEAEYCA&sclient=gws-wiz-serp



po wyszukaniu słowa „piesek” wyszukiwarka przesyła do serwera google dane (nazwy parametrów oraz ich wartości są doklejane do adresu)

Express przesyłanie informacji

początek definicji parametrów

adres URL

nazwa parametru

wartość parametru

separator

`https://www.google.com/search?q=piesek&client=firefox-b-d&ei=7QFyY7LToAQ`

```
// obsługa żądania GET strony głównej  
app.get('/', (req, res) => {
```

```
  // pobieramy przesłane zmienne  
  const q = req.query.q;  
  const client = req.query.client;
```

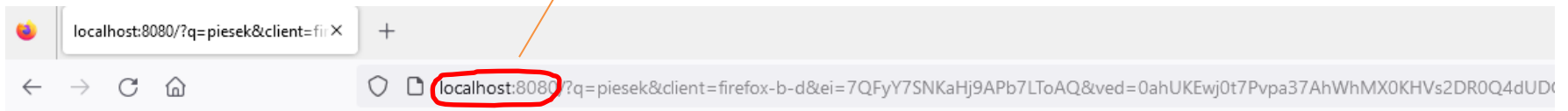
```
  // drugi sposób - destrukuryzacja  
  const { q, client } = req.query;
```

parametry i wartości
przesłane w adresie stają
się kluczami oraz
wartościami obiektu query

Express przesyłanie informacji

na podstawie adresu wygenerowanego przy wyszukiwaniu słowa „piesek” wyświetlamy wszystkie parametry i ich wartości na stronie lokalnie

adres google zamieniamy na localhost:8080



Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":

```
q: piesek,  
client: firefox-b-d,  
ei: 7QFyY7SNKaHj9APb7LToAQ,  
wed: undefined,  
uact: 5,  
oq: piesek,  
gs_lcp:  
Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMqgwEQzIOCC4QgAAQsQMqgwEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAELEEDEIMB!  
ved: 0ahUKEwj0t7Pvpa37AhWhMX0KHVs2DR0Q4dUDCBk,  
sclient: gws-wiz-serp
```

Express przesyłanie informacji

```
EXPLORER  ...  JS app.js  X
└─ PRZESYŁANIEZMIENNYCH
  └─ node_modules
    └─ JS app.js
      └─ {} package-lock.json
      └─ {} package.json

JS app.js > ...
1 // import frameworka express
2 const express = require('express');
3
4 // wywołanie frameworka
5 const app = express();
6 const port = 8080;
7
8 // obsługa żądania GET strony głównej
9 app.get('/', (req, res) => {
10
11     // pobieramy dane z adresu, który wyszukiwarka przesłała na serwer
12     const { q, client, ei, wed, uact, oq, gs_lcp, ved, sclient} = req.query;
13     res.send(`
14         <h1>Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":</h1>
15         q: ${q},<br>
16         client: ${client},<br>
17         ei: ${ei},<br>
18         wed: ${wed},<br>
19         uact: ${uact},<br>
20         oq: ${oq},<br>
21         gs_lcp: ${gs_lcp},<br>
22         ved: ${ved},<br>
23         sclient: ${sclient}
24     `);
25 });
26
27 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
28 app.listen(port, () => {
29     console.log(`serwer nasłuchuje pod adresem: http://localhost:\${port}`);
30 });
```

Express przesyłanie informacji

inny sposób dołączenia parametrów do adresu

wartości parametrów q oraz client

http://localhost:8080/piesek/firefox-b-d



Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":

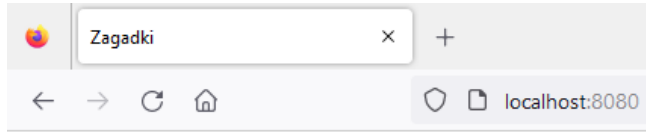
q: piesek,
client: firefox-b-d,

```
EXPLORER  JS app.js  X
PRZESYŁANIEPARAMETRO...
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json
JS app.js > ...
1 // import frameworka express
2 const express = require('express');
3
4 // wywołanie frameworka
5 const app = express();
6 const port = 8080;
7
8 // obsługa żądania GET strony głównej
9 app.get('/:q/:client', (req, res) => {
10
11     // pobieramy dane z adresu, który wyszukiwarka przesłała na serwer
12     const q = req.params.q;
13     // alternatywnie
14     const client = req.params['client'];
15     res.send(`
16         <h1>Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":</h1>
17         q: ${q},<br>
18         client: ${client},<br>
19     `);
20 });
21
22 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
23 app.listen(port, () => {
24     console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
25 });
```

utwórz stronę z zagadkami

zadanie

formularz przesyła dane metodą get



Zagadki

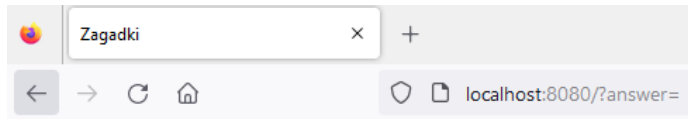
zagadka

stolica Czech ?

Praga

sprawdź

od początku



Zagadki

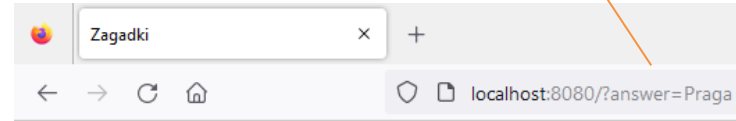
Pytania się skończyły

wpisz odpowiedź

sprawdź

od początku

komunikat końcowy
gdy pytania się
wyczerpią



Zagadki

3! = ?

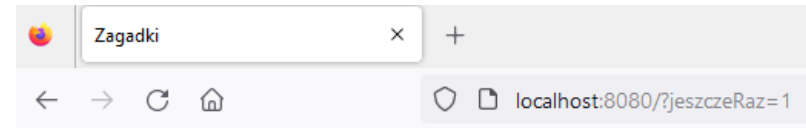
wpisz odpowiedź

sprawdź

dobrze

od początku

weryfikacja odpowiedzi –
gdy użytkownik zgadnie
wyświetla się następną
zagadka



Zagadki

stolica Czech ?

wpisz odpowiedź

sprawdź

od początku

po naciśnięciu przycisku
„od początku” pytania zaczynają
się od nowa

odpowieź do zadania

```
EXPLORER  ...  JS app.js  X  1/3
└─ FORMZAGADKA
  └─ > node_modules
    JS app.js
    {} package-lock.json
    {} package.json

JS app.js > app.get('/') callback
1 // import frameworka express
2 const express = require('express');
3 // wywołanie frameworka
4 const app = express();
5 // port, na którym serwer nasłuchuje
6 const port = 8080;
7 // pytania
8 const pytania = ["stolica Czech ?",
9                 "3! = ?",
10                "12 * 5 = ?",
11                "Pytania się skończyły"];
12 // odpowiedzi
13 const odpowiedzi = ["Praga", "6", "60"];
14 // index tablic
15 let i = 0;
16 // flaga wyczerpania limitu pytań
17 let jestKoniec = false;
```



zmienne wykorzystane na stronie

odpowieź do zadania

```
19 // obsługa żądania GET strony głównej
20 app.get('/', (req, res) => {
21     // zmienna do wypisania wyniku na stronie
22     wynik = "";
23
24     // pobieramy odpowiedź wysłaną z formularza
25     // za pomocą metody get
26     const answer = req.query.answer;
27
28     // pobieramy zmienną jeszczeRaz wysłaną
29     // przyciskiem "od początku"
30     const jeszczeRaz= req.query.jeszczeRaz;
31
32     // sprawdzamy odpowiedź użytkownika
33     if (answer != "" && answer != undefined && !jestKoniec){
34         if (answer == odpowiedzi[i]){
35             wynik = "dobrze";
36             i++;
37             // limit pytań się wyczerpał
38             if(i==odpowiedzi.length){
39                 //i = 0;
40                 // komunikat końcowy
41                 wynik = "koniec!";
42                 // ustawiamy flagę zakończenia
43                 jestKoniec = true;
44             }
45         }else{
46             wynik = "źle";
47         }
48     }
49     // gdy pula pytań się wyczerpie
50     // można nacisnąć przycisk "od początku"
51     // do serwera zostanie wysłana zmienna
52     // jeszczeRaz = 1
53     // resetujemy ustawienia
54     if (jeszczeRaz == "1"){
55         jestKoniec = false;
56         i=0;
57     }
```

2/3

pobieranie
zmiennych wysłanych
przez formularze na
stronie

sprawdzenie
odpowiedzi

obsługa przycisku
„od początku”

odpowiedź do zadania

```
59 // strona główna
60 strona = `
61     <!doctype html>
62 <html>
63 <head>
64     <meta charset="utf-8">
65     <title>Zagadki</title>
66 </head>
67 <body>
68 <h1>Zagadki</h1>
69     `${pytania[i]}
70     <form name="zagadka" method="get">
71         <input type="text" name="answer" placeholder="wpisz odpowiedź">
72         <input type="submit" value="sprawdź">
73     </form>
74     `${wynik}
75     <form name="odPoczatku" method="get">
76         <input type="hidden" name="jeszczeRaz" value="1">
77         <input type="submit" value="od początku">
78     </form>
79 </body>
80 </html>
81 `
82 res.send(strona);
83 });
84
85
86 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
87 app.listen(port, () => {
88     console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
89 });
```

3/3

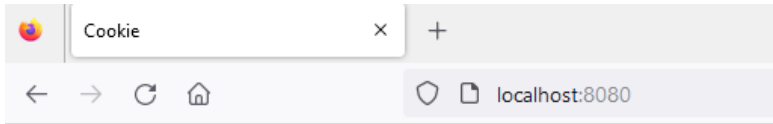
wypisanie pytania na stronie głównej

weryfikacja odpowiedzi na stronie głównej

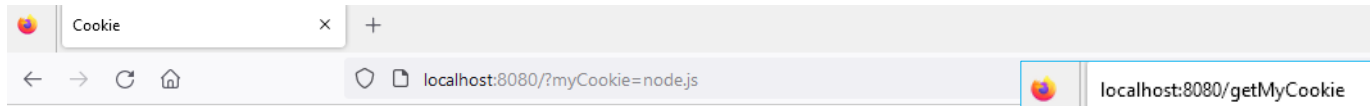
wysłanie strony do przeglądarki

cookie

formularz do zapisywania pliku cookie



Wpisz zawartość pliku cookie

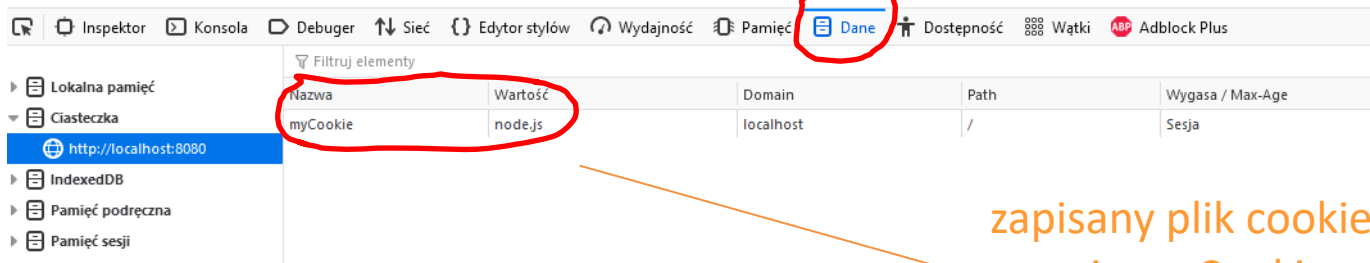
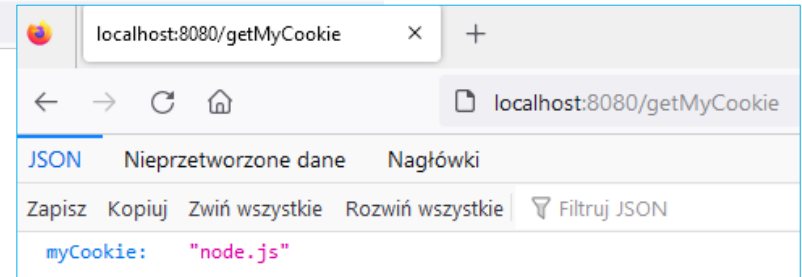


Wpisz zawartość pliku cookie

Twoja informacja została zapisana w pliku cookie o nazwie myCookie

[pokaż plik cookie](#)

[usuń plik cookie](#)



zapisany plik cookie o nazwie myCookie z zawartością z formularza

cookie

```
npm install express cookie-parser
```

instalacja
modułu
cookie-parser

```
// import parsera cookie  
const cookieParser = require('cookie-parser');  
// uruchomienie cookie parsera  
app.use(cookieParser());
```

użycie modułu
cookie-parser
w kodzie

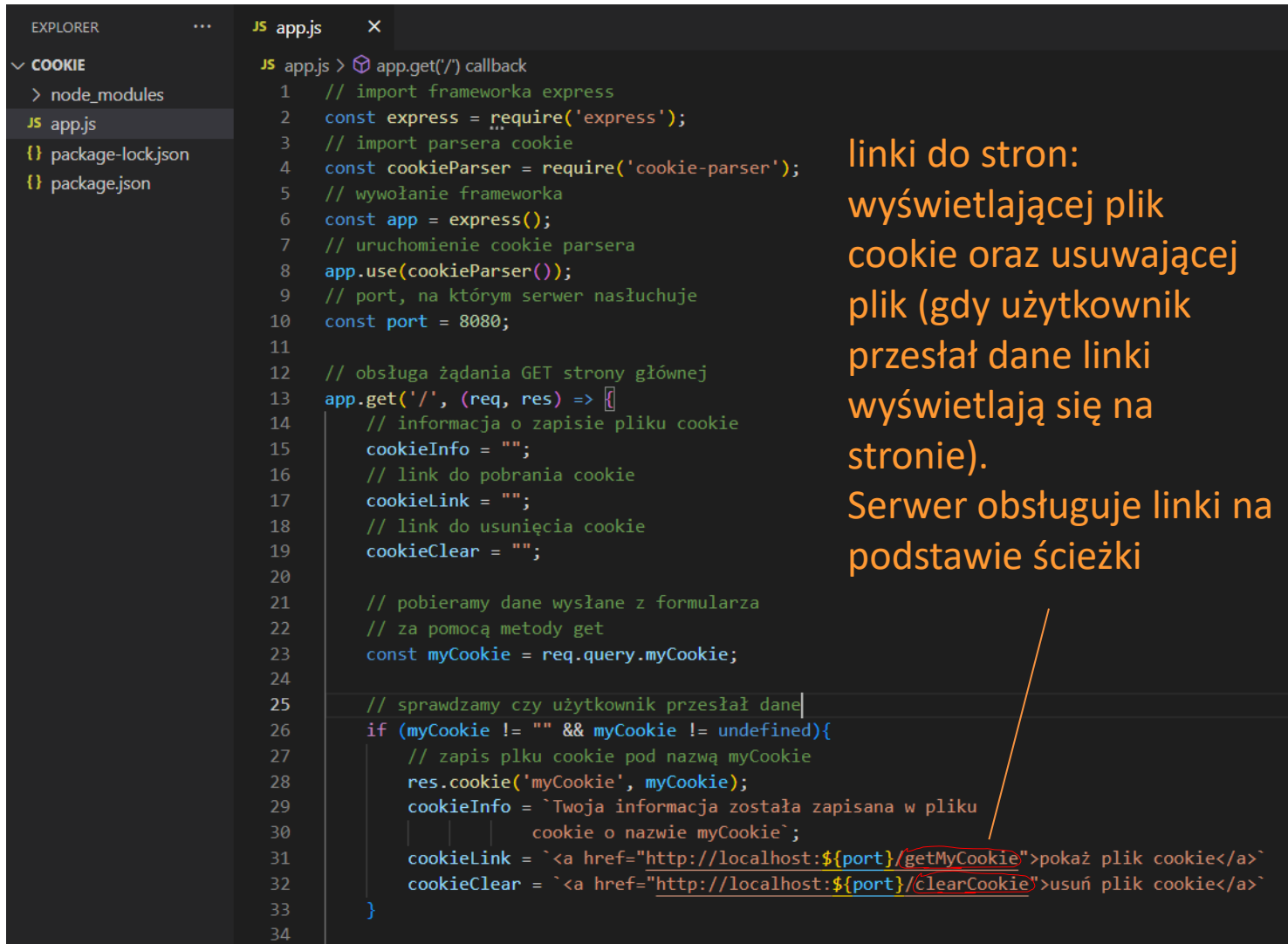
```
// zapis pliku cookie o wartości 'myCookie' pod nazwą myCookie  
res.cookie('myCookie', myCookie);
```

```
// pobiera wartość pliku cookie  
myCookie = req.cookies.myCookie;
```

```
// usuwamy plik cookie  
res.clearCookie('myCookie');
```

obsługa
plików cookie

cookie



```
EXPLORER
...
JS app.js x
  COOKIE
    node_modules
  JS app.js
  package-lock.json
  package.json

JS app.js > app.get('/') callback
1 // import frameworka express
2 const express = require('express');
3 // import parsera cookie
4 const cookieParser = require('cookie-parser');
5 // wywołanie frameworka
6 const app = express();
7 // uruchomienie cookie parsera
8 app.use(cookieParser());
9 // port, na którym serwer nasłuchuje
10 const port = 8080;
11
12 // obsługa żądania GET strony głównej
13 app.get('/', (req, res) => {
14   // informacja o zapisie pliku cookie
15   cookieInfo = "";
16   // link do pobrania cookie
17   cookieLink = "";
18   // link do usunięcia cookie
19   cookieClear = "";
20
21   // pobieramy dane wysłane z formularza
22   // za pomocą metody get
23   const myCookie = req.query.myCookie;
24
25   // sprawdzamy czy użytkownik przesłał dane
26   if (myCookie != "" && myCookie != undefined){
27     // zapis plku cookie pod nazwą myCookie
28     res.cookie('myCookie', myCookie);
29     cookieInfo = `Twoja informacja została zapisana w pliku
30     |         |         |         |         |
31     |         |         |         |         | cookie o nazwie myCookie`;
32     cookieLink = `

linki do stron:  
wyświetlającej plik  
cookie oraz usuwającej  
plik \(gdy użytkownik  
przesłał dane linki  
wyświetlają się na  
stronie\).  
Serwer obsługuje linki na  
podstawie ścieżki


```

cookie

```
35 // strona główna
36 strona = `
37   <!doctype html>
38   <html>
39   <head>
40     <meta charset="utf-8">
41     <title>Cookie</title>
42   </head>
43   <body>
44     <h1>Wpisz zawartość pliku cookie</h1>
45     <form name="zagadka" method="get">
46       <input type="text" name="myCookie" placeholder="wpisz plik cookie">
47       <input type="submit" value="zapisz">
48     </form>
49     ${cookieInfo}<br>
50     ${cookieLink}<br>
51     ${cookieClear}
52   </body>
53   </html>
54   `
55   res.send(strona);
56 });
57
58 // pobierz plik cookie
59 app.get('/getMyCookie', (req, res) => {
60   // wyświetla plik cookie
61   console.log(req.cookies)
62   res.send(req.cookies);
63   // pobiera wartość pliku cookie
64   myCookie = req.cookies.myCookie;
65   console.log(myCookie);
66 });
67
68 // usuń plik cookie
69 app.get('/clearCookie', (req, res) => {
70   // usuwamy plik cookie
71   res.clearCookie('myCookie');
72   // ładujemy stronę główną
73   res.redirect('/');
74   res.end();
75   console.log("myCookie usuniety");
76 });
77
78
79 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
80 app.listen(port, () => {
81   console.log(`serwer nasłuchuje pod adresem: http://localhost:\${port}`);
82 });
```

strona główna
/

strona wyświetlająca
plik cookie, serwer
obsługuje stronę na
podstawie ścieżki
/getMyCookie

strona usuwająca
plik cookie, serwer
obsługuje stronę na
podstawie ścieżki
/clearCookie

middleware

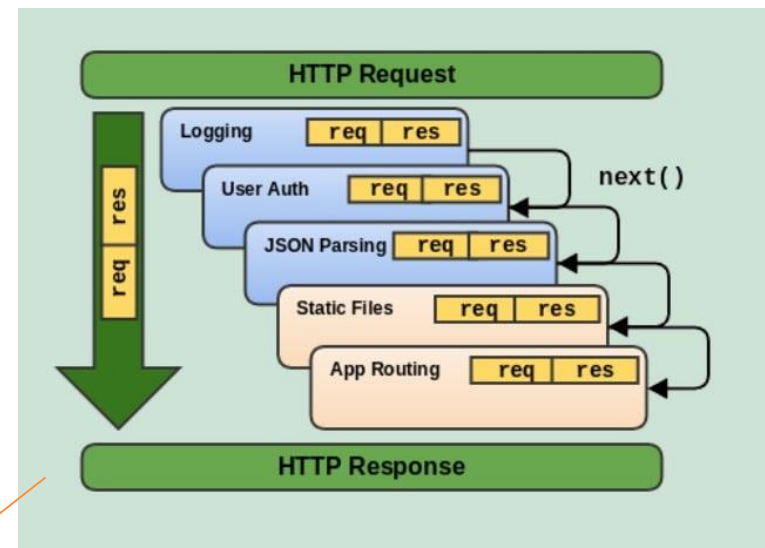
middleware to funkcje, które mają dostęp do obiektu żądania (req), obiektu odpowiedzi (res) i następnej funkcji middleware w cyklu żądanie-odpowieź aplikacji. Następna funkcja middleware jest zwykle oznaczana przez zmienną o nazwie next.

funkcja middleware może:

- wykonać dowolny kod
- wprowadzić zmiany w obiektach żądania i odpowiedzi
- zakończyć cykl żądanie-odpowieź
- wywołać następną funkcję middleware

middleware

aplikacja Express to zasadniczo seria wywołań funkcji middleware.



przed wykonaniem ostatecznej procedury obsługi żądań wykonywane są funkcje middleware

middleware

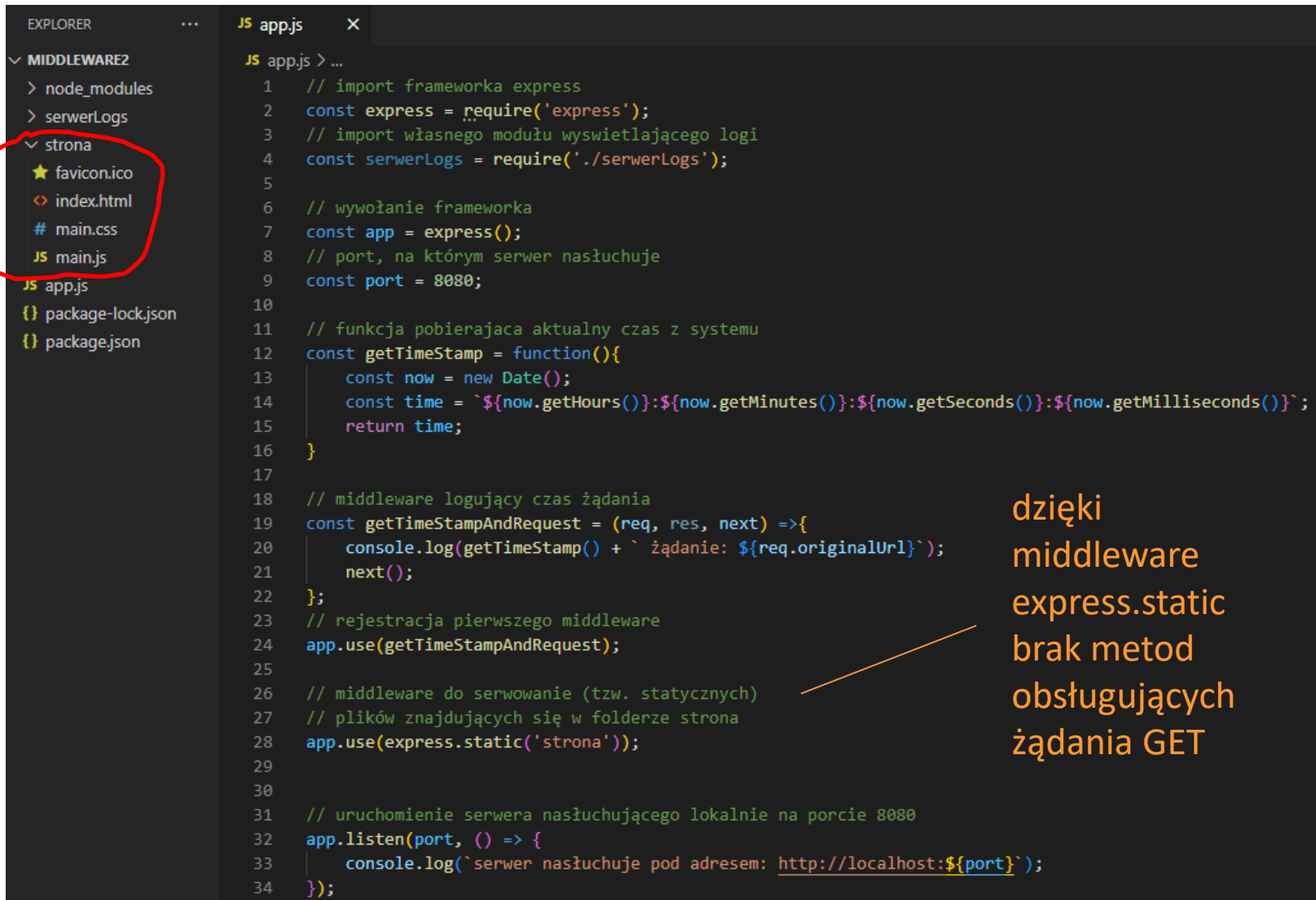
```
EXPLORER  ...  JS app.js  X
MIDDLEWARE
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json

JS app.js > ...
1 // import frameworka express
2 const express = require('express');
3
4 // wywołanie frameworka
5 const app = express();
6 const port = 8080;
7
8 // funkcja pobierająca aktualny czas z systemu
9 const getTimeStamp = function(){
10   const now = new Date();
11   const time = `${now.getHours()}:${now.getMinutes()}:${now.getSeconds()}:${now.getMilliseconds()}`;
12   return time;
13 }
14
15 // pierwszy middleware
16 const firstMiddleware = (req, res, next) =>{
17   console.log(getTimeStamp() + ` firstMiddleware żądanie: ${req.originalUrl}`);
18   next();
19 };
20 // rejestracja pierwszego middleware
21 app.use(firstMiddleware);
22
23
24 // drugi middleware
25 const secondMiddleware = (req, res, next) =>{
26   console.log(getTimeStamp() + " secondMiddleware");
27   next();
28 };
29 // rejestracja drugiego middleware
30 app.use(secondMiddleware);
31
32 // obsługa żądania GET strony głównej
33 app.get('/', (req, res) => {
34   console.log(getTimeStamp() + " beginning of get /");
35   res.send('<h1>Hello World Express!</h1>');
36 });
37
38 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
39 app.listen(port, () => {
40   console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
41 });
```

```
PS D:\middleware> node app.js
serwer nasłuchuje pod adresem: http://localhost:8080
15:39:25:488 firstMiddleware żądanie: /
15:39:25:490 secondMiddleware
15:39:25:491 beginning of get /
█
```


pliki statyczne w folderze strona

middleware static files



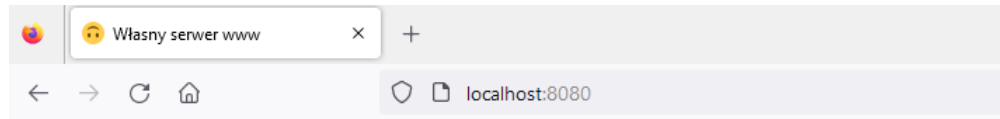
```
EXPLORER  ...  JS app.js  X

MIDDLEWARE2
├── node_modules
├── serwerLogs
└── strona
    ├── favicon.ico
    ├── index.html
    ├── main.css
    └── main.js
JS app.js
package-lock.json
package.json

JS app.js > ...
1 // import frameworka express
2 const express = require('express');
3 // import własnego modułu wyświetlającego logi
4 const serwerLogs = require('./serwerLogs');
5
6 // wywołanie frameworka
7 const app = express();
8 // port, na którym serwer nasłuchuje
9 const port = 8080;
10
11 // funkcja pobierająca aktualny czas z systemu
12 const getTimeStamp = function(){
13     const now = new Date();
14     const time = `${now.getHours()}:${now.getMinutes()}:${now.getSeconds()}:${now.getMilliseconds()}`;
15     return time;
16 }
17
18 // middleware logujący czas żądania
19 const getTimeStampAndRequest = (req, res, next) =>{
20     console.log(getTimeStamp() + ` żądanie: ${req.originalUrl}`);
21     next();
22 };
23 // rejestracja pierwszego middleware
24 app.use(getTimeStampAndRequest);
25
26 // middleware do serwowania (tzw. statycznych)
27 // plików znajdujących się w folderze strona
28 app.use(express.static('strona'));
29
30
31 // uruchomienie serwera nasłuchującego lokalnie na porcie 8080
32 app.listen(port, () => {
33     console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
34 });
```

dzięki
middleware
express.static
brak metod
obsługujących
żądania GET

middleware static files



nawet działa

**Własny serwer www
oparty na Node.js i
Express**

Stan	Metoda	Domena	Plik	Inicjator	Typ	Przesłano	Rozmiar
200	GET	localhost:8080	/	document	html	629 B	313 B
200	GET	localhost:8080	main.css	stylesheet	css	407 B	94 B
200	GET	localhost:8080	main.js	script	js	386 B	59 B
200	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:186 (img)	x-icon	15,71 kB	15,41 kB

dzięki
middleware
express.static
wszystkie pliki w
folderze strona
załadowały się
do przeglądarki

```
PS D:\middleware2> node app.js
serwer nasłuchuje pod adresem: http://localhost:8080
16:30:18:35 żądanie: /
16:30:18:216 żądanie: /main.css
16:30:18:221 żądanie: /main.js
16:30:18:446 żądanie: /favicon.ico
```

bazy danych

relacyjna (tabele powiązane ze sobą relacjami,
zapytania do bazy można wykonywać w języku SQL, przykład: Oracle, MySQL)



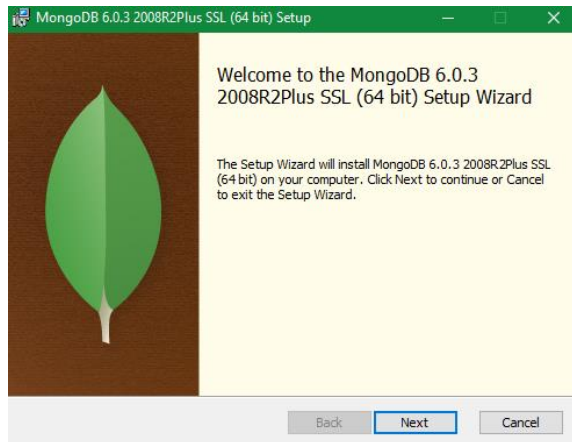
baza danych

nierelacyjna (baza zawiera kolekcje,
w których przechowywane są dokumenty, baza typu noSQL, przykład: MongoDB)

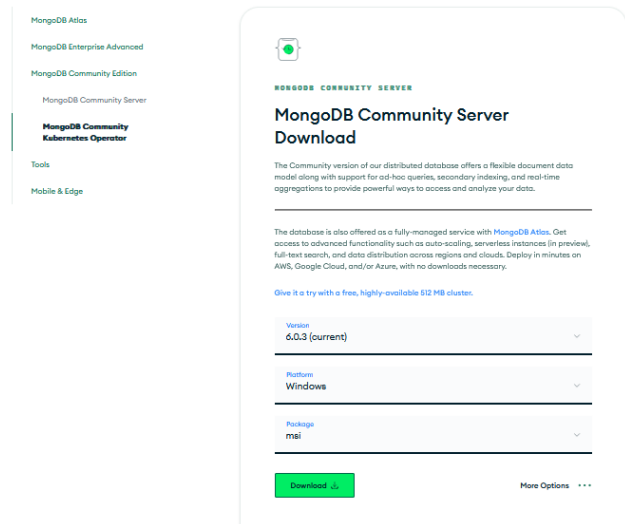
MongoDB

nierelacyjny system zarządzania bazą danych napisany w języku C++. Charakteryzuje się brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON.

<https://pl.wikipedia.org/wiki/MongoDB>

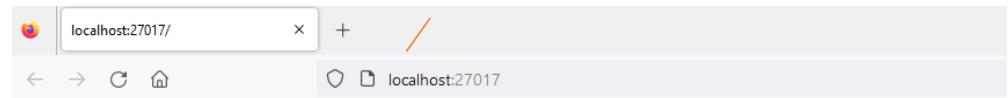


instalacja MongoDB



[download MongoDB](#)

MongoDB nasłuchuje <http://localhost:27017/>

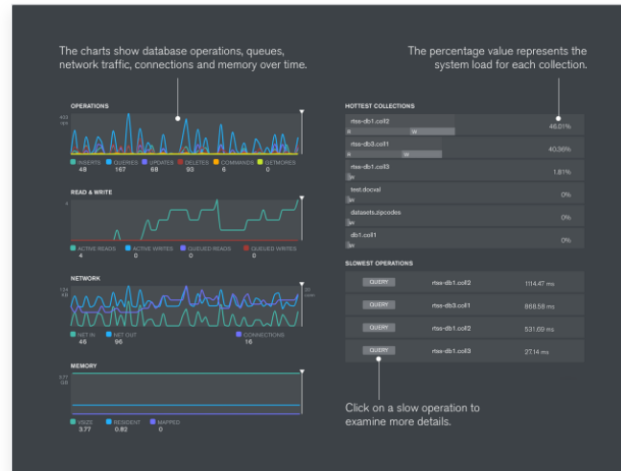


It looks like you are trying to access MongoDB over HTTP on the native driver port.

MongoDB

program do zarządzania danymi instalowany razem z bazą (odpowiednik phpMyAdmin)

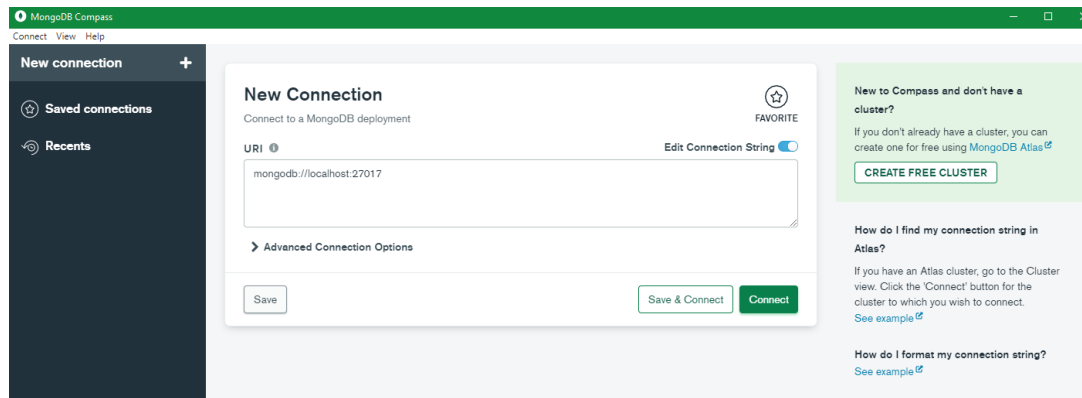
Welcome to MongoDB Compass



Performance Charts.

Real-time server statistics let you view key server metrics and database operations. Drill down into database operations easily and understand your most active collections.

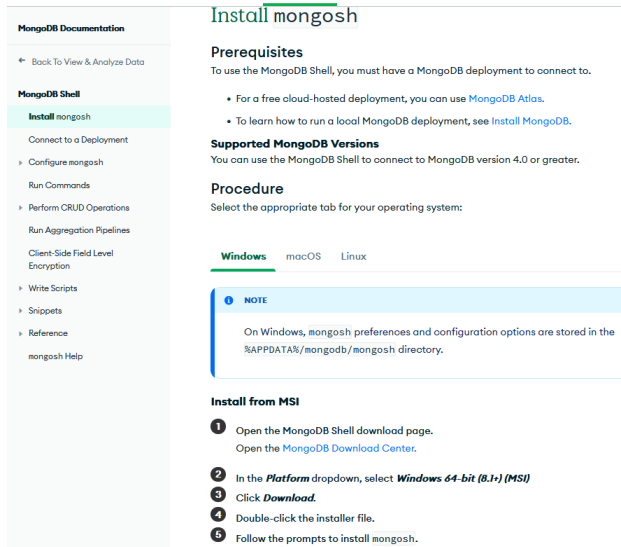
Next >



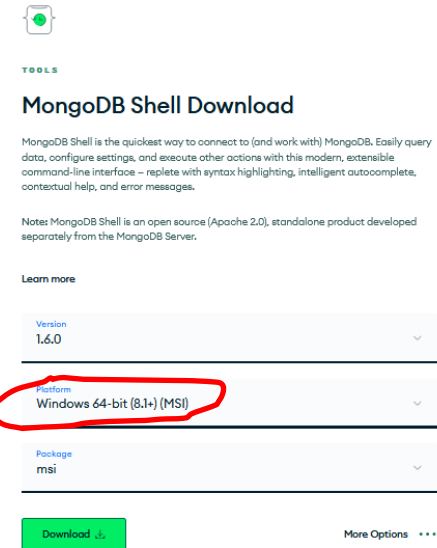
MongoDB shell

(MongoDB shell jest też w MongoDB Compass na samym dole)

instalacja MongoDB shell



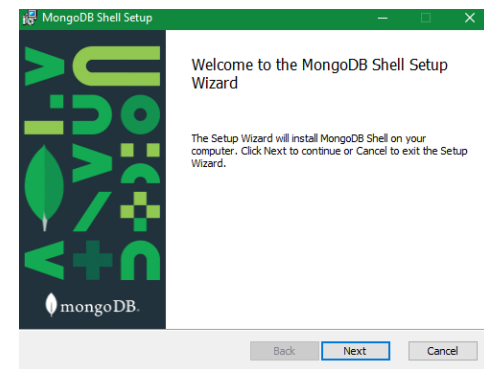
The screenshot shows the MongoDB documentation page for installing the MongoDB Shell (mongosh). The page is titled "Install mongosh" and includes sections for prerequisites, supported versions, and a procedure. The procedure is divided into tabs for Windows, macOS, and Linux. A note indicates that on Windows, preferences and configuration options are stored in the %APPDATA%\mongodb\mongosh directory. The "Install from MSI" section provides a five-step guide: 1. Open the MongoDB Shell download page. 2. In the Platform dropdown, select Windows 64-bit (8.1+) (MSI). 3. Click Download. 4. Double-click the installer file. 5. Follow the prompts to install mongosh.



The screenshot shows the "MongoDB Shell Download" page. It features a "Tools" header and a "MongoDB Shell Download" section. Below this, there is a "Learn more" section and a dropdown menu for "Version" set to "1.6.0". The "Platform" dropdown is highlighted with a red circle and set to "Windows 64-bit (8.1+) (MSI)". The "Package" dropdown is set to "msi". A green "Download" button is visible at the bottom left, and "More Options" is at the bottom right.

<https://www.mongodb.com/docs/mongodb-shell/install/>

defaultowa ścieżka instalacji shella

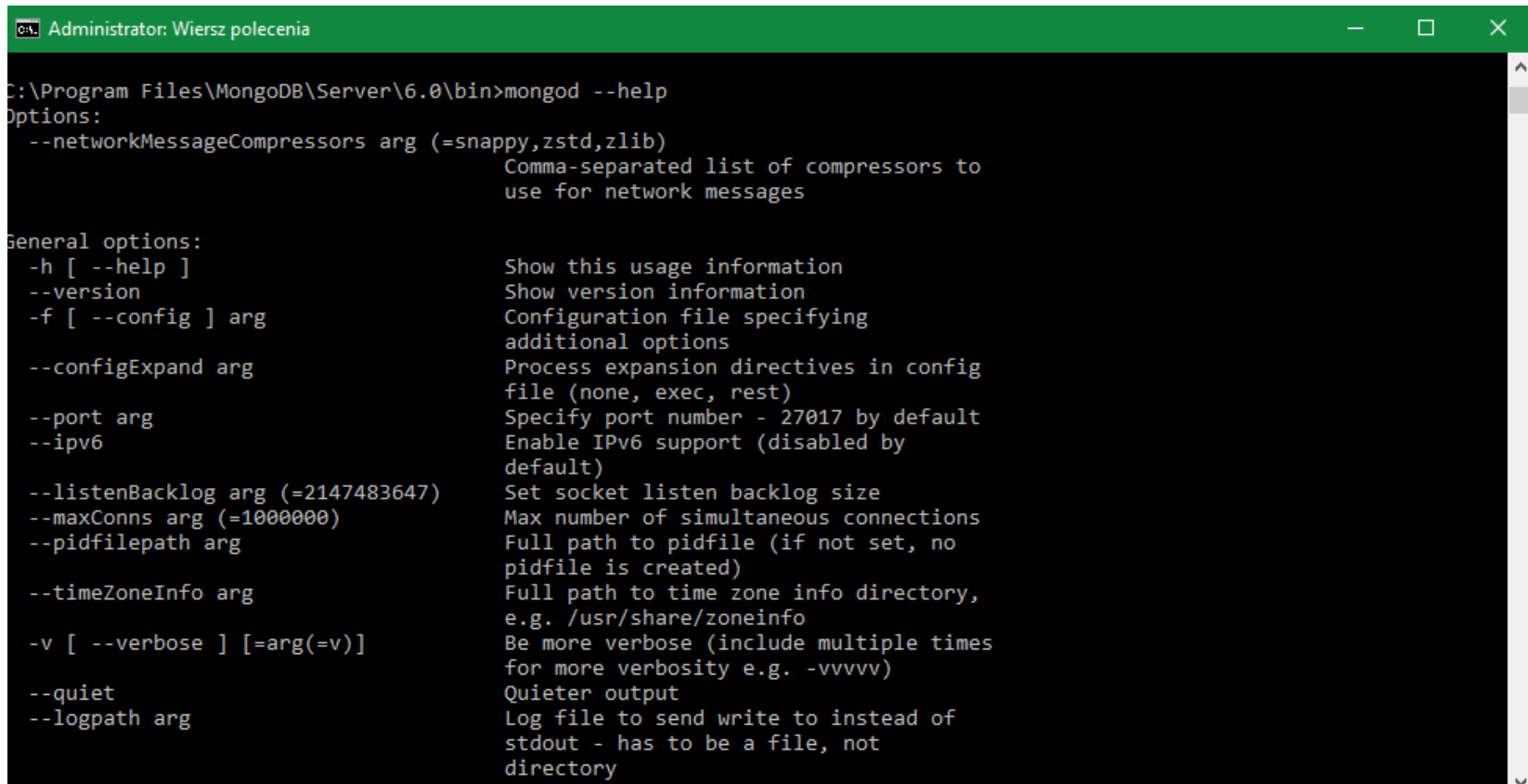


The screenshot shows the "MongoDB Shell Setup" wizard. The window title is "MongoDB Shell Setup". The main content area says "Welcome to the MongoDB Shell Setup Wizard" and "The Setup Wizard will install MongoDB Shell on your computer. Click Next to continue or Cancel to exit the Setup Wizard." At the bottom, there are "Back", "Next", and "Cancel" buttons.

C:\Users\\AppData\Local\Programs\mongosh\

MongoDB help

```
C:\Program Files\MongoDB\Server\6.0\bin>mongod --help
```



```
Administrator: Wiersz polecenia
C:\Program Files\MongoDB\Server\6.0\bin>mongod --help
Options:
  --networkMessageCompressors arg (=snappy,zstd,zlib)
                                Comma-separated list of compressors to
                                use for network messages

General options:
  -h [ --help ]                 Show this usage information
  --version                     Show version information
  -f [ --config ] arg          Configuration file specifying
                                additional options
  --configExpand arg           Process expansion directives in config
                                file (none, exec, rest)
  --port arg                   Specify port number - 27017 by default
  --ipv6                       Enable IPv6 support (disabled by
                                default)
  --listenBacklog arg (=2147483647)
                                Set socket listen backlog size
  --maxConns arg (=1000000)    Max number of simultaneous connections
  --pidfilepath arg           Full path to pidfile (if not set, no
                                pidfile is created)
  --timeZoneInfo arg          Full path to time zone info directory,
                                e.g. /usr/share/zoneinfo
  -v [ --verbose ] [=arg(=v)] Be more verbose (include multiple times
                                for more verbosity e.g. -vvvvv)
  --quiet                      Quieter output
  --logpath arg               Log file to send write to instead of
                                stdout - has to be a file, not
                                directory
```

MongoDB

uruchomienie serwera

```
C:\Program Files\MongoDB\Server\6.0\bin>mongod
```

```
C:\Program Files\MongoDB\Server\6.0\bin>mongod
{"t":{"$date":"2022-11-16T21:49:50.173+01:00"},"s":"I", "c":"CONTROL", "id":23285, "ctx":"thread1","msg":"Aut
{"t":{"$date":"2022-11-16T21:49:51.767+01:00"},"s":"I", "c":"NETWORK", "id":4915701, "ctx":"thread1","msg":"In
InternalClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInter
{"t":{"$date":"2022-11-16T21:49:51.768+01:00"},"s":"I", "c":"NETWORK", "id":4648602, "ctx":"thread1","msg":"Imp
{"t":{"$date":"2022-11-16T21:49:51.771+01:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1","msg":"Suc
Donors"}}}
{"t":{"$date":"2022-11-16T21:49:51.772+01:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1","msg":"Suc
tionRecipients"}}}
{"t":{"$date":"2022-11-16T21:49:51.777+01:00"},"s":"I", "c":"REPL", "id":5123008, "ctx":"thread1","msg":"Suc
{"t":{"$date":"2022-11-16T21:49:51.778+01:00"},"s":"I", "c":"CONTROL", "id":5945603, "ctx":"thread1","msg":"Mul
{"t":{"$date":"2022-11-16T21:49:51.780+01:00"},"s":"I", "c":"CONTROL", "id":4615611, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.780+01:00"},"s":"I", "c":"CONTROL", "id":23398, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.780+01:00"},"s":"I", "c":"CONTROL", "id":23403, "ctx":"initandlisten","msg
allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2022-11-16T21:49:51.781+01:00"},"s":"I", "c":"CONTROL", "id":51765, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.781+01:00"},"s":"I", "c":"CONTROL", "id":21951, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.785+01:00"},"s":"W", "c":"STORAGE", "id":22271, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.793+01:00"},"s":"I", "c":"STORAGE", "id":22270, "ctx":"initandlisten","msg
{"t":{"$date":"2022-11-16T21:49:51.795+01:00"},"s":"W", "c":"STORAGE", "id":22302, "ctx":"initandlisten","msg
```

w przypadku błędu podajemy ścieżkę do lokalizacji baz dbpath
(lub można utworzyć katalog C:\data\db – defaultowo serwer szuka go przy starcie)

```
C:\Program Files\MongoDB\Server\6.0\bin>mongod --dbpath ..\data
```


MongoDB shell

uruchomienie MongoDB shella

C:\Users\

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000

C:\Users\A\AppData\Local\Programs\mongosh>mongosh.exe
Current Mongosh Log ID: 63751f677d532a7091f2433e
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.6.0
Using MongoDB:      6.0.3
Using Mongosh:      1.6.0

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2022-11-16T17:35:23.075+01:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).
test>
The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
-----
test>
```

MongoDB shell

polecenie help

```
test> help

Shell Help:

use          Set current database
show        'show databases'/'show dbs': Print a list of all available databases.
           'show collections'/'show tables': Print a list of all collections for current database.
           'show profile': Prints system.profile information.
           'show users': Print a list of all users for current database.
           'show roles': Print a list of all roles for current database.
           'show log <type>': log for current connection, if type is not set uses 'global'
           'show logs': Print all logs.

exit        Quit the MongoDB shell with exit/exit()/exit
quit       Quit the MongoDB shell with quit/quit()
Mongo      Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optional])
connect    Create a new connection and return the Database object. Usage: connect(URI, username [optional], password [optional])
it        result of the last line evaluated; use to further iterate
version    Shell version
load       Loads and runs a JavaScript file into the current shell environment
enableTelemetry Enables collection of anonymous usage data to improve the mongosh CLI
disableTelemetry Disables collection of anonymous usage data to improve the mongosh CLI
passwordPrompt Prompts the user for a password
sleep     Sleep for the specified number of milliseconds
print     Prints the contents of an object to the output
printjson Alias for print()
cls       Clears the screen like console.clear()
isInteractive Returns whether the shell will enter or has entered interactive mode

For more information on usage: https://docs.mongodb.com/manual/reference/method
```

MongoDB shell

```
mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000
test> db.samochody.insertOne({id: 1, marka: "volvo", cena: "20000", powypadkowy: true});
{
  acknowledged: true,
  insertedId: ObjectId("63752521cec8081a176d5cf8")
}
test> db.samochody.find()
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  }
]
test> db.samochody.insertOne({id: 1, marka: "opel", cena: "10000", powypadkowy: false});
{
  acknowledged: true,
  insertedId: ObjectId("63752593cec8081a176d5cf9")
}
test> db.samochody.find()
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  },
  {
    _id: ObjectId("63752593cec8081a176d5cf9"),
    id: 1,
    marka: 'opel',
    cena: '10000',
    powypadkowy: false
  }
]
```

insertOne()

find()

sprawdzenie

utworzenie kolekcji
samochody z jednym
dokumentem

dodanie kolejnego
dokumentu do
kolekcji samochodu

zapisane dokumenty
w kolekcji
samochody

MongoDB shell

insertMany()

```
test> db.samochody.insertMany(
... [
...   {
...     id: 3,
...     marka: 'mercedes',
...     cena: '40000',
...     powypadkowy: true
...   },
...   {
...     id: 4,
...     marka: 'bmw',
...     cena: '12000',
...     powypadkowy: false
...   }
... ]
... );
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("637528adcec8081a176d5cfa"),
    '1': ObjectId("637528adcec8081a176d5cfb")
  }
}
```

wszystkie
dodane
dokumenty
w kolekcji
samochody

```
test> db.samochody.find()
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  },
  {
    _id: ObjectId("63752593cec8081a176d5cf9"),
    id: 1,
    marka: 'opel',
    cena: '10000',
    powypadkowy: false
  },
  {
    _id: ObjectId("637528adcec8081a176d5cfa"),
    id: 3,
    marka: 'mercedes',
    cena: '40000',
    powypadkowy: true
  },
  {
    _id: ObjectId("637528adcec8081a176d5cfb"),
    id: 4,
    marka: 'bmw',
    cena: '12000',
    powypadkowy: false
  }
]
```

MongoDB shell

```
test> db.samochody.find({id:1})
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  },
  {
    _id: ObjectId("63752593cec8081a176d5cf9"),
    id: 1,
    marka: 'opel',
    cena: '10000',
    powypadkowy: false
  }
]
```

wyszukanie dokumentów z id=1

```
test> db.samochody.find({id:1, powypadkowy:false})
[
  {
    _id: ObjectId("63752593cec8081a176d5cf9"),
    id: 1,
    marka: 'opel',
    cena: '10000',
    powypadkowy: false
  }
]
```

wyszukanie dokumentów z id=1
oraz powypadkowy=false

MongoDB shell

```
test> db.samochody.find({marka: {$in: ["bmw", "volvo"]}})
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  },
  {
    _id: ObjectId("637528adcec8081a176d5cfb"),
    id: 4,
    marka: 'bmw',
    cena: '12000',
    powypadkowy: false
  }
]
```

użycie kryterium \$in
wybiera dokumenty, których
marka pokrywa się z którąkolwiek
marką podaną w tablicy

kryteria:

\$in – pole równe którejś wartości w tablicy

\$nin - pole inne niż wartości w tablicy

\$eq – równe

\$ne – nierówne

\$gt - większe niż

\$lt – mniejsze niż

\$gte – większe lub równe

\$lte – mniejsze lub równe

<https://www.mongodb.com/docs/manual/reference/operator/query-comparison/>

MongoDB Compass

The screenshot shows the MongoDB Compass interface. The left sidebar displays the database structure: localhost:27017, 4 DBS, 4 COLLECTIONS, and a list of databases including admin, config, local, and test. The 'test' database is selected, and the 'samochoody' collection is highlighted. The main area shows the 'test.samochoody' collection with 4 documents and 1 index. The 'Documents' tab is active, displaying a list of documents with their JSON structure. The filter bar shows a filter: { field: 'value' }. The document list shows four entries with fields: _id, id, marka, cena, and powypadkowy. The status bar indicates 'Displaying documents 1 - 4 of 4'.

```
{ "_id": ObjectId('63752521ceec8081a176d5cf8'), "id": 1, "marka": "volvo", "cena": "20000", "powypadkowy": true }
{ "_id": ObjectId('63752593ceec8081a176d5cf9'), "id": 1, "marka": "opel", "cena": "10000", "powypadkowy": false }
{ "_id": ObjectId('637528adceec8081a176d5cfa'), "id": 3, "marka": "mercedes", "cena": "40000", "powypadkowy": true }
{ "_id": ObjectId('637528adceec8081a176d5cfb'), "id": 4, "marka": "bmw", "cena": "12000", "powypadkowy": false }
```

MongoDB shell

wszystkie dodane dokumenty w kolekcji samochody widoczne w aplikacji Compass

Node.js MongoDB

MongoDb tutorial w3schools

Node.js MongoDB

[← Previous](#)

Node.js can be used in database applications.

One of the most popular NoSQL database is MongoDB.

MongoDB

To be able to experiment with the code examples, you will need access to a MongoDB database.

You can download a free MongoDB database at <https://www.mongodb.com>.

Or get started right away with a MongoDB cloud service at <https://www.mongodb.com/cloud/atlas>.

Install MongoDB Driver

Let us try to access a MongoDB database with Node.js.

To download and install the official MongoDB driver, open the Command Terminal and execute the following:

https://www.w3schools.com/nodejs/nodejs_mongodb.asp

Node.js MongoDB

```
npm install mongodb --save
```

instalacja sterownika mongodb

```
EXPLORER  ...  JS app.js  X  w przypadku błędów wymienić localhost na 127.0.0.1
v MONGODB
  > node_modules
  JS app.js
  {} package-lock.json
  {} package.json

połączenie z bazą test i wybranie pierwszego dokumentu w kolekcji samochody

JS app.js > ...
1 // konfiguracja klienta
2 var url = "mongodb://localhost:27017/";
3 var MongoClient = require('mongodb').MongoClient;
4
5 // połączenie z bazą test
6 MongoClient.connect(url, function(err, db) {
7   if (err) throw err;
8   console.log("Połączenie z MongoDB ustanowione");
9   var dbo = db.db("test");
10  // wybranie pierwszego dokumentu w kolekcji samochody
11  dbo.collection("samochody").findOne({}, function(err, result) {
12    if (err) throw err;
13    console.log(`
14      Pierwszy dokument w kolekcji samochody:
15      id: ${result.id},
16      marka: ${result.marka},
17      powypadkowy: ${result.powypadkowy},
18    `);
19    db.close();
20  });
21 });
```

Node.js MongoDB

```
PS D:\arch\Technik Programista\_aplikacjeWebowe\node.js\programy\mongoDB> node app.js
Połączenie z MongoDB ustanowione

Pierwszy dokument w kolekcji samochody:
  id: 1,
  marka: volvo,
  powypadkowy: true,
```



logi połączenia z bazą

help

npm -y init // tworzenie projektu w node.js

npm install express //instalacja modułu express

stworzyć plik app.js

node app.js // uruchamianie aplikacji

<http://localhost:8080> // serwer nasłuchuje na danym porcie (w przypadku aplikacji serwera)