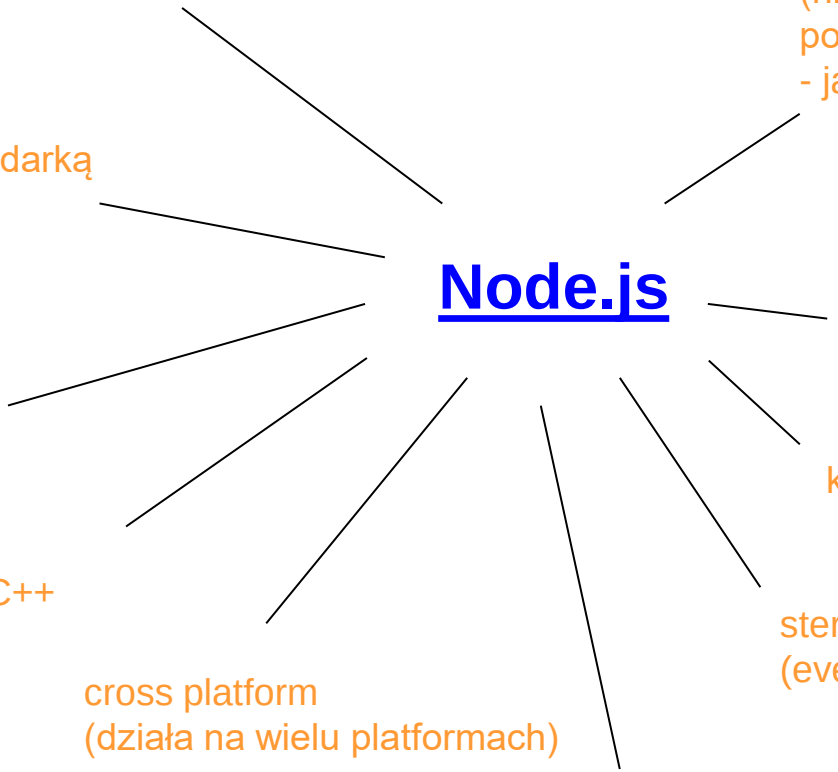
Node.js

Node.js

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

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

działa poza przeglądarką
(back-end)



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

open-source server

korzysta z wielowątkowości V8

napisany w C++

sterowany zdarzeniami
(event-driven architecture)

cross platform
(działa na wielu platformach)

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

JS

1995

silnik JS open-source napisany przez Google w C++, używany w przeglądarce Chrome

V8

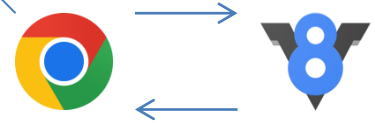
2008

środowisko uruchomieniowe poza przeglądarką, napisane przez Ryan Dahl, używa V8

Node.js

2009

przeglądarka Chrome



kod JS

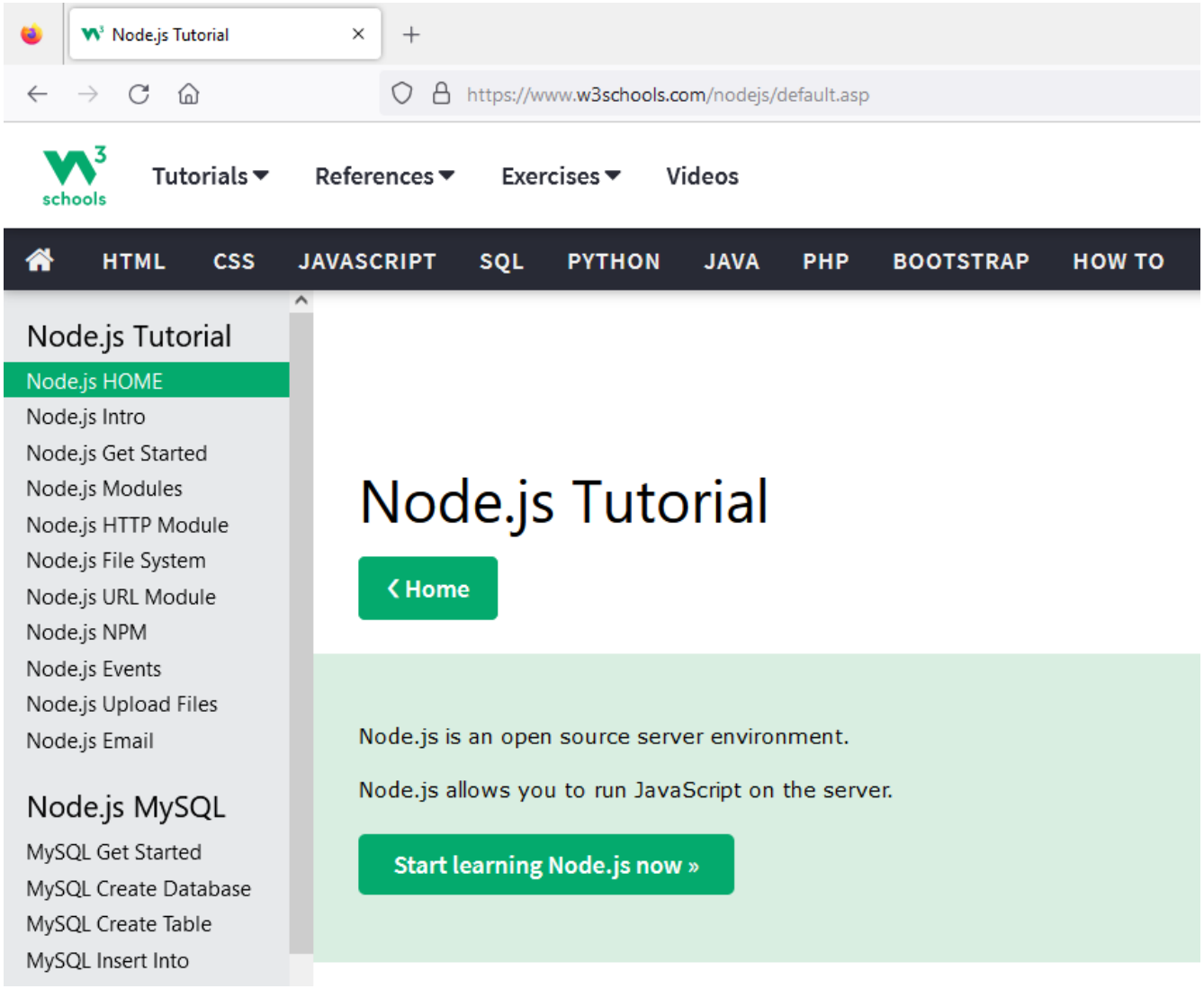
interpretacja kodu JS

środowisko uruchomieniowe kodu JS

kod JS

interpretacja kodu JS

Node.js

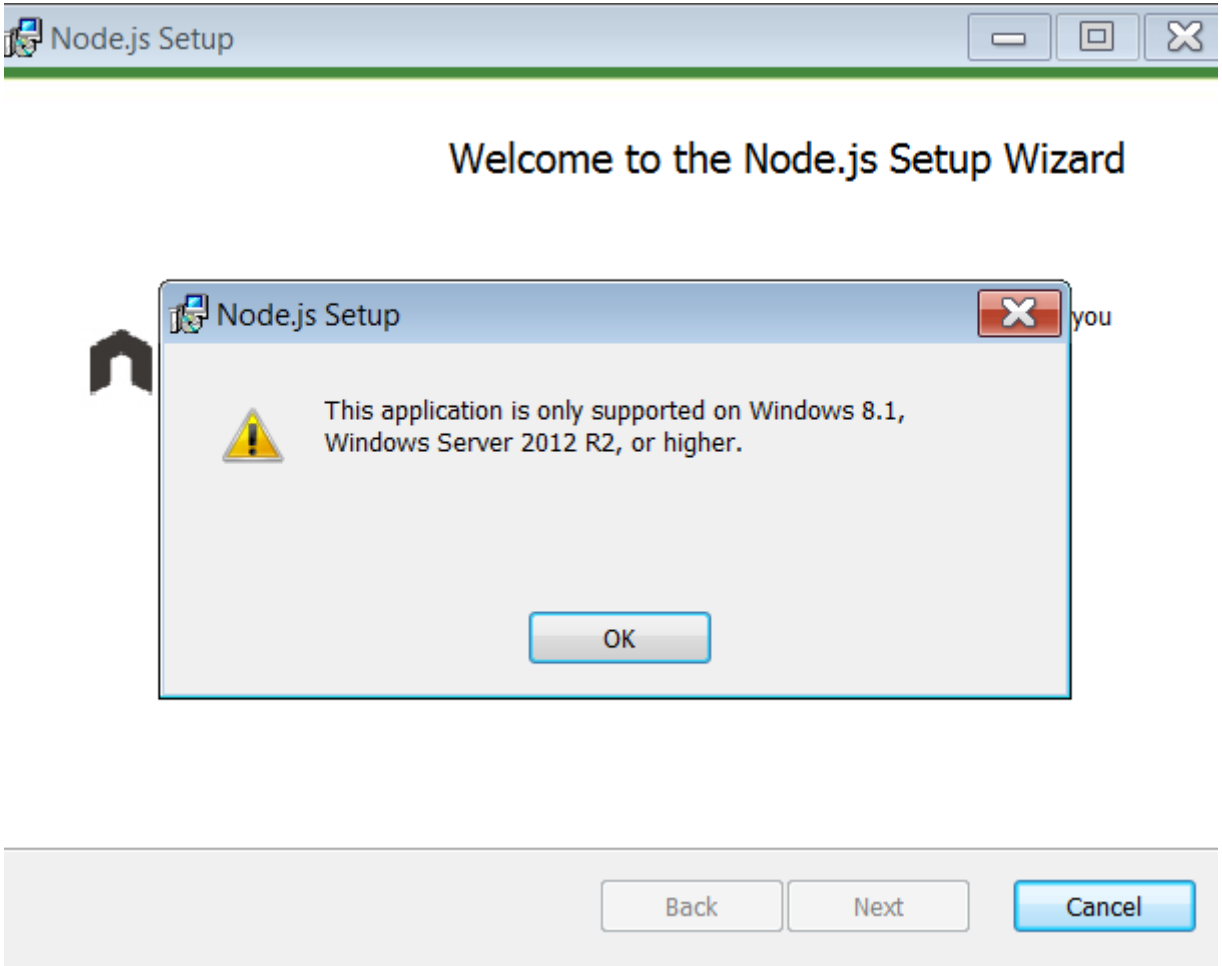


https://www.w3schools.com/nodejs/default.asp

Node.js

https://nodejs.org/en/

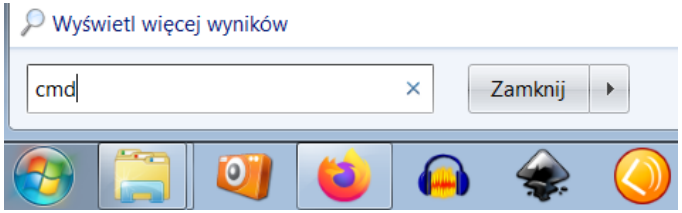
Node.js



Zainstaluj Node.js na Win7 (wersja 13.14.0)

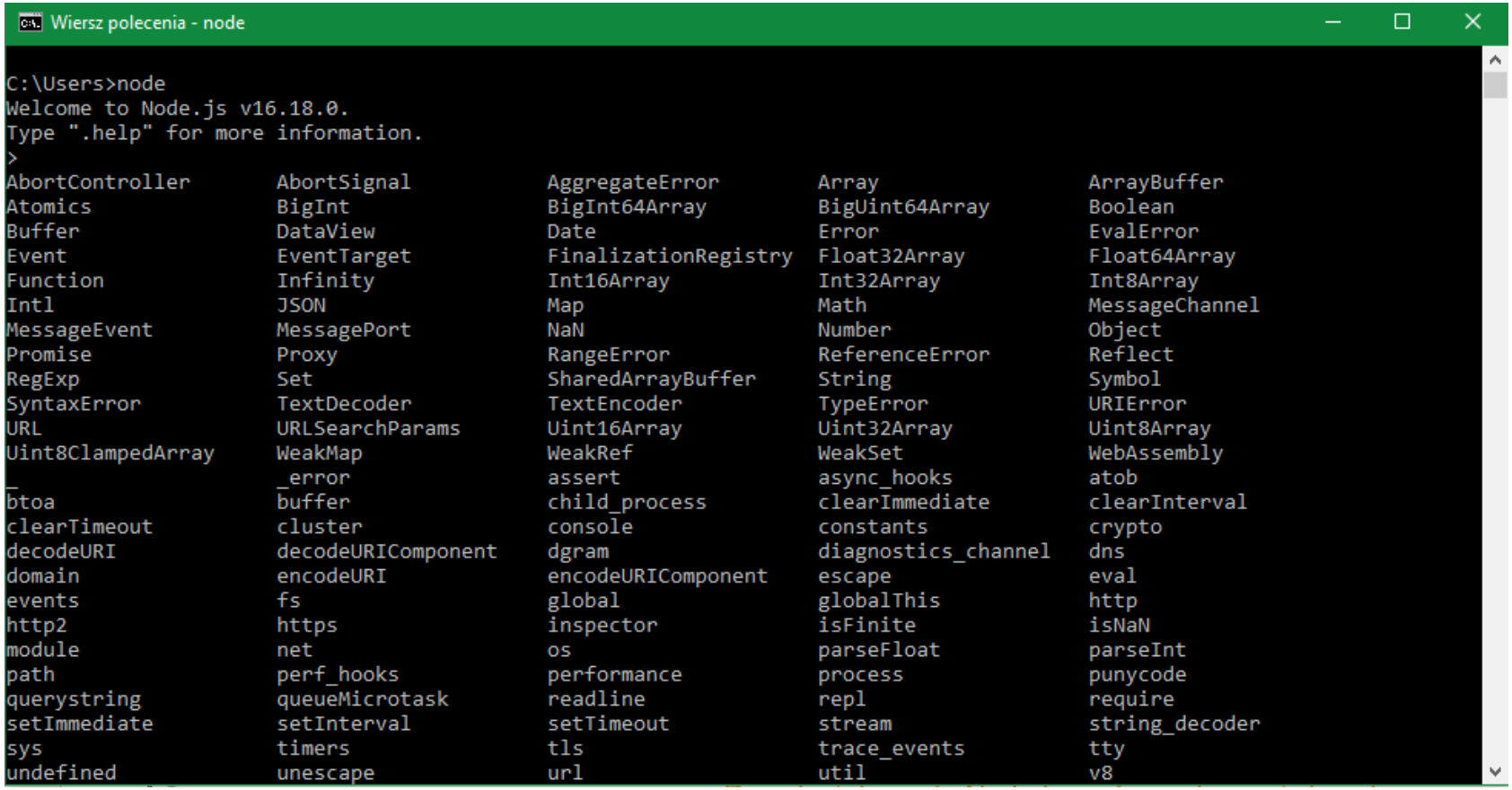
Node.js

```
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
>
```



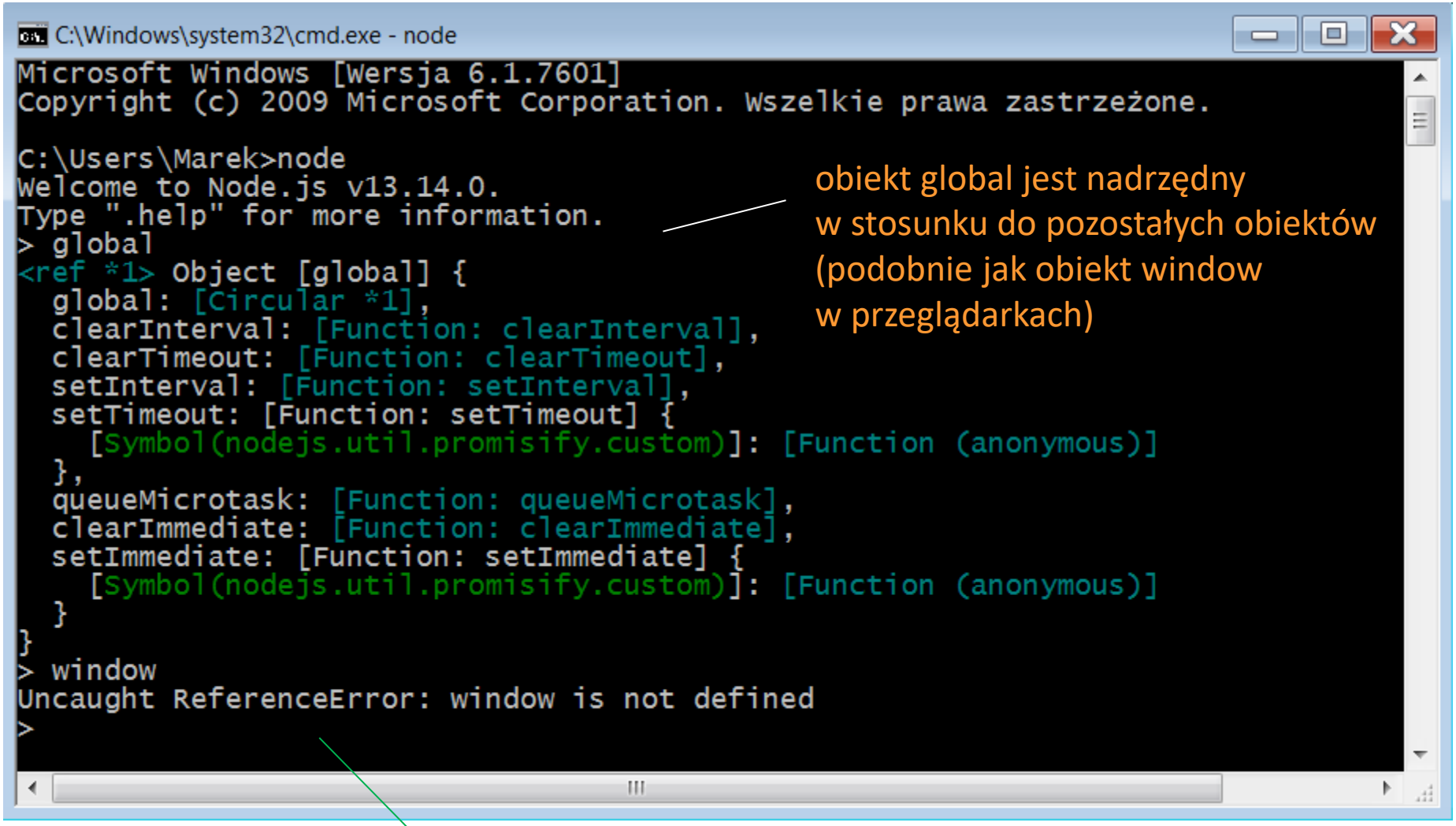
Po zainstalowaniu Node.js możemy korzystać z wiersza poleceń interaktywnego terminala REPL (read, eval, print, loop)
>node // włączenie terminala
>2*2
Polecenia nie są związane z przeglądarką
(np. brak obiektu window)

Node.js



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

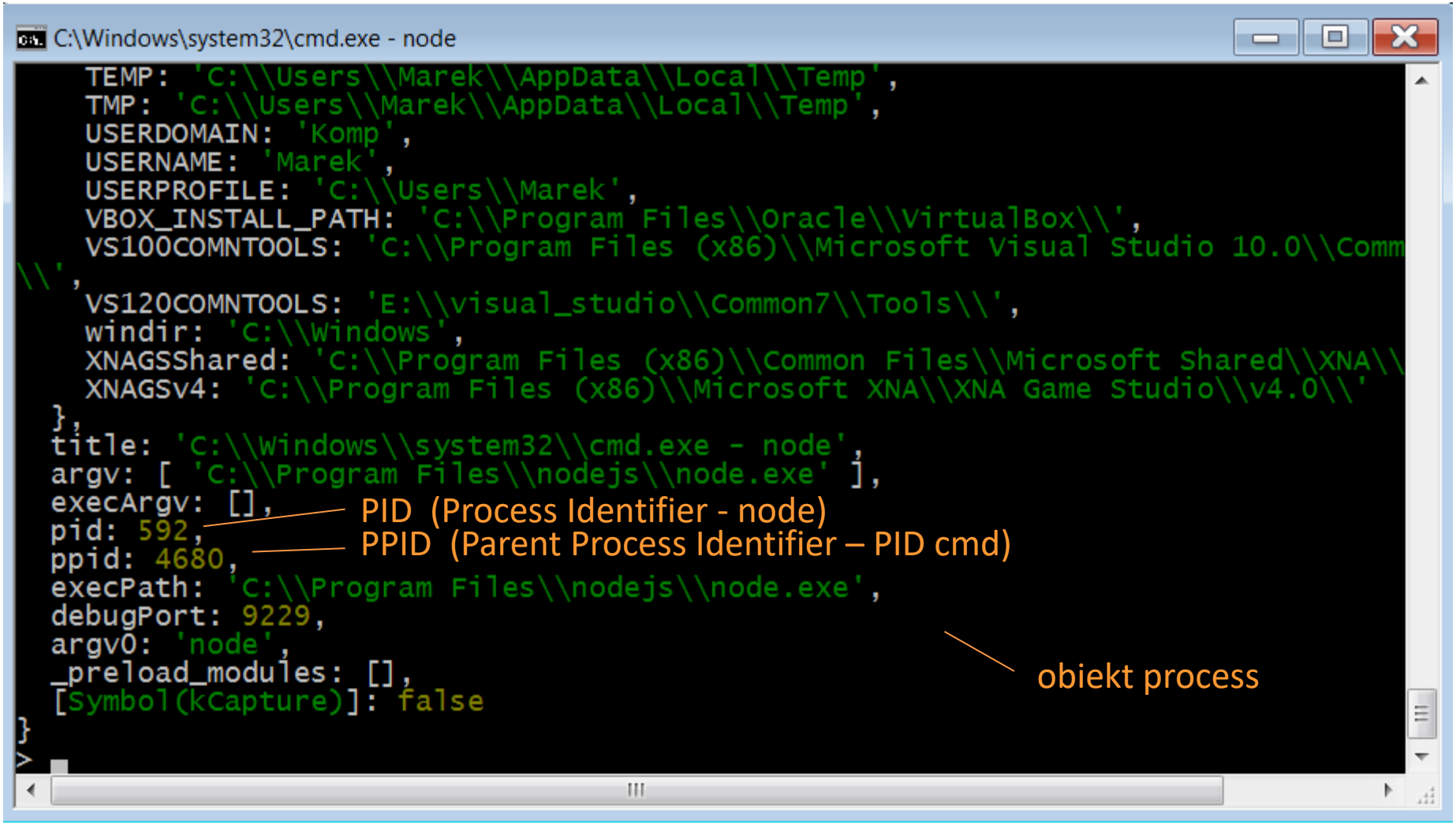
Node.js



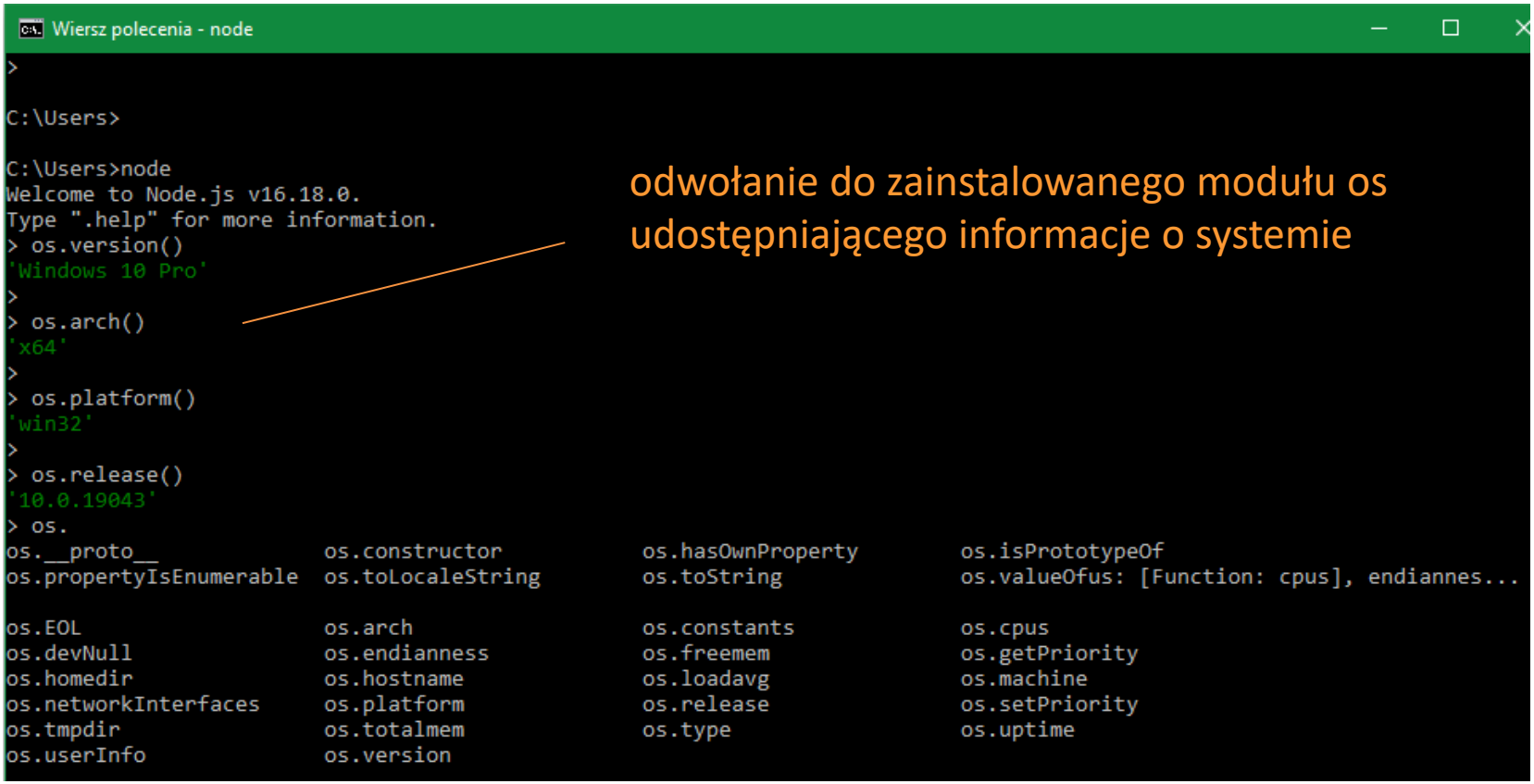
obiekt global jest nadrzędny
w stosunku do pozostałych obiektów
(podobnie jak obiekt window
w przeglądarkach)

brak obiektu window w Node.js

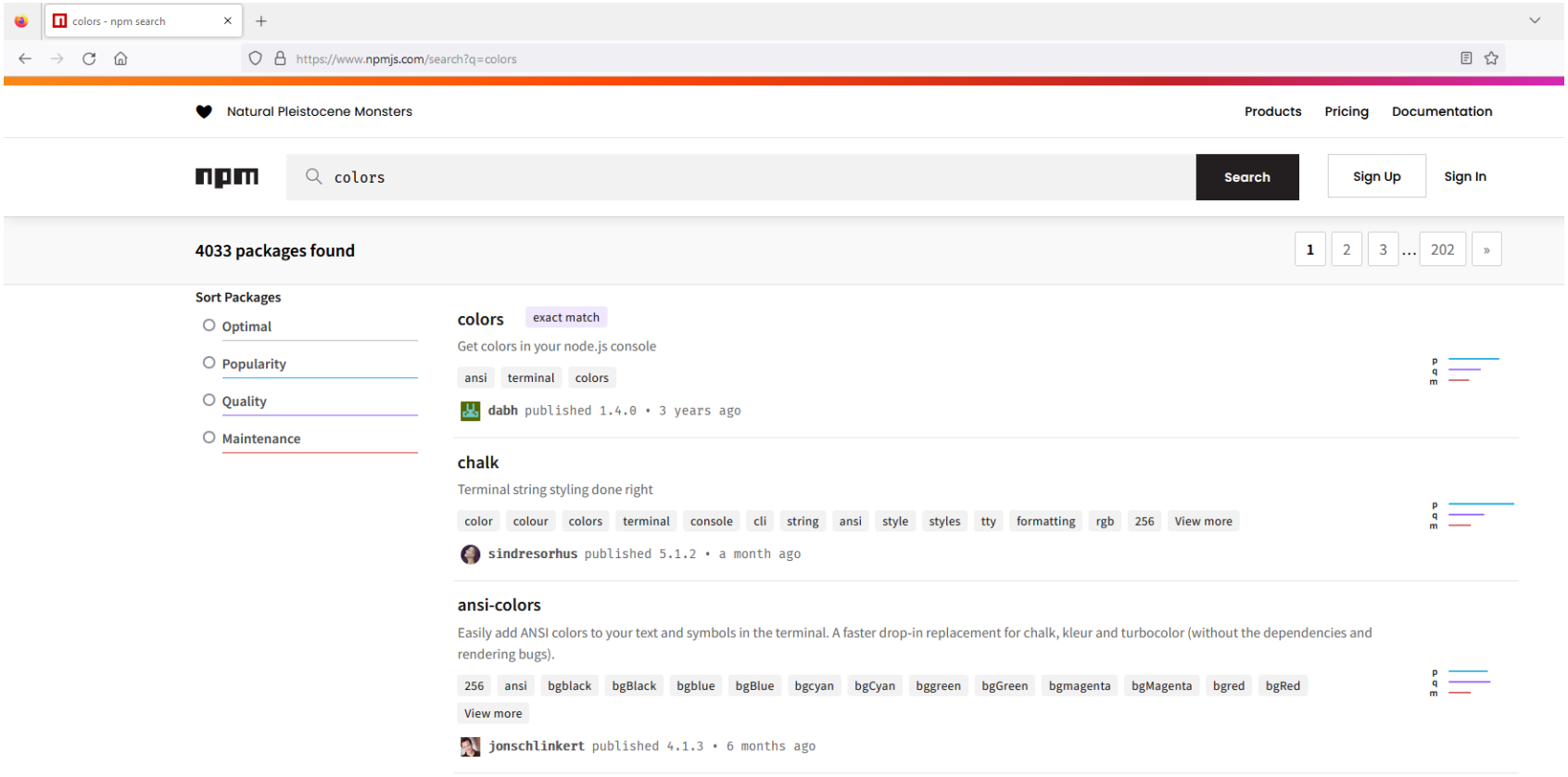
Node.js


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
udostępniającego informacje o systemie

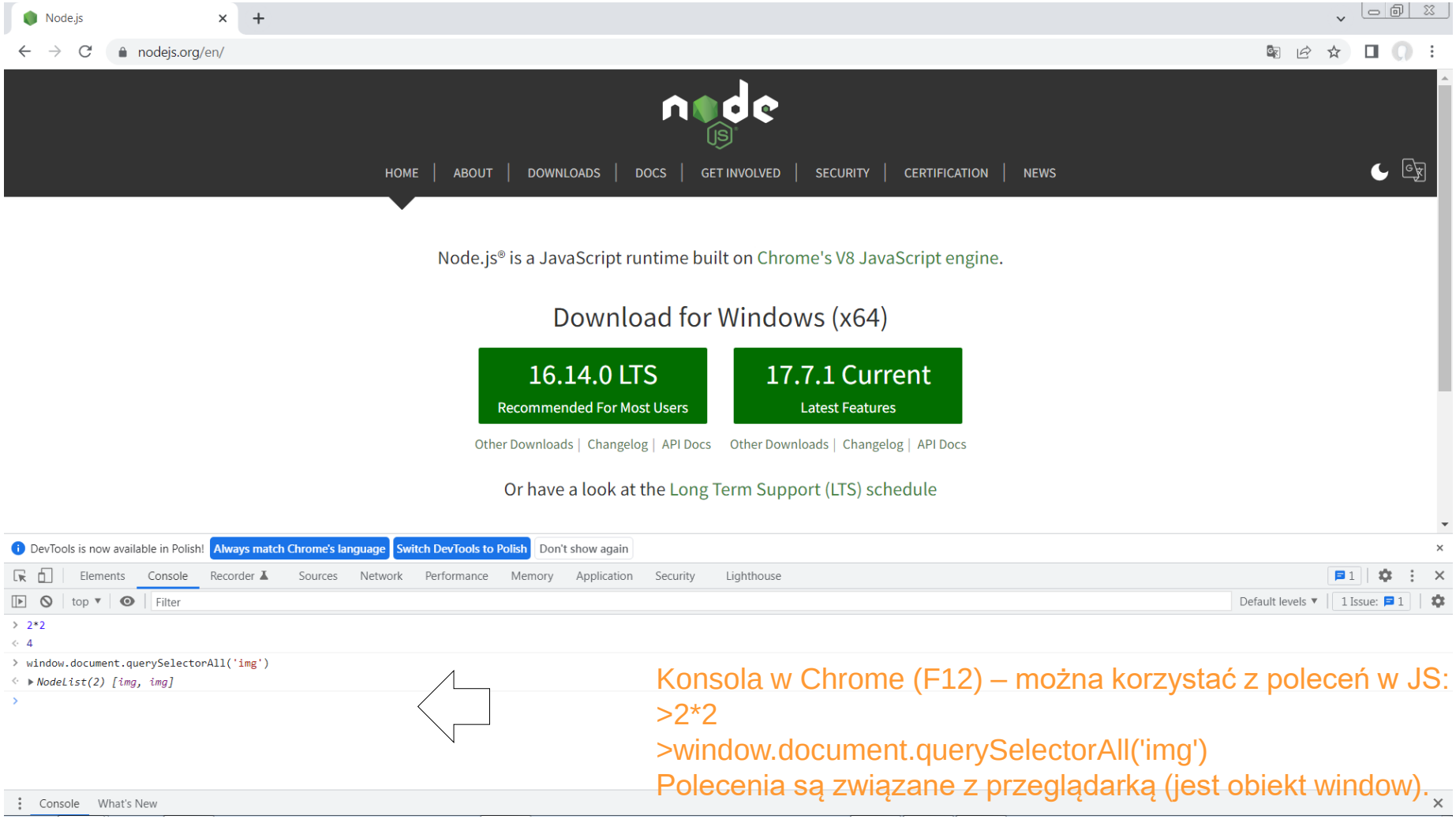
Node.js



https://www.npmjs.com/

wyszukiwarka pakietów

Node.js

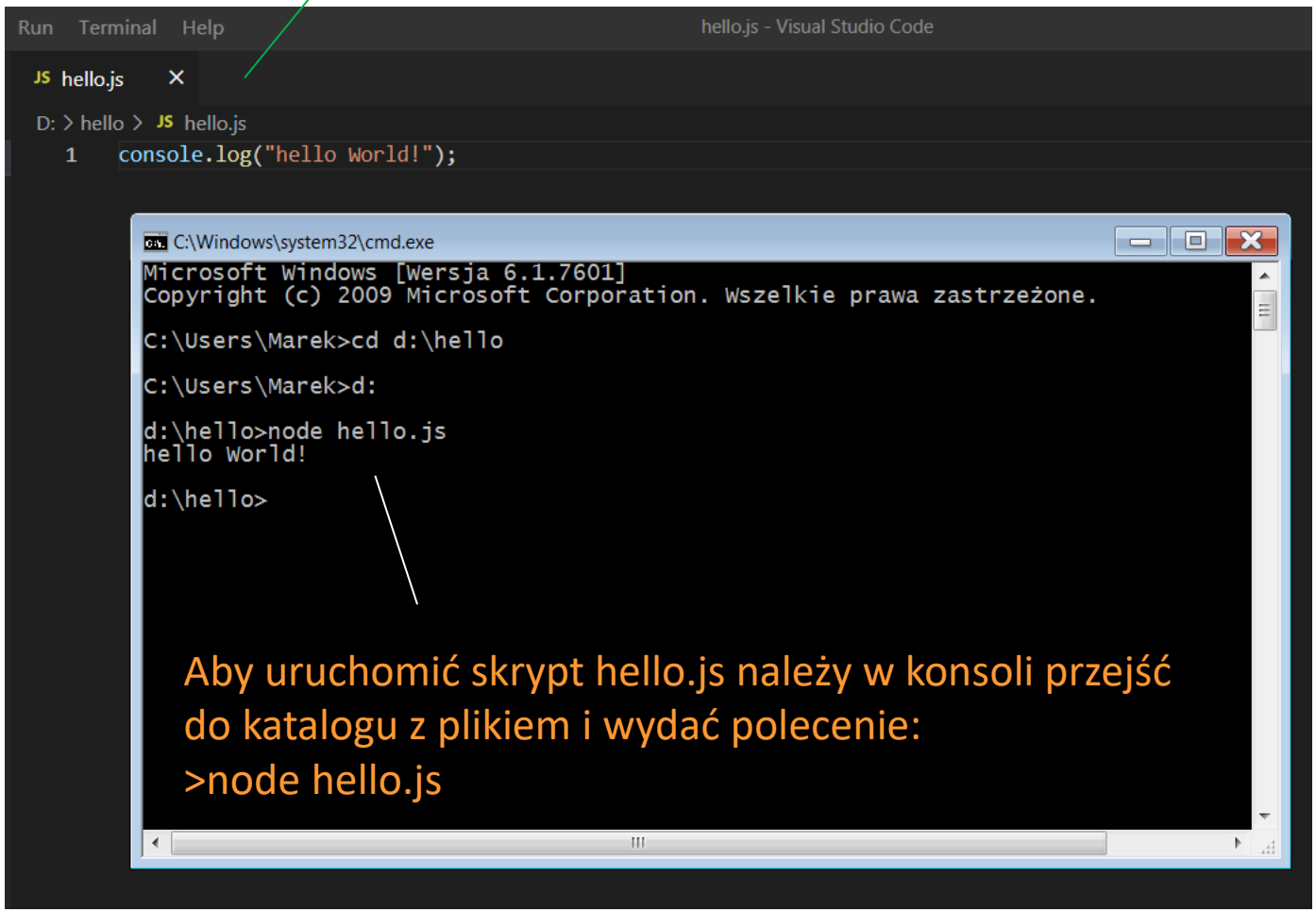


Konsola w Chrome (F12) – można korzystać z poleceń w JS:
>2*2
>window.document.querySelectorAll('img')
Polecenia są związane z przeglądarką (jest obiekt window).

Node.js

Uruchamianie aplikacji Noda – pliku hello.js bez zakładania projektu

Plik hello.js otwarty w edytorze Visual Studio Code

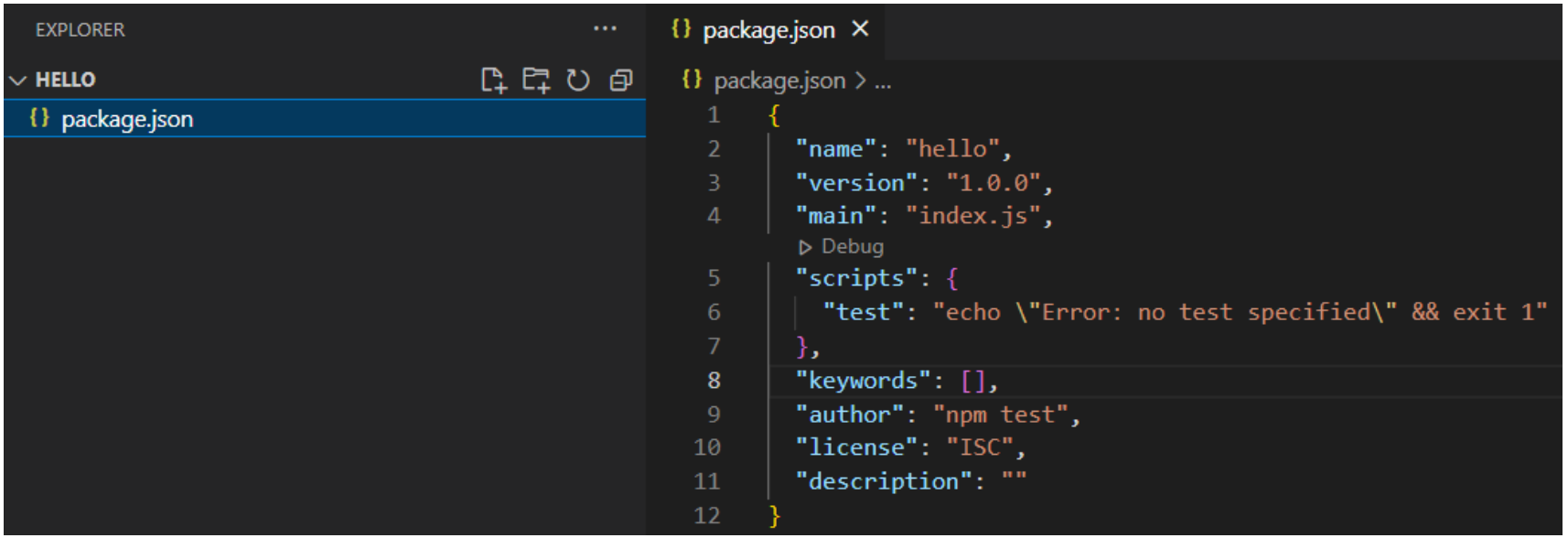


Node.js
nowy projekt

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

```
npm -y init
```

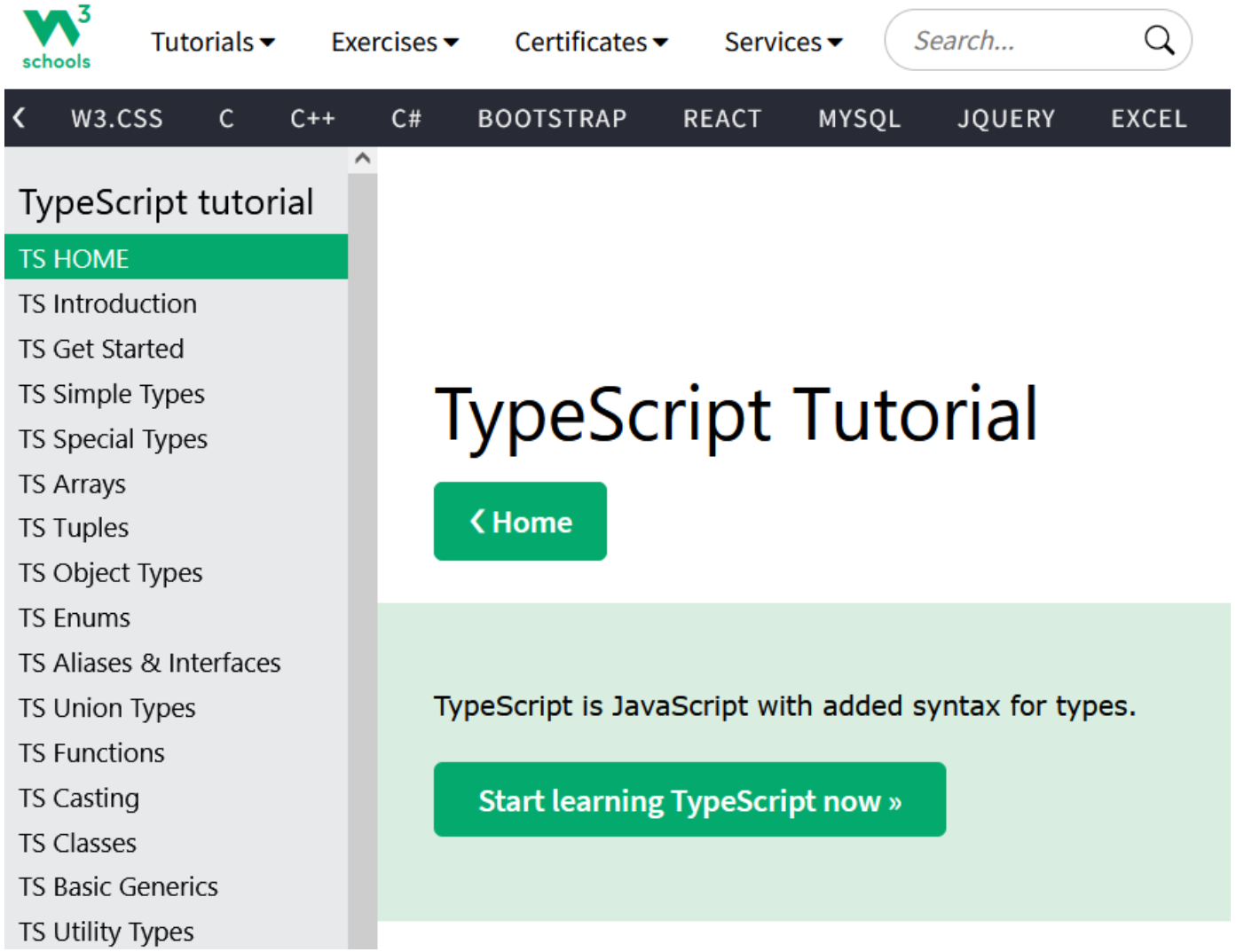
wygenerowany zostaje plik package.json z konfiguracją projektu



TypeScript

https://www.typescriptlang.org/

TypeScript



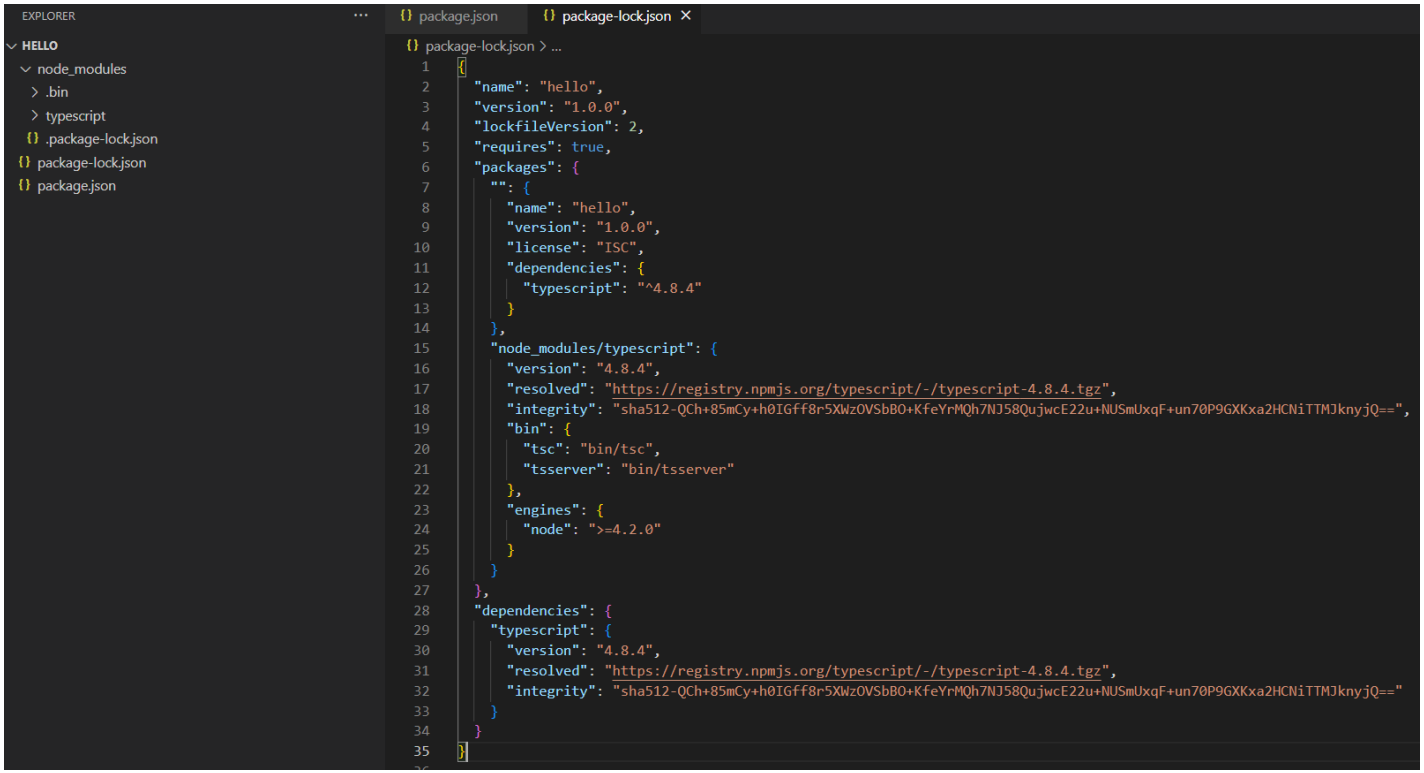
https://www.w3schools.com/typescript/

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



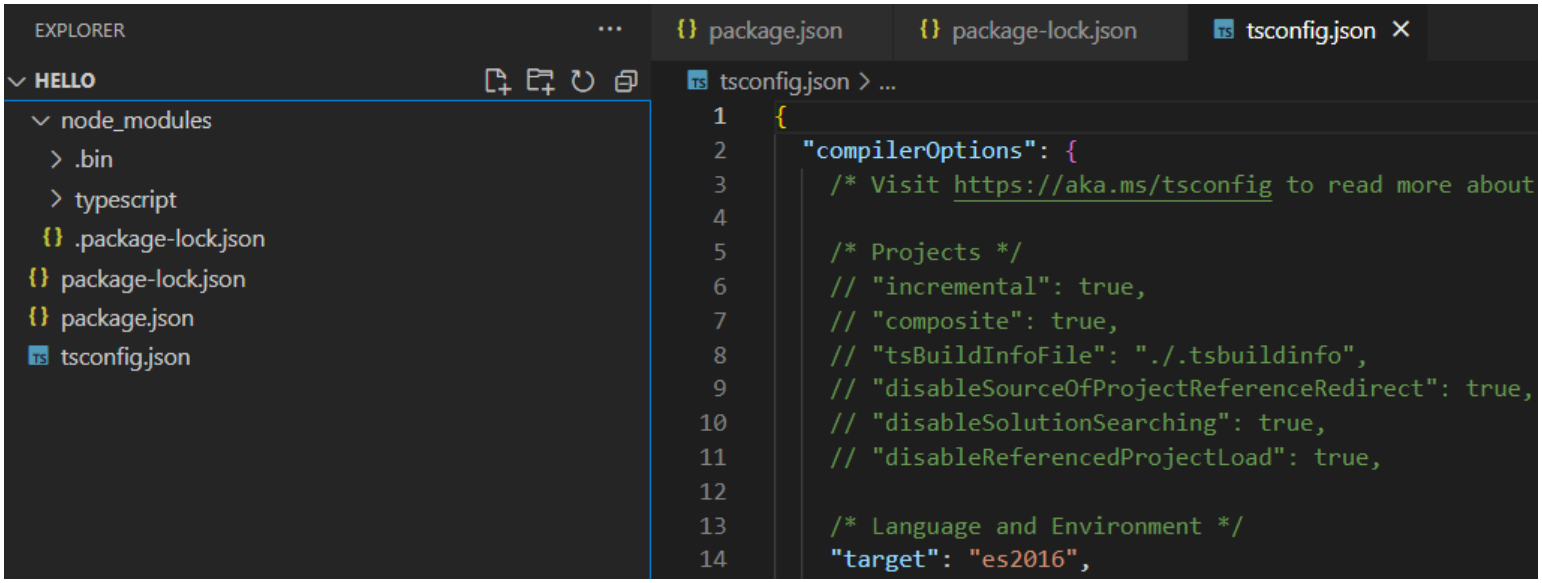
Node.js
inicjalizacja TypeScript

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

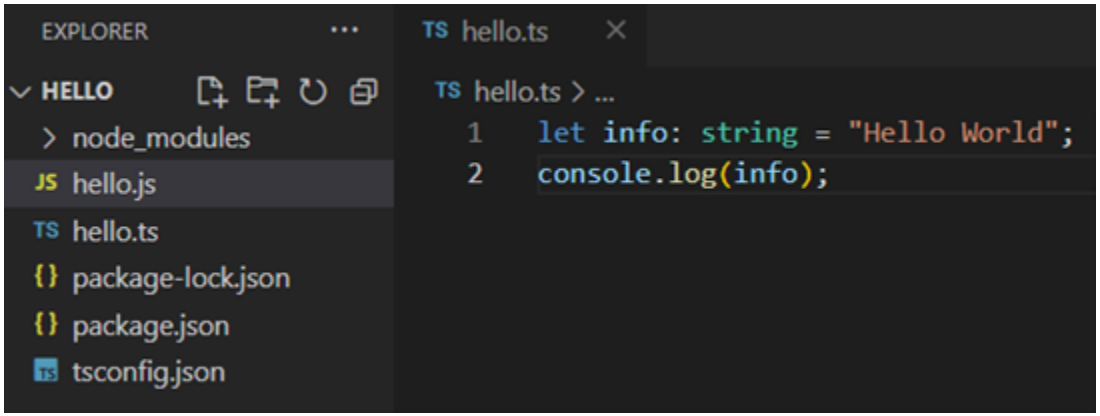
```
hello> npx tsc -init
```

npx: Node Package Execute - uruchomienie modułu z katalogu node_modules

wygenerowany zostaje plik tsconfig.json z konfiguracją kompilatora



Node.js
plik hello.ts



tworzymy plik hello.ts w języku TypeScript

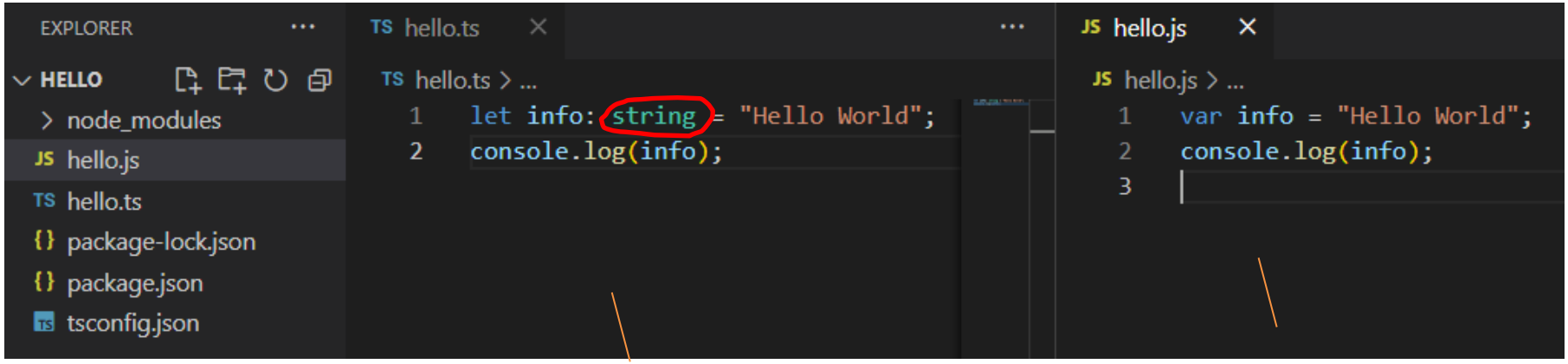
Node.js
TypeScript

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 konwerujące plik hello.ts do języka JavaScript:

```
npx tsc hello.ts
```

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



plik źródłowy (jest typ string - statyczny mechanizm typowania)

wygenerowny 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 uruchmiające tryb nasłuchu – automatyczną kompilacje pliku ts po wykryciu w nim zmiany

Node.js
TypeScript

hello.ts

hello.js

TS hello.ts > ...

```ts
let tekst: string = "napis"; // typ tekstowy
let decimal: number = 3.14;     // typ liczbowy dziesiętny
let hex: number = 0xfff;    // typ liczbow szesnastkowy
let octal: number = 0o765;  // typ liczbowy ósemkowy
let binary: number = 0b1110; // typ liczbowy binarny
let logic: boolean = true;   // typ logiczny

let wszystkoJednoCo: any = 23; // typ dynamiczny
wszystkoJednoCo = "napis"; // typ string
wszystkoJednoCo = false; // typ boolean

console.log(`wszystkoJednoCo:
            wartość: ${wszystkoJednoCo}
            nazwa typu: ${typeof wszystkoJednoCo}`);

let common: number | string // zmienna może przyjmować
                            // typ tekstowy lub liczbowy
let marka: "bmw" | "mercedes" | "ford"; // typ jako zbór wartości
//marka = "opel";    // błąd - brak opla w zdefiniowanych wartościach'

let auta: string[] = ["bmw", "mercedes", "ford"]; // tablica
let krotka: [number, boolean] = [25, true];  // krotka - tablica róznych typów
enum colors {red, green, blue}; // enum - wyliczenie
let czerwony = colors.red;      // korzystanie z enum colors
console.log(czerwony);
```

statyczne typowanie w TypeScript

JS hello.js > ...

```js
var tekst = "napis"; // typ tekstowy
var decimal = 3.14; // typ liczbowy dziesiętny
var hex = 0xfff; // typ liczbow szesnastkowy
var octal = 501; // typ liczbowy ósemkowy
var binary = 14; // typ liczbowy binarny
var logic = true; // typ logiczny

var wszystkoJednoCo = 23; // typ dynamiczny
wszystkoJednoCo = "napis"; // typ string
wszystkoJednoCo = false; // typ boolean

console.log(`wszystkoJednoCo:
            wartość: ${wszystkoJednoCo}
            nazwa typu: ${typeof wszystkoJednoCo}`);

var common; // zmienna może przyjmować
// typ tekstowy lub liczbowy
var marka; // typ jako zbór wartości
//marka = "opel";    // błąd - brak opla w zdefiniowanych wartościach'

var auta = ["bmw", "mercedes", "ford"]; // tablica
var krotka = [25, true]; // krotka - tablica róznych typów
var colors;
(function (colors) {
    colors[colors["red"] = 0] = "red";
    colors[colors["green"] = 1] = "green";
    colors[colors["blue"] = 2] = "blue";
})(colors || (colors = {}));
; // enum - wyliczenie
var czerwony = colors.red; // korzystanie z enum colors
console.log(czerwony);
```

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

dynamiczne typowanie w JavaScript

uruchomienie hello.js

Node.js
asynchroniczność

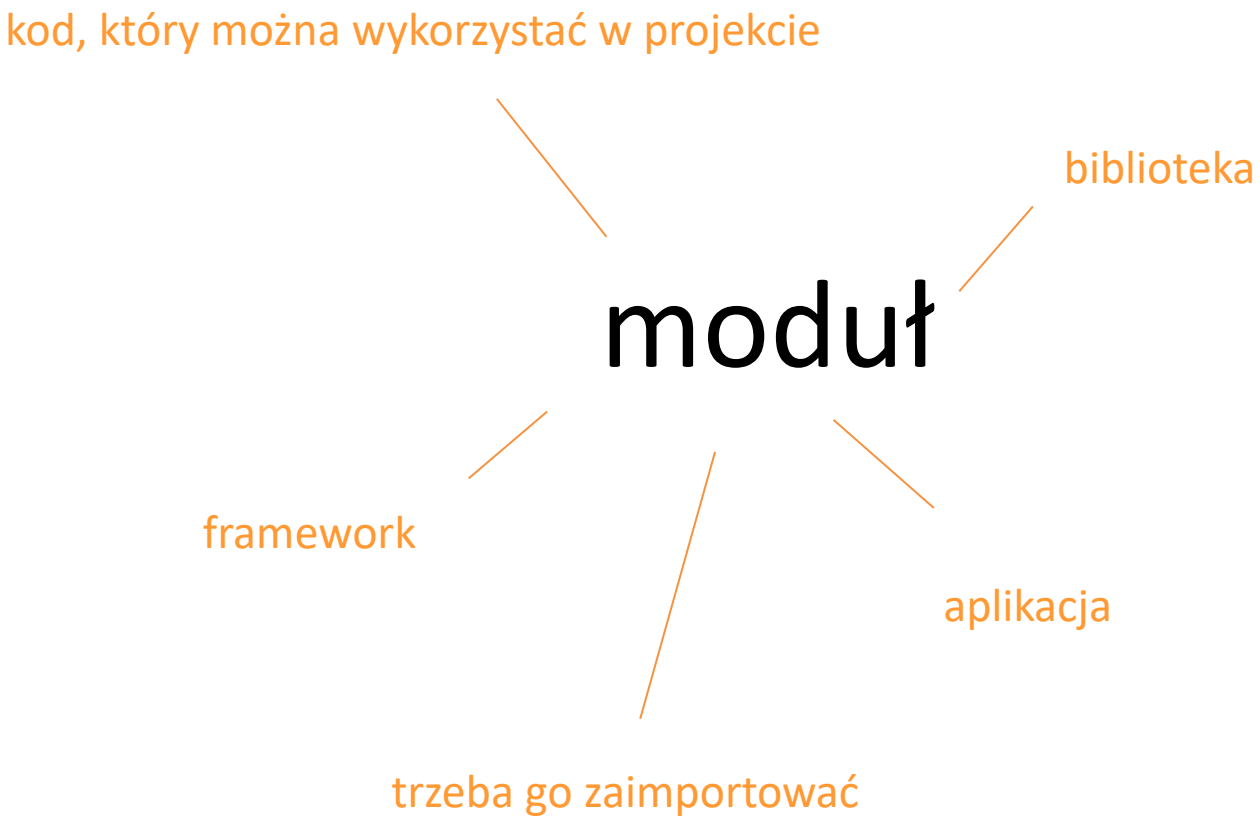
funkcja one ma ustawiony 2s timeout

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

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

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:

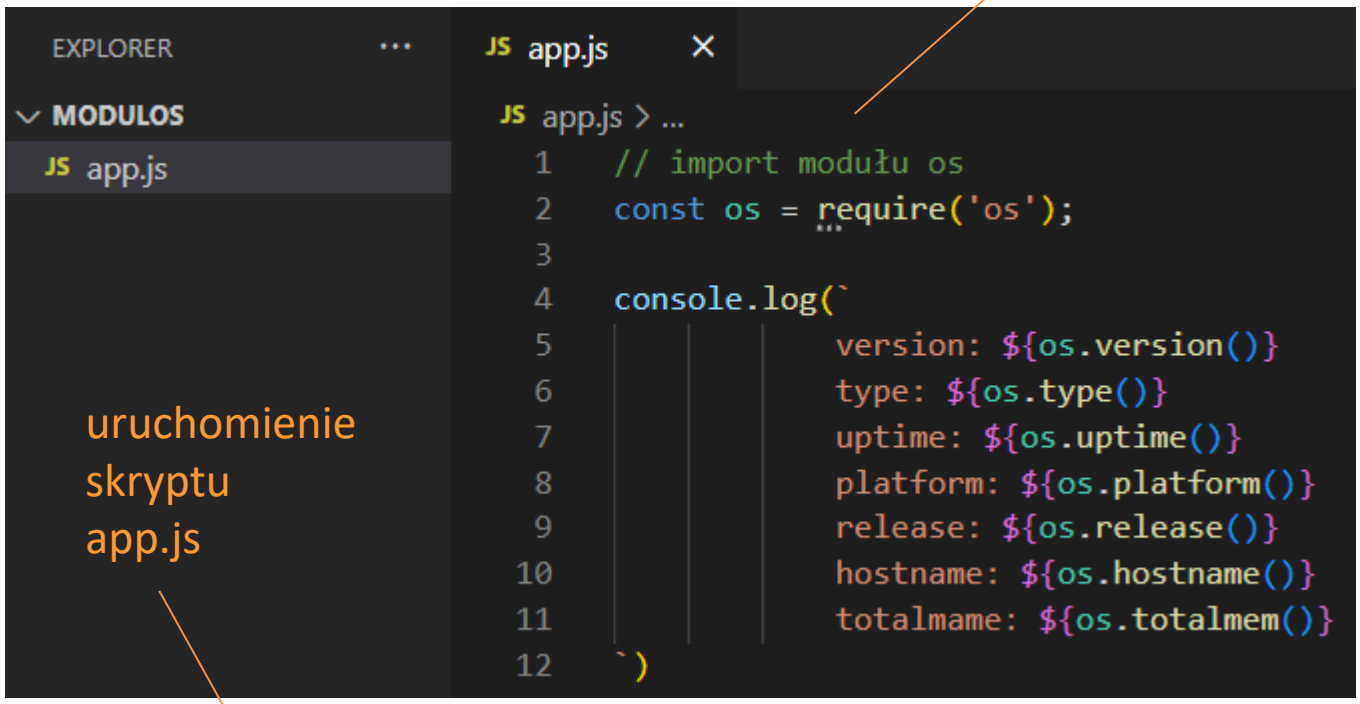
Module	Description
assert	Provides a set of assertion tests
buffer	To handle binary data
child_process	To run a child process
cluster	To split a single Node process into multiple processes
crypto	To handle OpenSSL cryptographic functions
dgram	Provides implementation of UDP datagram sockets
dns	To do 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

moduł os
plik app.js

informacje o systemie operacyjnym

import modułu



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
// import modułu os
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

MODULURL

app.js

app.js

app.js > ...

```
1   // import modułu url
2   const url = require('url');
3
4   // adres dp 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                  host: ${addr.host}
11                  href: ${addr.href}
12                  hostname: ${addr.hostname}
13                  path: ${addr.path}
14                  protocol: ${addr.protocol}
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:
```

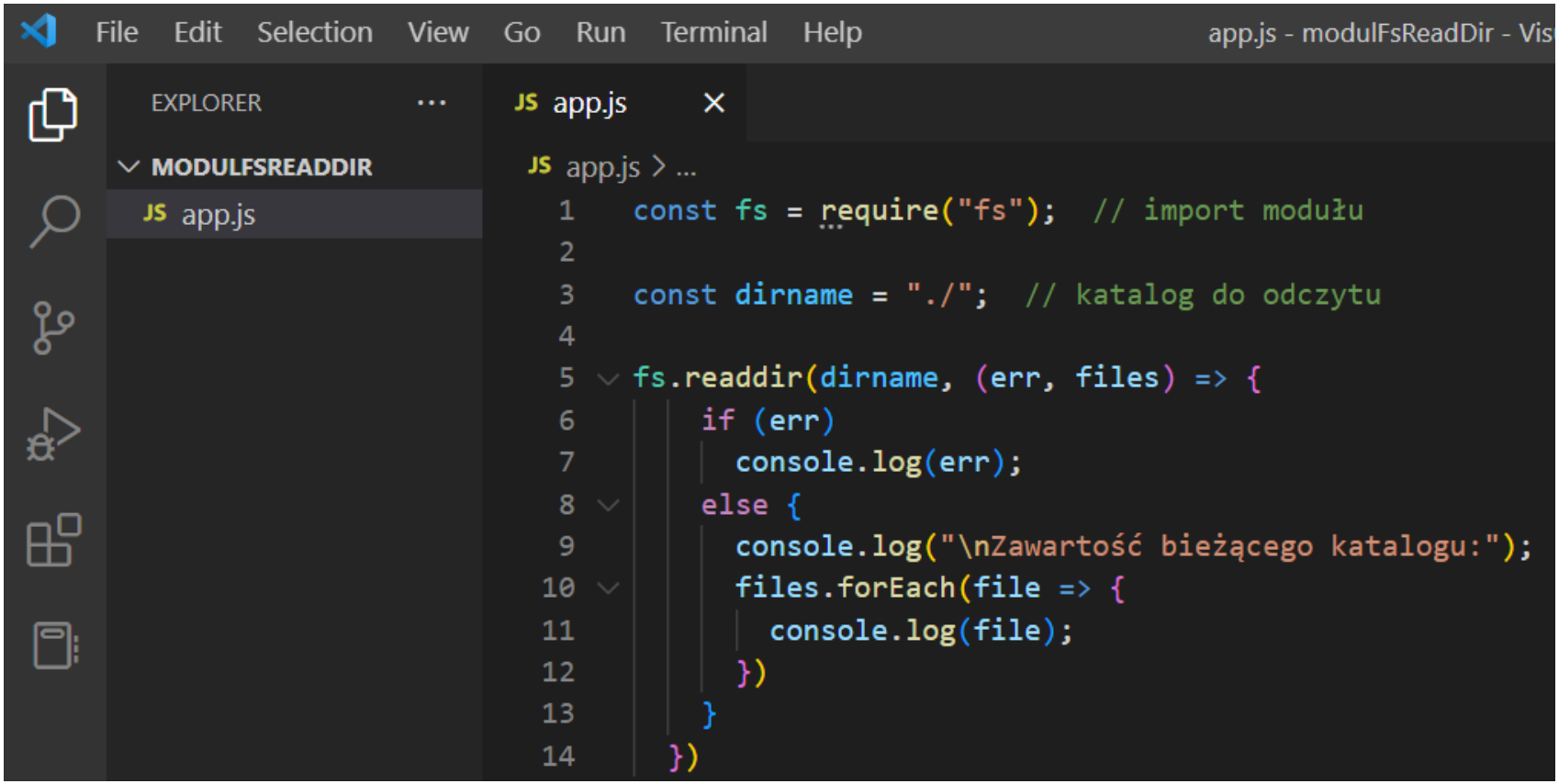
https://www.w3schools.com/nodejs/ref_url.asp

moduł fs

dostęp do plików

zawartość katalogu

readdir



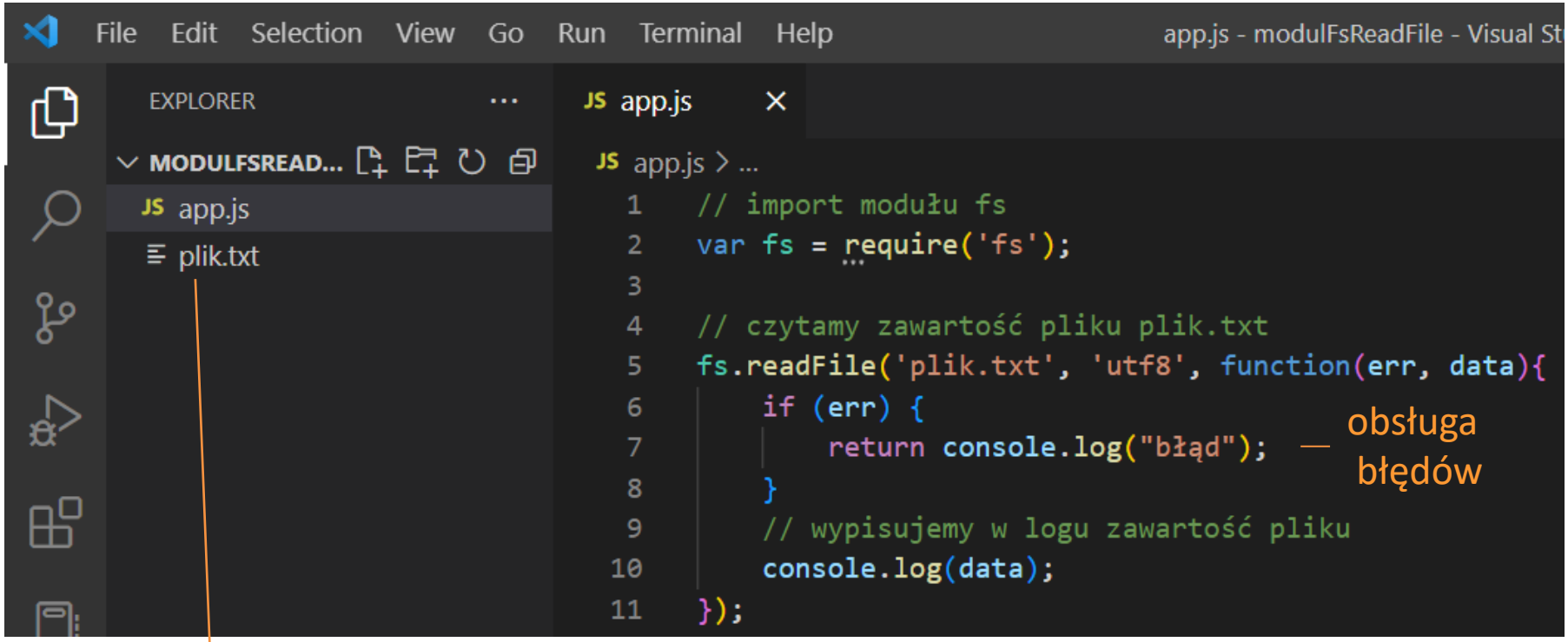

```
PS D:\> node app.js

Zawartość bieżącego katalogu:
app.js
```

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

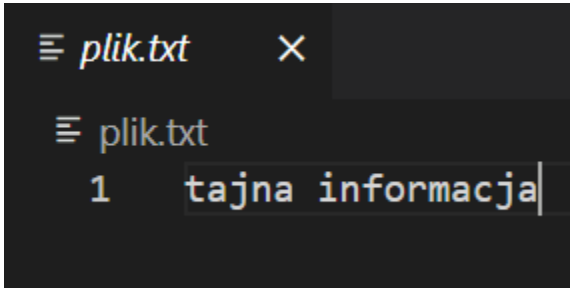
moduł fs
readfile

zawartość pliku

dostęp do plików



obsługa błędów




```
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

EXPLORER

MODULFSWRITEFILE

app.js

plik.txt

plik.txt

1 tajna informacja

app.js

```
// import modułu fs
const fs = require('fs');

let data = "Druga tajna informacja";

fs.writeFile("plik.txt", data, (err) => {
  if (err)
    console.log(err);
  else {
    console.log("Zapis się udał (ale zawartość pliku została nadpisana)\n");
    console.log("Zawartość pliku po zapisie:");
    console.log(fs.readFileSync("plik.txt", "utf8"));
  }
});
```

dane do zapisu

obsługa błędów

plik.txt

plik.txt

1 Druga tajna informacja

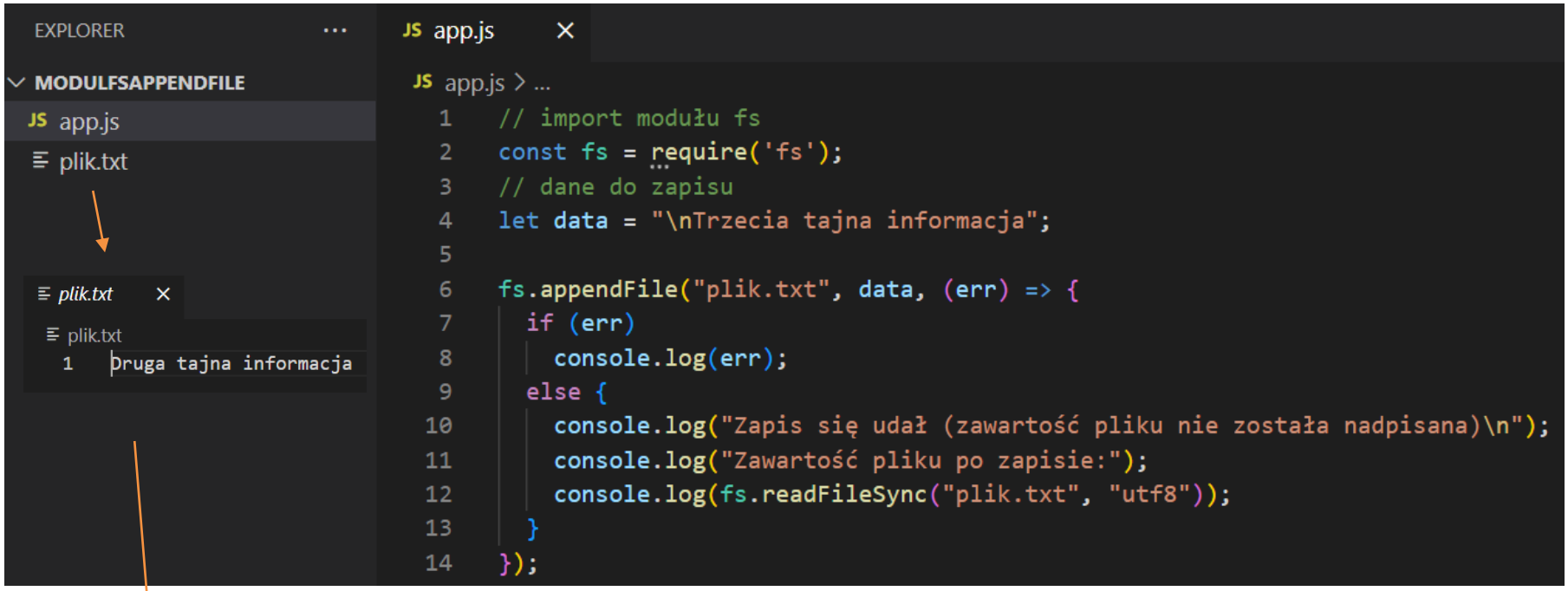
```
PS D:\> node app.js
Zapis się udał (ale zawartość pliku została nadpisana)

Zawartość pliku po zapisie:
Druga tajna informacja
```

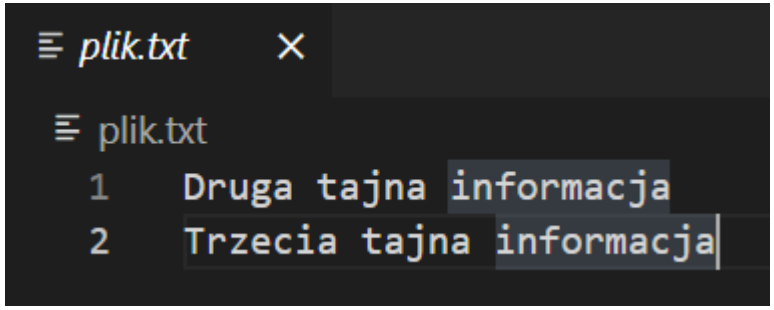
zawartość pliku plik.txt
(nadpisanie zawartości)

moduł fs
appendfile

dostęp do plików

zapis do pliku
(dopisanie zawartości)






```
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

MODULPATH

app.js

app.js

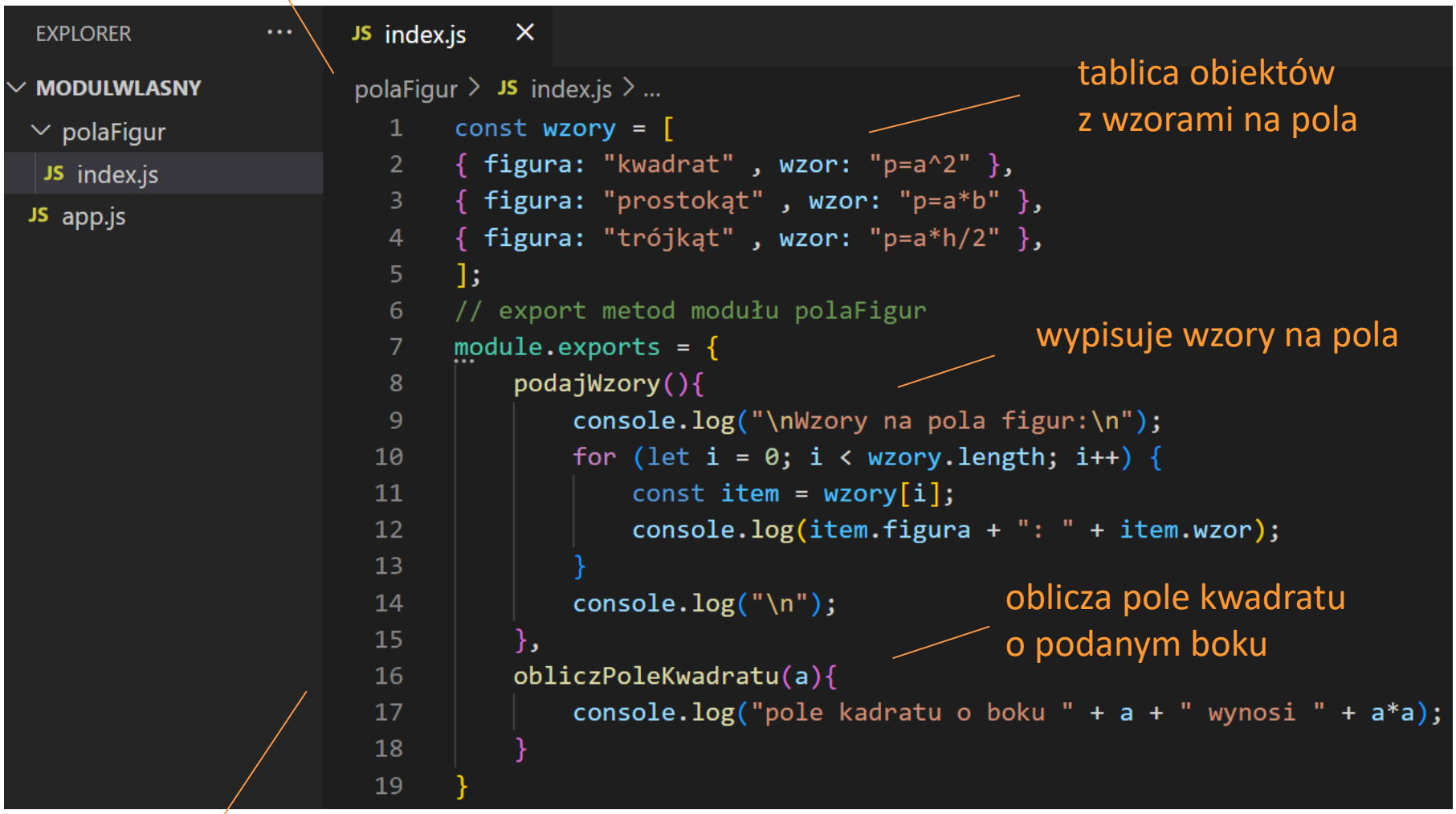
```
1   // import modułu path
2   const path = require('path');
3   // ścieżka do pliku
4   const filePath = '/a/b/main.js';
5   // wyciąga nazwa pliku z podanej ścieżki
6   const filename = path.basename(filePath);
7   // wyciąga nazwę katalogu z podanej ścieżki
8   const dirname = path.dirname(filePath);
9   // dokłada 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              filePath: ${filePath}
18              filename: ${filename}
19              dirname:  ${dirname}
20              join:  ${join}
21              isAbsolute:  ${isAbsolute}
22  `);
```

```
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
```

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

własny moduł polaFigur

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



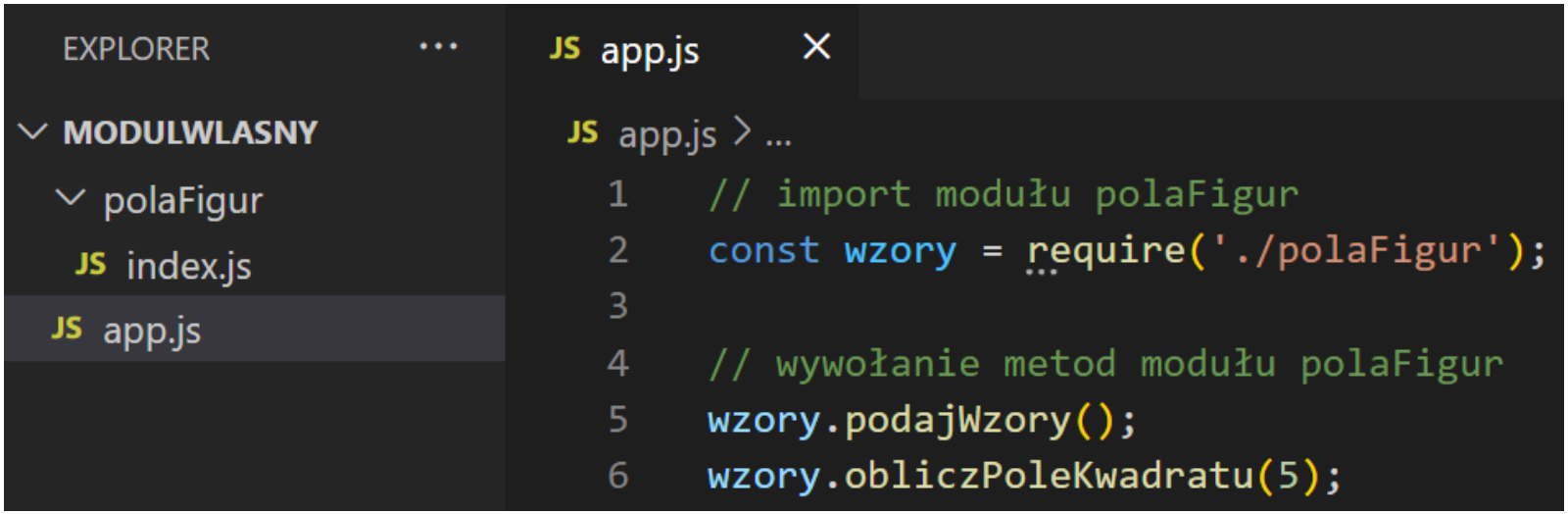
tablica obiektów
z wzorami na pola

wypisuje wzory na pola

oblicza pole kwadratu
o podanym boku

moduł polaFigur ma 2 metody: podajWzory oraz obliczPoleKwadratu

własny moduł polaFigur




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

Wzory na pola figur:

kwadrat: p=a^2
prostokąt: p=a*b
trójkąt: p=a*h/2


pole kadratu o boku 5 wynosi 25
```

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

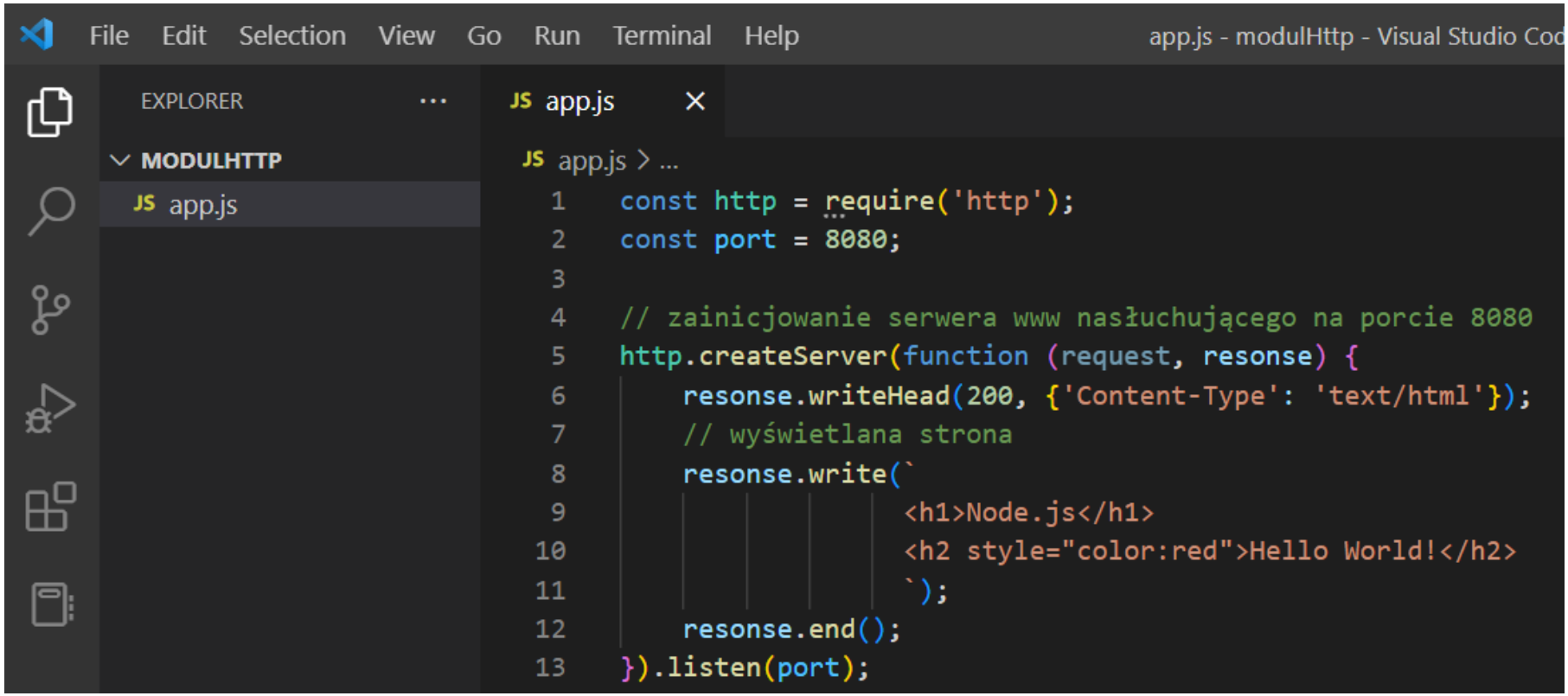
metoda podajWzory wypisuje wzory na pola figur

metoda obliczPoleKwadratu(bok) oblicza pole kwadratu o podanym boku

prosty serwer http

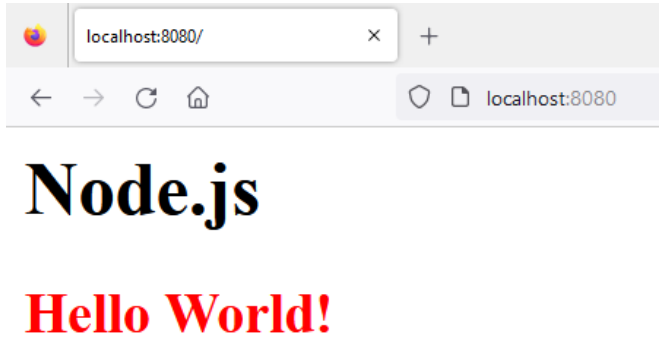
moduł http

serwer http



uruchomienie serwera www

```
node app.js
```



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

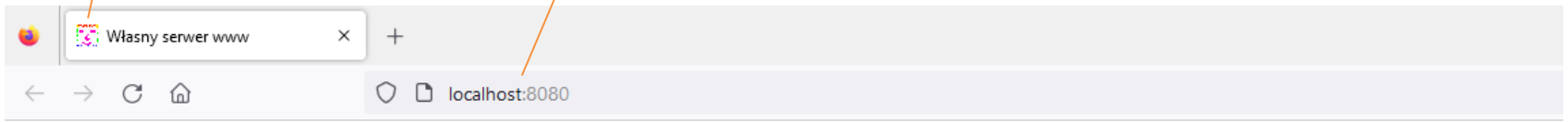
serwer http

serwer http wyświetlający pliki strony

serwer nasłuchuje lokalnie na porcie 8080

https://favicon.io/

favicon.ico



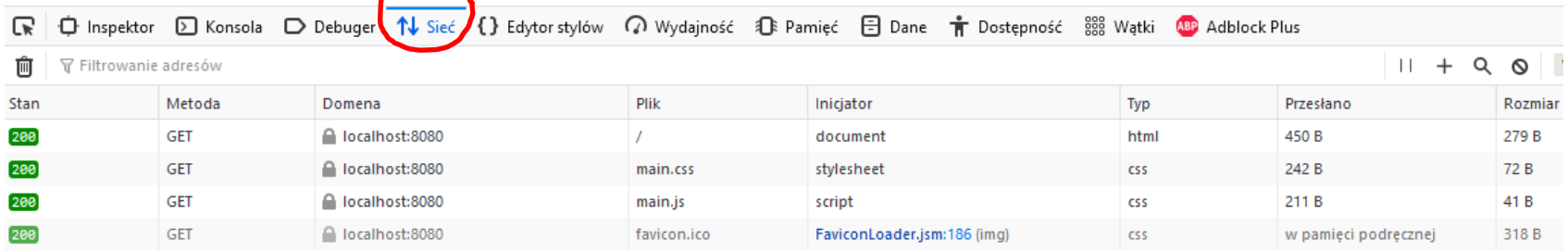
nawet działa

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

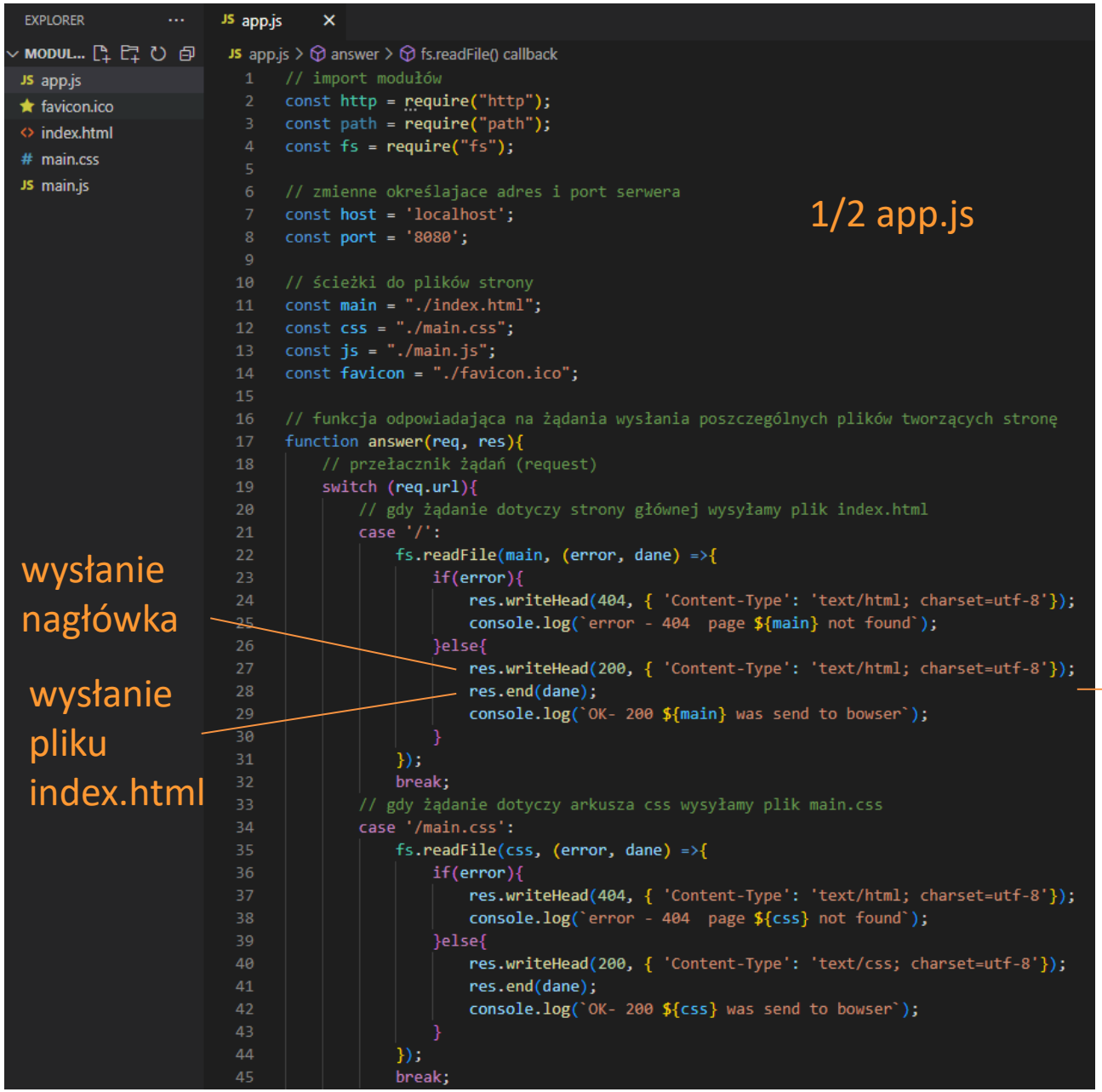
index.html
main.css
main.js
favicon.ico

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




```
// import modułów
const http = require("http");
const path = require("path");
const fs = require("fs");

// zmienne określajace adres i port serwera
const host = 'localhost';
const port = '8080';

// ścieżki do plików strony
const main = "./index.html";
const css = "./main.css";
const js = "./main.js";
const favicon = "./favicon.ico";

// funkcja odpowiadająca na żądania wysłania poszczególnych plików tworzących stronę
function answer(req, res){
    // przełacznik żądań (request)
    switch (req.url){
        // gdy żądanie dotyczy strony głównej wysyłamy plik index.html
        case '/':
            fs.readFile(main, (error, dane) =>{
                if(error){
                    res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
                    console.log(`error - 404  page ${main} not found`);
                }else{
                    res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8'});
                    res.end(dane);
                    console.log(`OK- 200 ${main} was send to bowser`);
                }
            });
            break;
        // gdy żądanie dotyczy arkusza css wysyłamy plik main.css
        case '/main.css':
            fs.readFile(css, (error, dane) =>{
                if(error){
                    res.writeHead(404, { 'Content-Type': 'text/html; charset=utf-8'});
                    console.log(`error - 404  page ${css} not found`);
                }else{
                    res.writeHead(200, { 'Content-Type': 'text/css; charset=utf-8'});
                    res.end(dane);
                    console.log(`OK- 200 ${css} was send to bowser`);
                }
            });
            break;
```

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 bowser`);
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 bowser`);
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

MODULHTTP2
- app.js
- favicon.ico
- index.html
- main.css
- main.js

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8'>
    <title>Własny serwer www</title>
    <link rel='stylesheet' type='text/css' media='screen' href='main.css'>
    <script src='main.js'></script>
</head>
<body>
    <h1>Własny serwer www</h1>
</body>
</html>
```

main.css

```
h1{
    background-color: blue;
    color: white;
    width: 300px;
}
```

main.js

```
document.write("<p>nawet działa</p>");
```

https://favicon.io/

favicon.ico
16x16
png

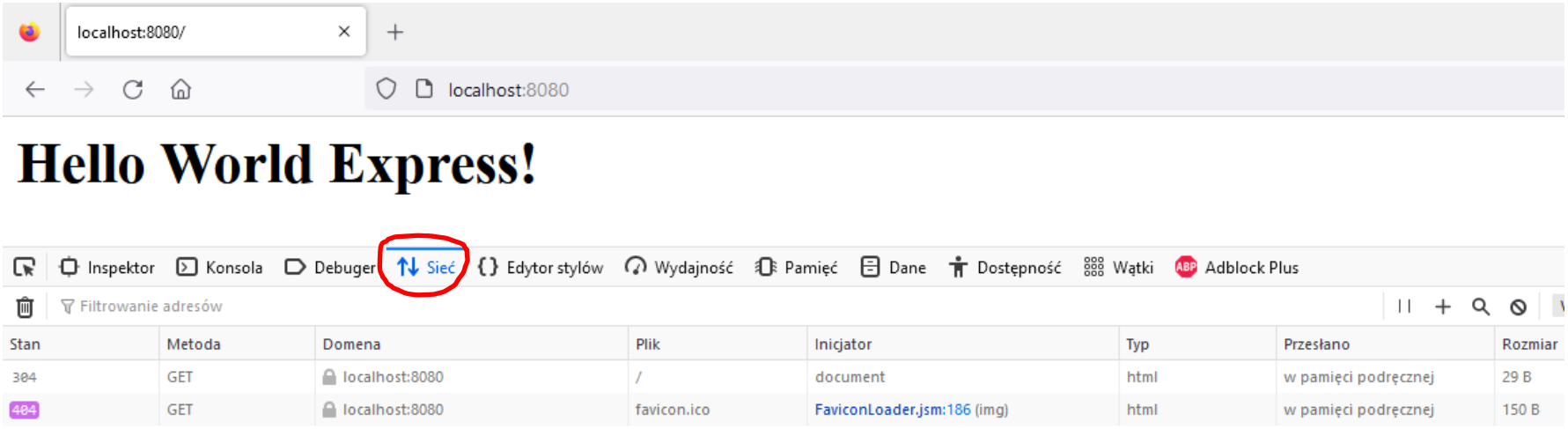
prosty serwer http

Express
serwer http

framework webowy

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

instalacja frameworka



serwer http w oparciu o framework Express

Developer Tools F12

prosty serwer http

Express serwer http

framework

kod serwera http w oparciu o framework Express j

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

pliki konfiguracyjne framworka Express są generowane po jego instalacji

uruchomienie skryptu

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

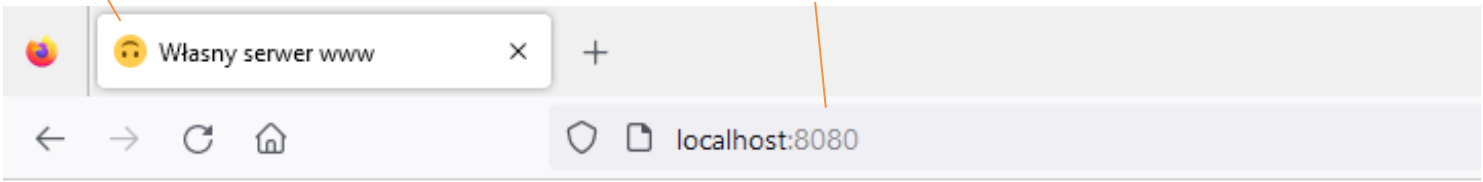
Express serwer http

serwer http wyświetlający pliki strony

framework

favicon.ico https://favicon.io/

serwer nasłuchuje lokalnie na porcie 8080



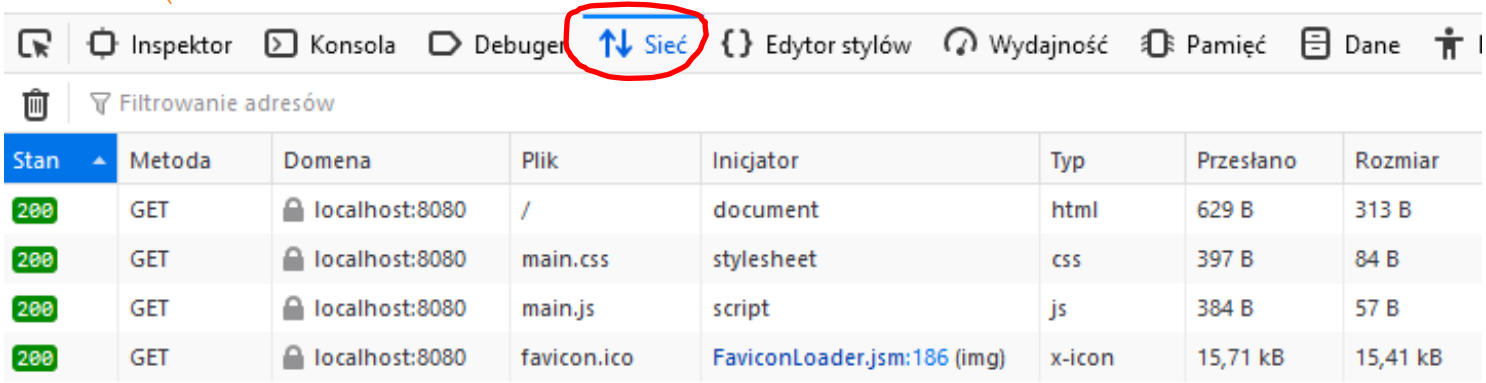
nawet działa

wynik działania skryptu w pliku main.js

Własny serwer www oparty na Node.js i Express

styl umieszczony w odrębnym pliku main.css

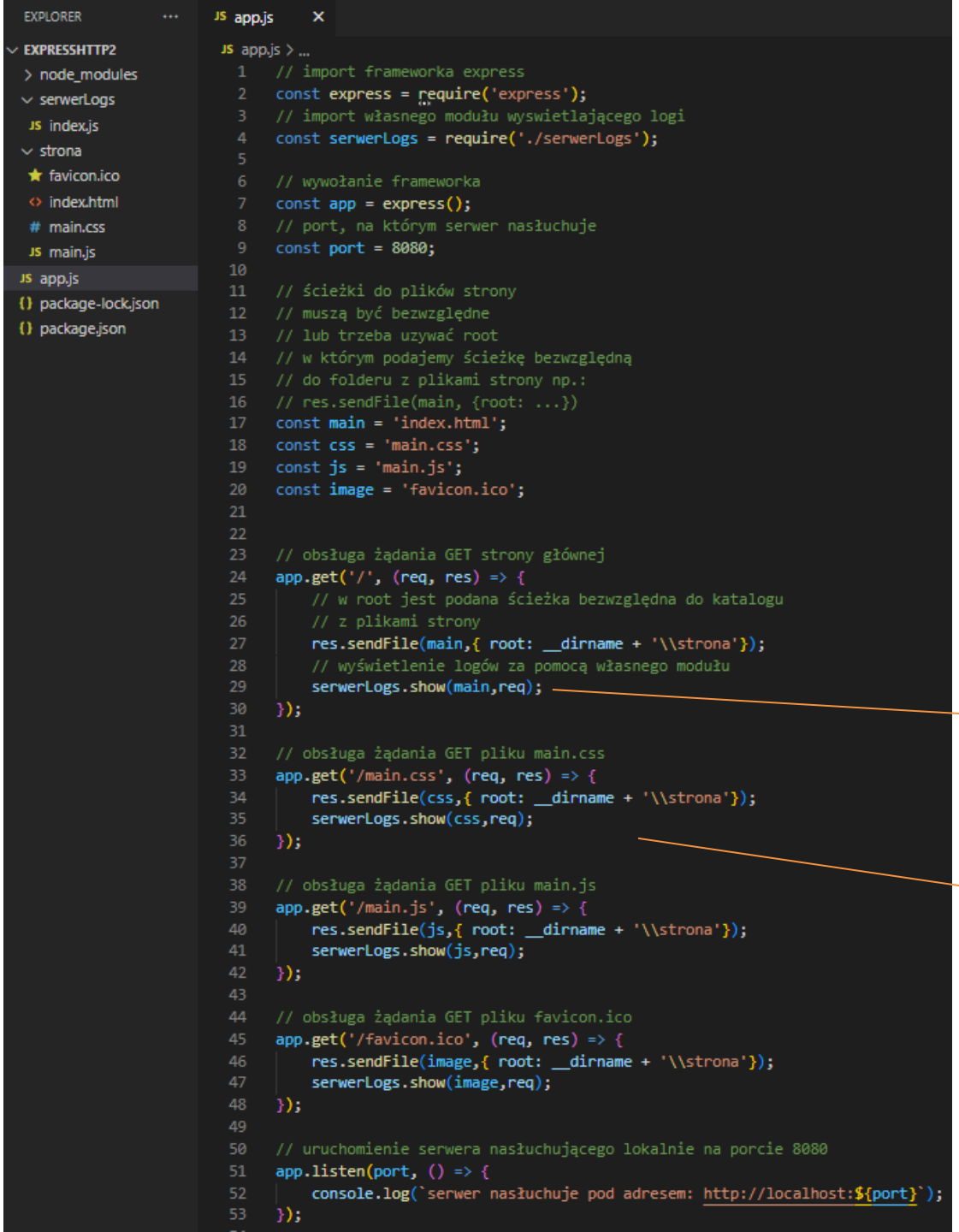
żądania GET przeglądarki dotyczące poszczególnych plików na które składa się strona



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

Developer Tools F12

Express serwer http



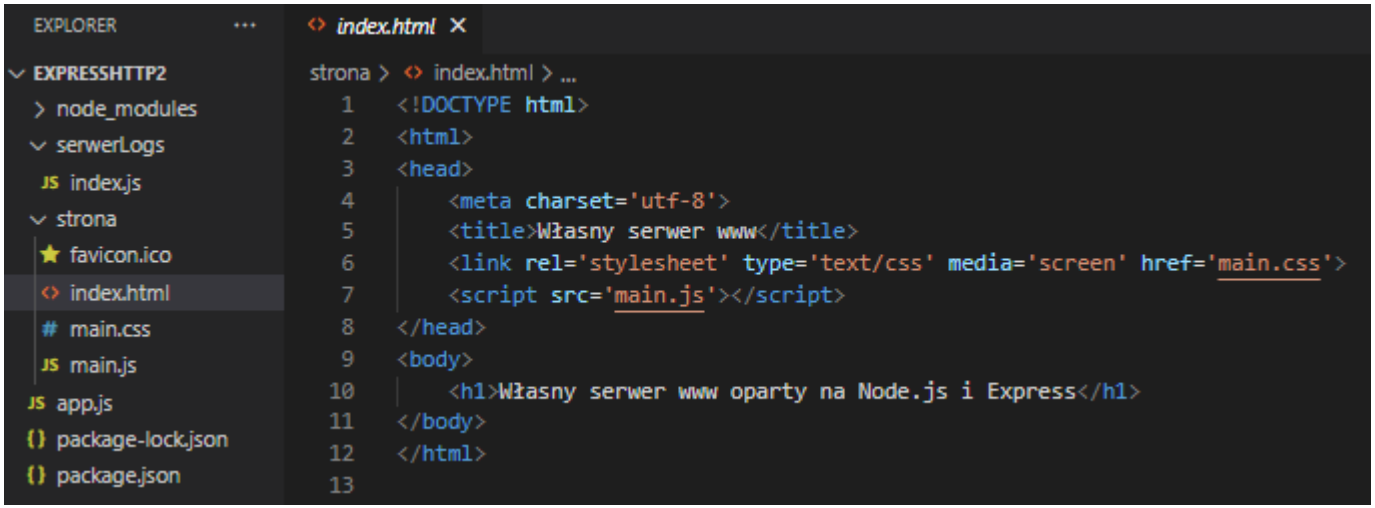
app.js

logi serwera

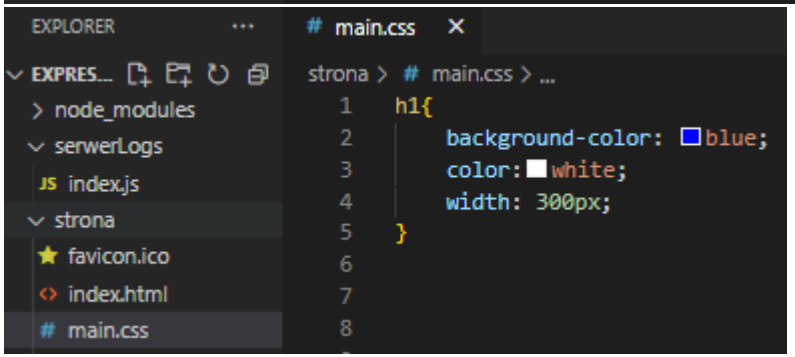
obsługa żądań GET poszczególnych plików strony – wysłanie żądanej strony do przeglądarki

Express serwer http

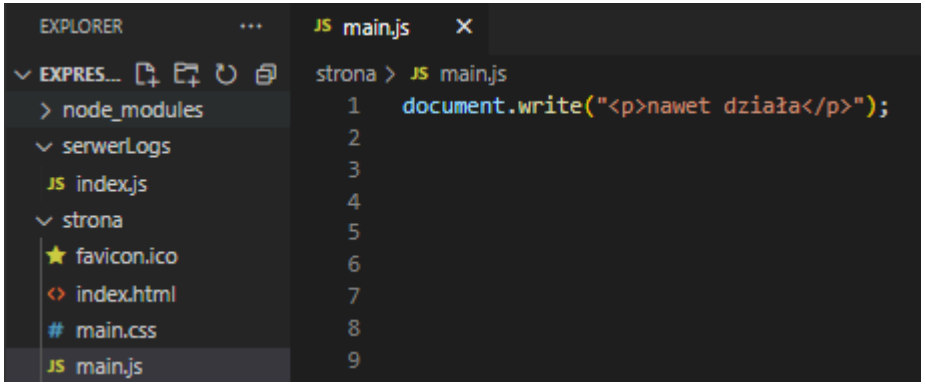
index.html



main.css

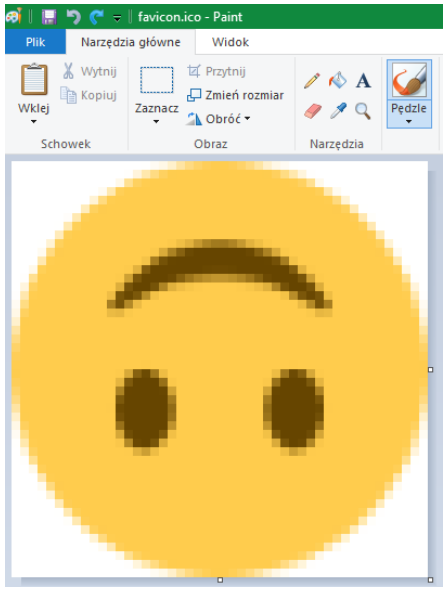


main.js



https://favicon.io/

favicon.ico
48x48



Express
serwer http

własny moduł wyświetlający logi serwera znajduje się w katalogu serwerLogs

kod znajduje się w pliku index.js

metoda show wyświetlająca logi serwera

Express
serwer http

logi serwera wygenerowane za pomocą własnego modułu serwerLogs

```
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

        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

        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

        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
```

wysłany plik

protokół

czy użyto https

ścieżka adresu URL, polecenie Node.js

ścieżka adresu URL, w przypadku przekierowania zachowuje oryginalną ścieżkę

ścieżka adresu URL, zawiera ostatnią część adresu URL

adres IP klienta (IPv6 oraz IPv4)

Google wyszukiwanie informacji

początek definicji parametrów

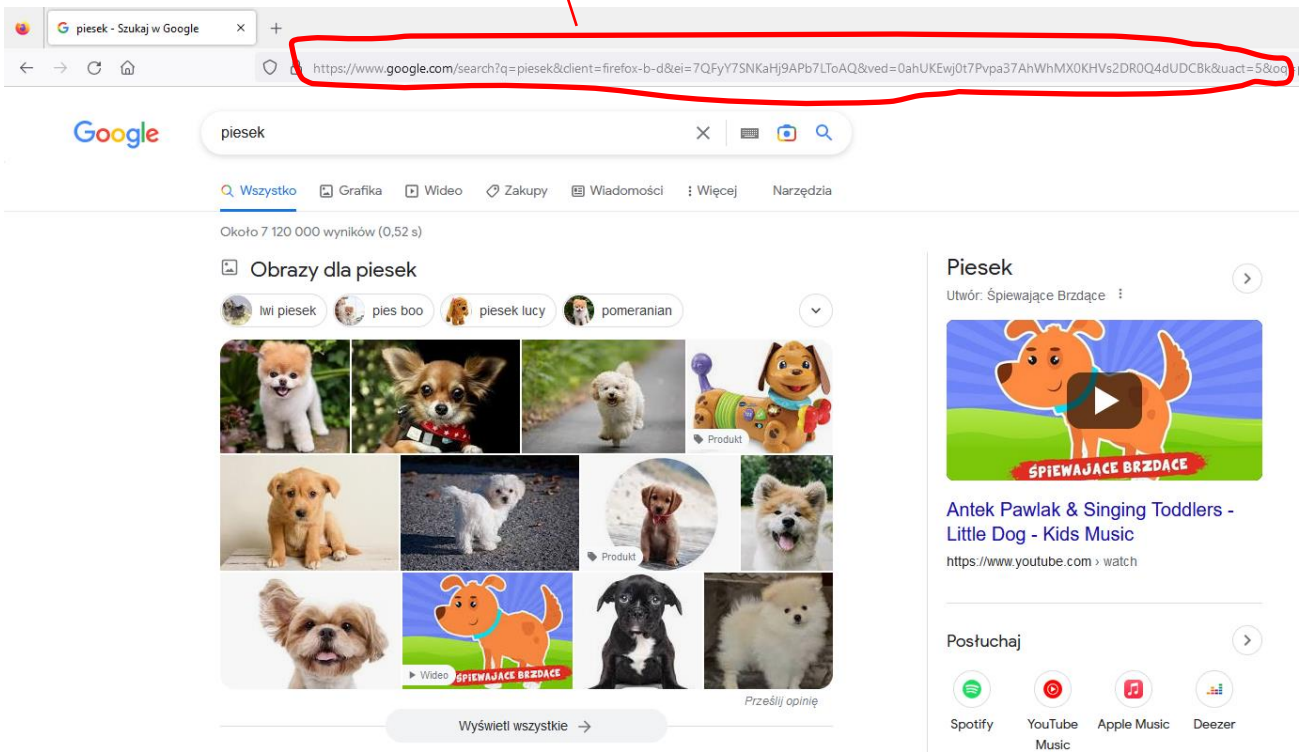
adres URL

nazwa parametru

wartość parametru

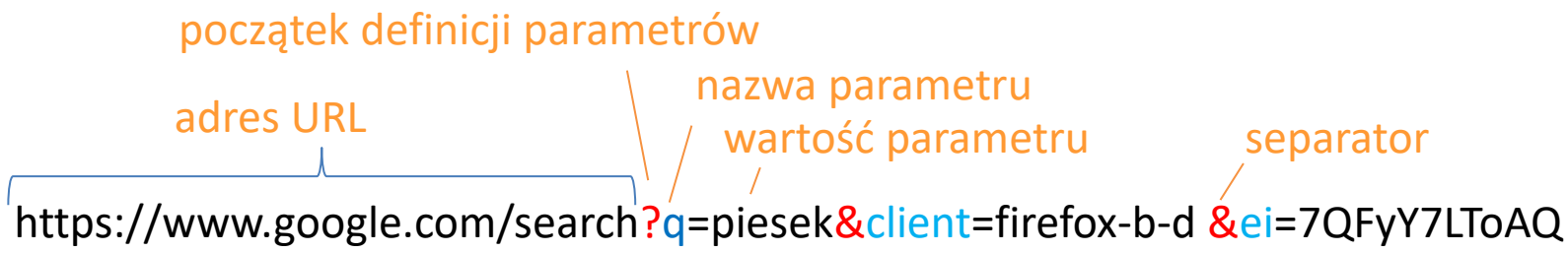
separator

https://www.google.com/search?q=piesek&client=firefox-b-d&ei=7QFyY7SNKaHj9APb7LToAQ&ved=0ahUKEwj0t7Pvpa37AhWhMX0KHVs2DR0Q4dUDCBk&uact=5&oq=piesek&gs_lcp=Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIOCC4QgAQQsQMQgwEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAEELEDEIMBMggIABCxAxCDATILCC4QgAQQsQMQgwEyCwgAEIAEELEDEIMBMgsIABCABBCxAxCDATIFCC4QgAQ6CggAEEcQ1gQQsAM6BwgAELADEEM6DAguEMgDELADEEMYATоFCAAQgAQ6EAguELEDEIMBEMcBENEDEEM6CAguELEDEIMBOgQIABBDOgoILhDHARDRAxBDOgQILhBDOggILhCABBDUAjoKCC4QsQMQgwEQQzoHCC4Q1AIQQzoQCC4QgwEQ1AIQsQMQgAQQQzoHCAAQsQMQQzoLCC4QgAQQsQMQ1AI6DQguELEDEIMBENQCEENKBAhBGABKBAhGGAFQ_A1YzxZgpBtoAnABeACAAY4CiAGTDJIBAzItNpgBAKABAcgBEcABAdoBBggBEAEYCA&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



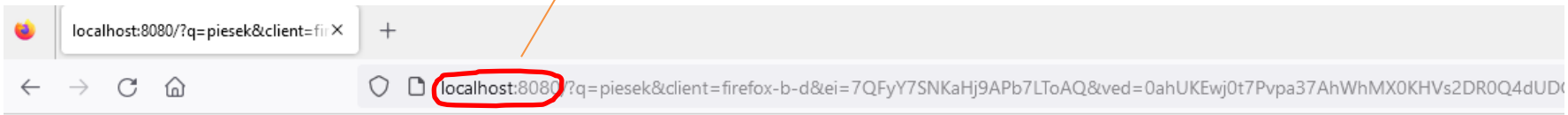

```
// obsługa żądania GET strony głównej
app.get('/', (req, res) => {

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

    // drugi sposób - destrukturyzacja
    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

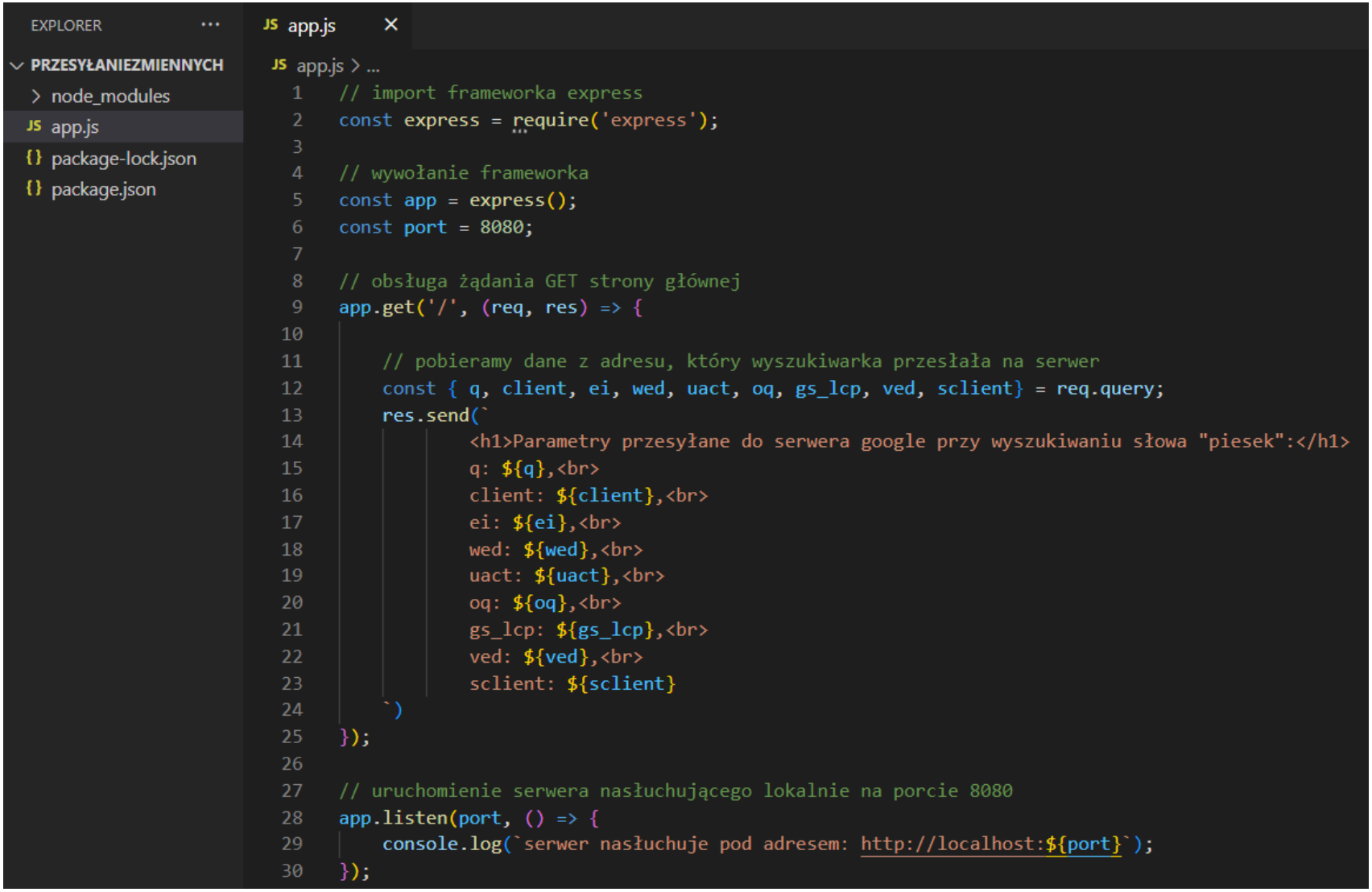
adres google zamieniamy na localhost:8080



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:
Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIOCC4QgAQQsQMQgwEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAEELEDEIMBI
ved: 0ahUKEwj0t7Pvpa37AhWhMX0KHVs2DR0Q4dUDCBk,
sclient: gws-wiz-serp

Express przesyłanie informacji

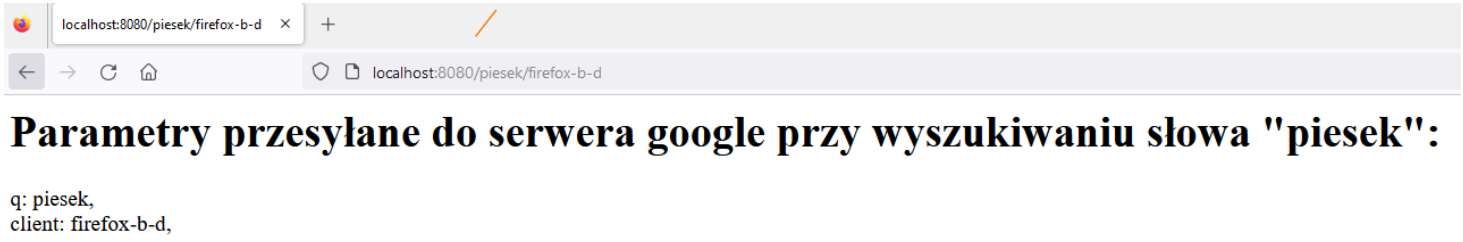


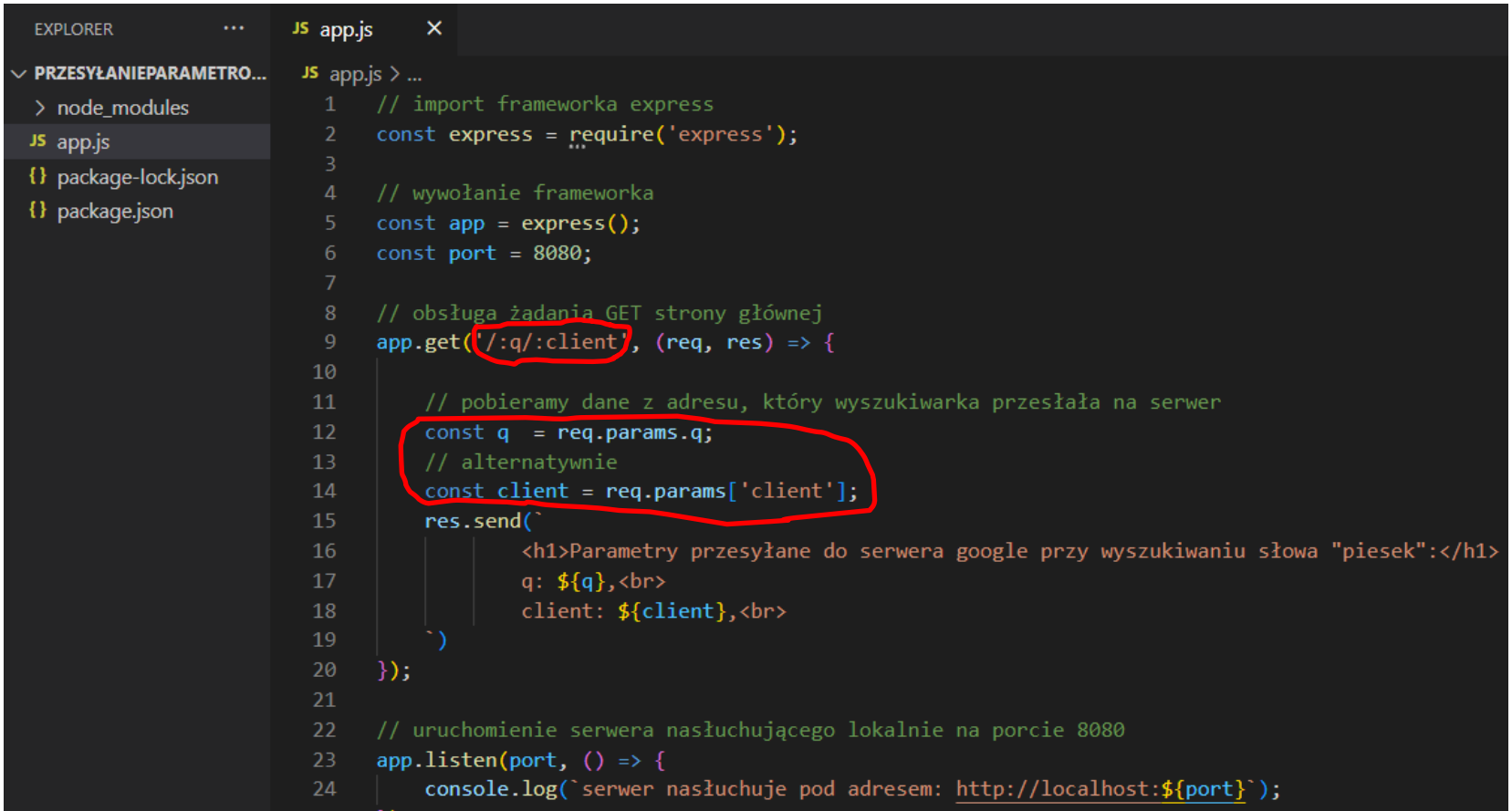
Express przesyłanie informacji

inny sposób doklejenia parametrów do adresu

wartości parametrów q oraz client

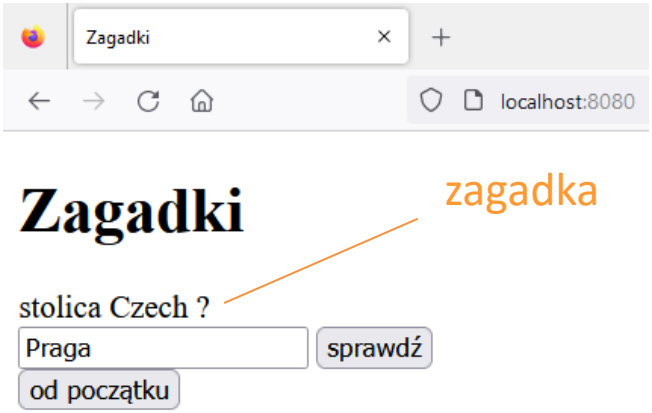
http://localhost:8080/piesek/firefox-b-d





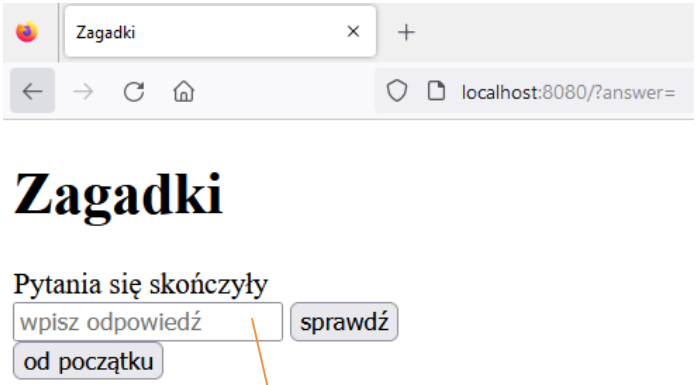
zadanie

utwórz stronę z zagadkami

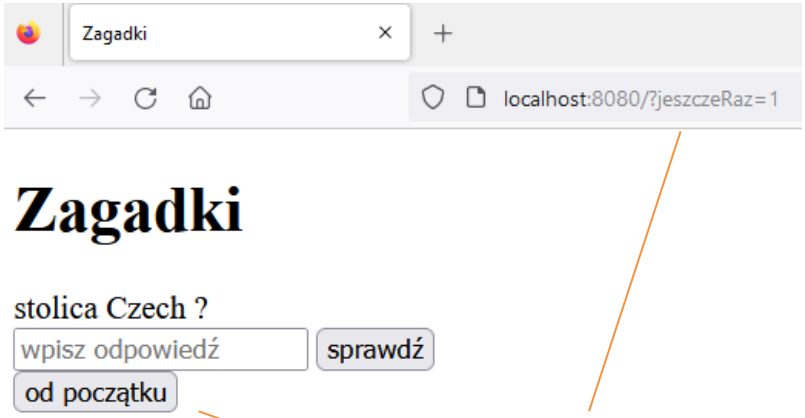


formularz przesyła dane metodą get

weryfikacja odpowiedzi – gdy użytkownik zgadnie wyświetla się następna zagadka

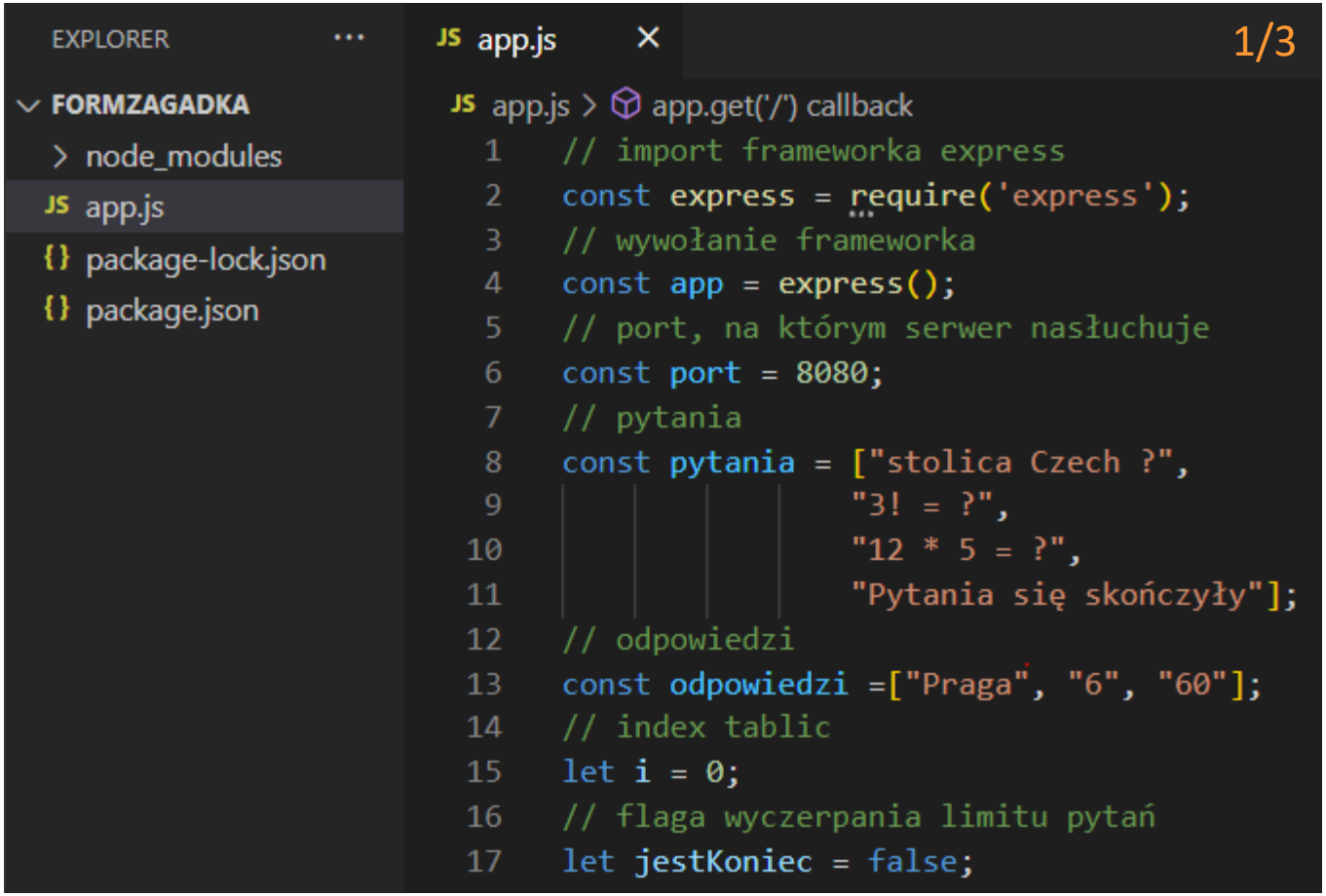


komunikat końcowy gdy pytania się wyczerpią



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

odpowiedź do zadania



zmienne wykorzystane na stronie

odpowiedź do zadania

2/3

```
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      // jeszczRaz = 1
53      // resetujemy ustawinia
54      if (jeszczeRaz == "1"){
55          jestKoniec = false;
56          i=0;
57      }
```

pobieranie zmiennych wysłanych przez formularze na stronie

sprawdzenie odpowiedzi

obsługa przycisku „od początku”

odpowiedź do zadania

3/3

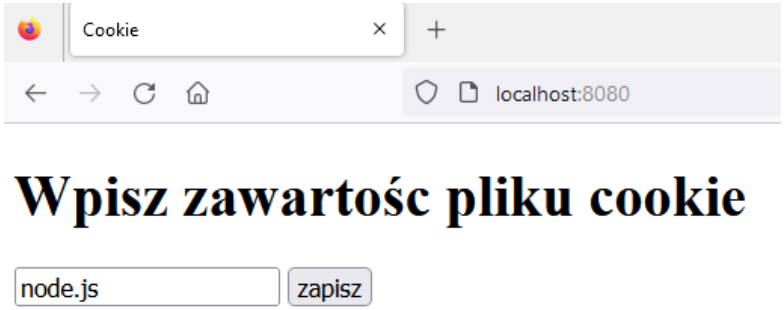
```
59      // strona główna
60      strona =`
61          <!doctype html>
62      <html>
63      <head>
64          <meta charset="utf-8">
65          <title>Zagadki</title>
66      </head>
67      <body>
68      <h1>Zagadki</h1>
69      ${pytania[i]}
70      <form name="zagadka"  method="get">
71          <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  });
```

wypisanie pytania na stronie głównej

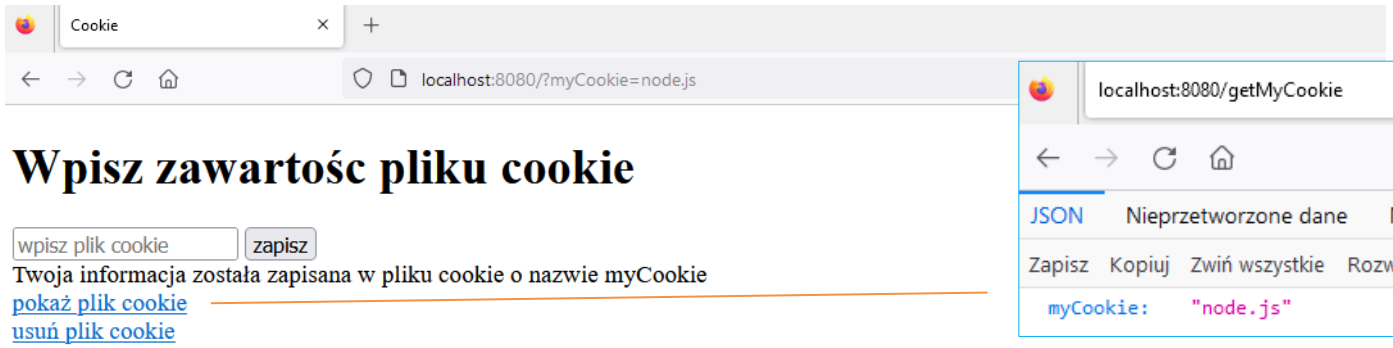
weryfikacja odpowiedzi na stronie głównej

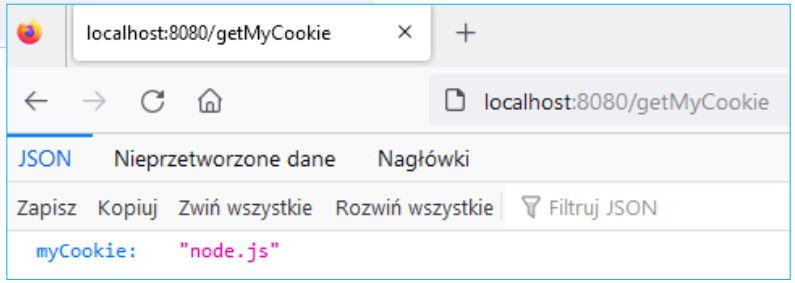
wysłanie strony do przeglądarki

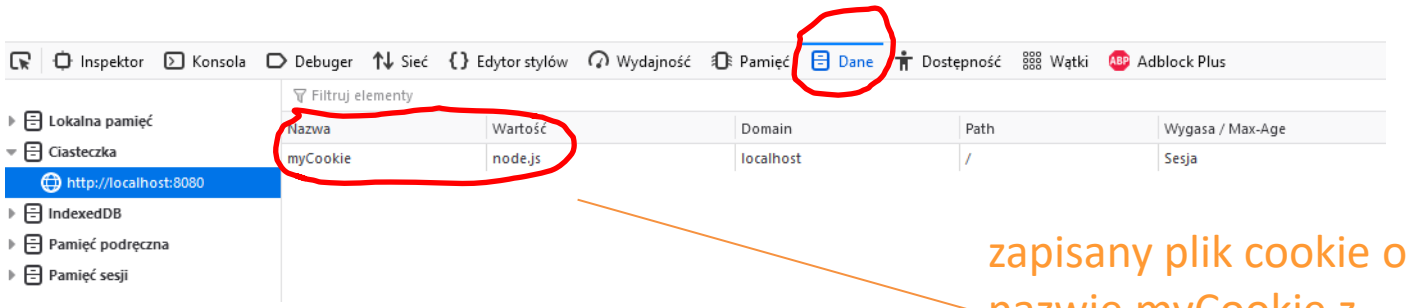
cookie



formularz do zapisywania pliku cookie







zapisany plik cookie o nazwie myCookie z zawartością z formularza

cookie

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

instalacja modułu cookie-parser

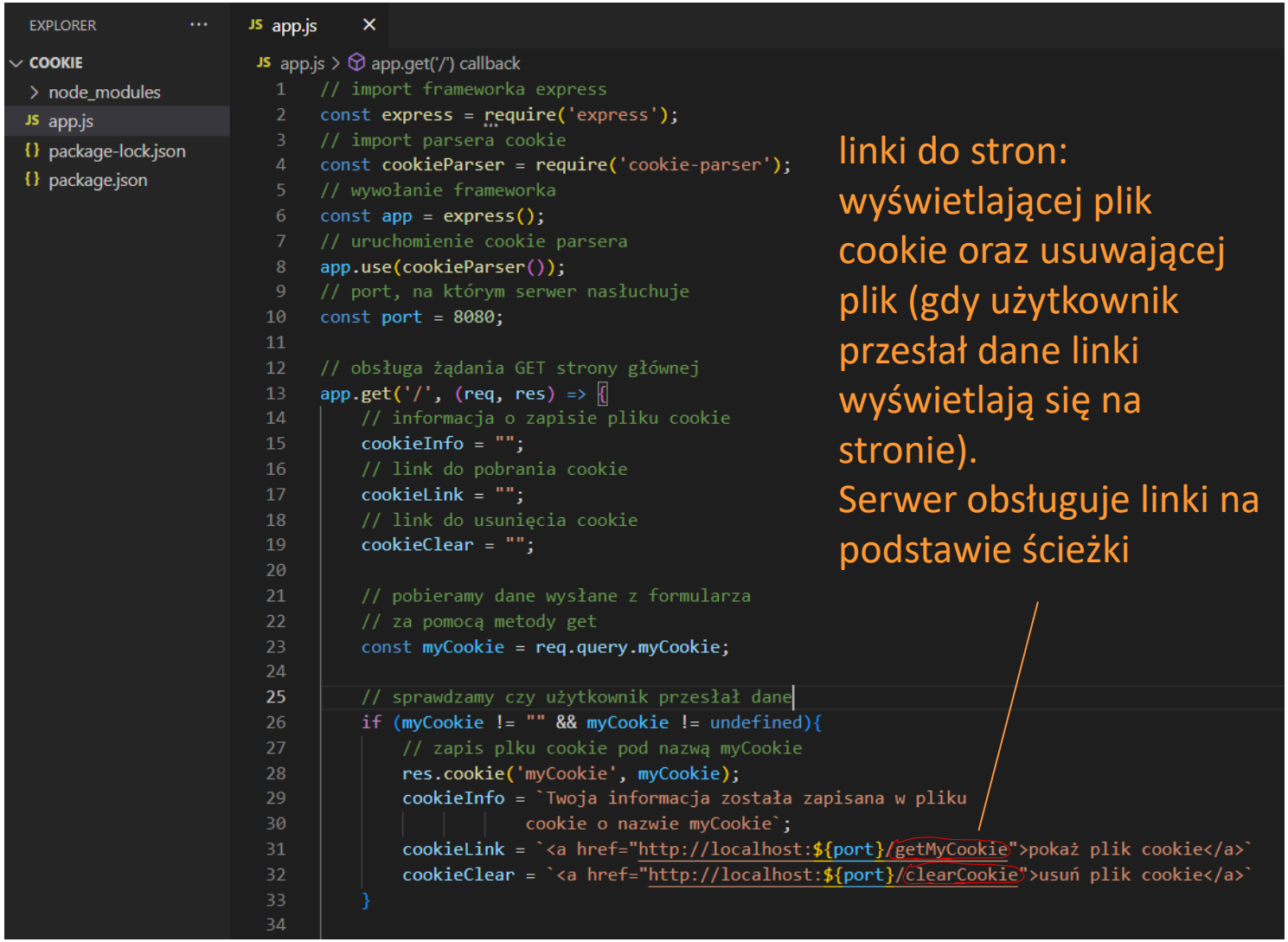
```
// import parsera cookie
const cookieParser = require('cookie-parser');
// uruchomienie cookie parsera
app.use(cookieParser());
```

użycie modułu cookie-parser w kodzie

```
// zapis plku 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



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 zawartośc 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

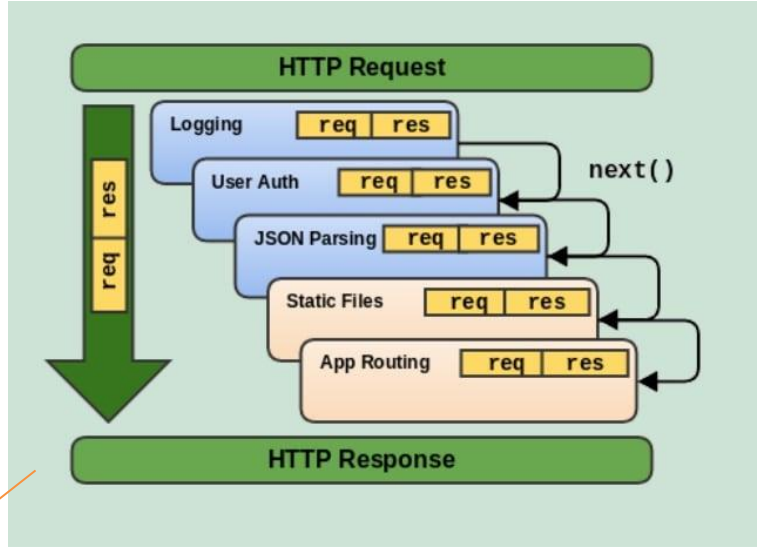
middleware

middleware to funkcje, które mają dostęp do obiektu żądania (req), obiektu odpowiedzi (res) i następnej funkcji middleware w cyklu żądanie-odpowiedź 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-odpowiedź
- wywołać następną funkcję middleware

middleware



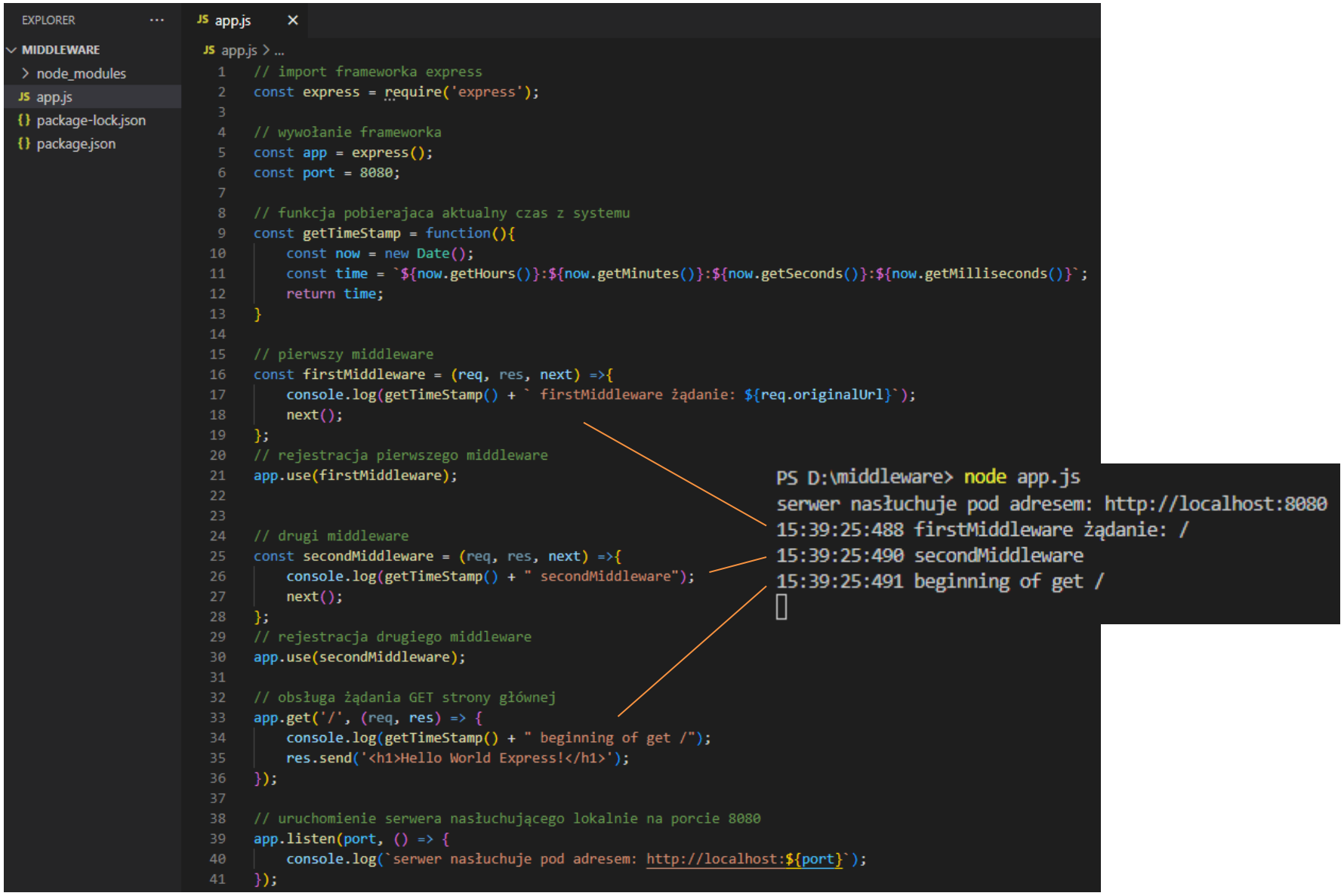
aplikacja Express to zasadniczo seria wywołań funkcji middleware.

przed wykonaniem ostatecznej procedury obsługi żądań wykonywane są funkcje middleware

https://expressjs.com/en/guide/using-middleware.html

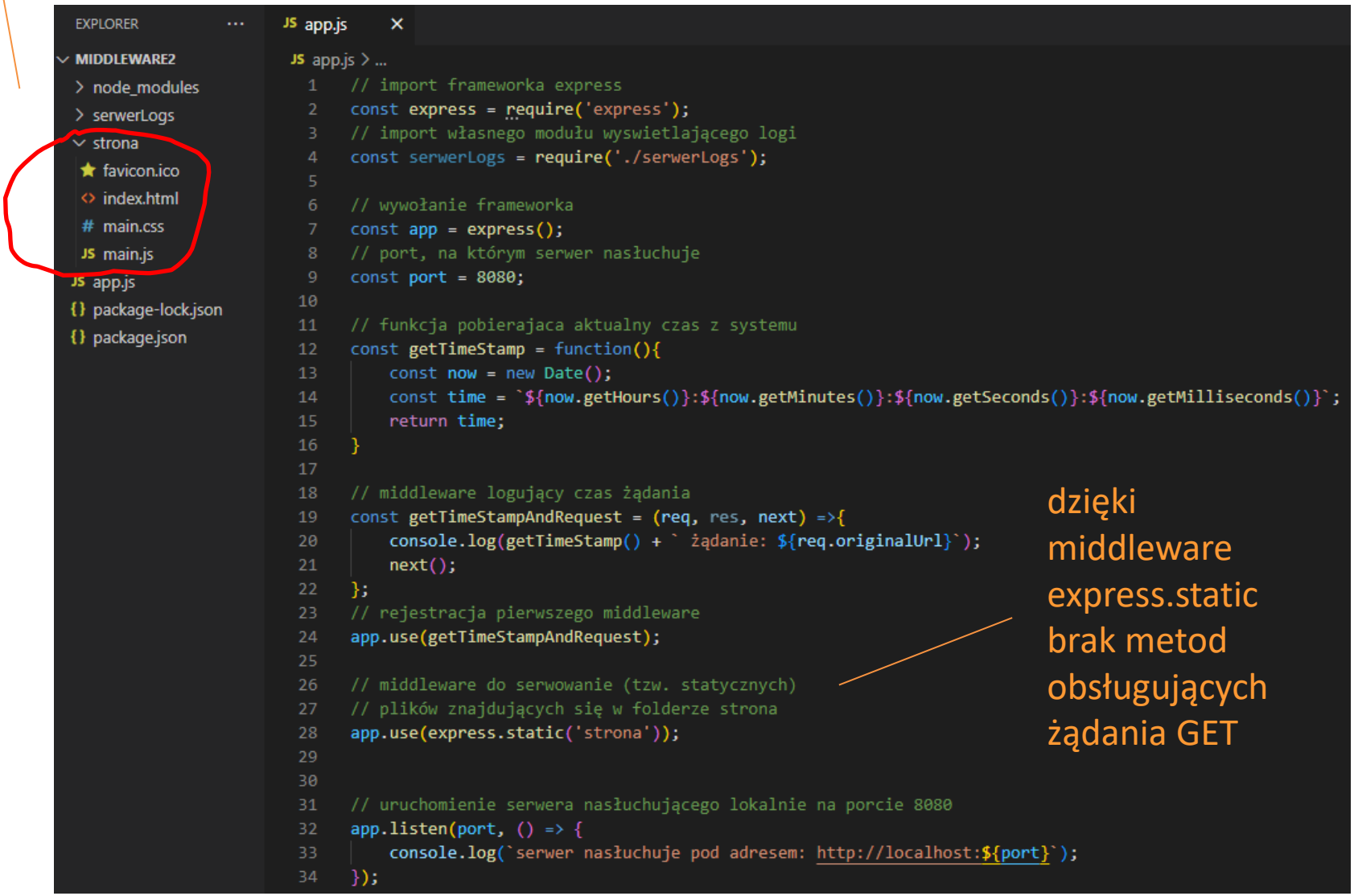
https://dev.to/ghvstcode/understanding-express-middleware-a-beginners-guide-g73

middleware


EXPLORER
app.js
MIDDLEWARE
node_modules
app.js
package-lock.json
package.json
app.js > ...
// import frameworka express
const express = require('express');
// wywołanie frameworka
const app = express();
const port = 8080;
// funkcja pobierajaca aktualny czas z systemu
const getTimeStamp = function(){
const now = new Date();
const time = `${now.getHours()}:${now.getMinutes()}:${now.getSeconds()}:${now.getMilliseconds()}`;
return time;
}
// pierwszy middleware
const firstMiddleware = (req, res, next) =>{
console.log(getTimeStamp() + ` firstMiddleware żądanie: ${req.originalUrl}`);
next();
};
// rejestracja pierwszego middleware
app.use(firstMiddleware);
// drugi middleware
const secondMiddleware = (req, res, next) =>{
console.log(getTimeStamp() + " secondMiddleware");
next();
};
// rejestracja drugiego middleware
app.use(secondMiddleware);
// obsługa żądania GET strony głównej
app.get('/', (req, res) => {
console.log(getTimeStamp() + " beginning of get /");
res.send('<h1>Hello World Express!</h1>');
});
// uruchomienie serwera nasłuchującego lokalnie na porcie 8080
app.listen(port, () => {
console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
});
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 /

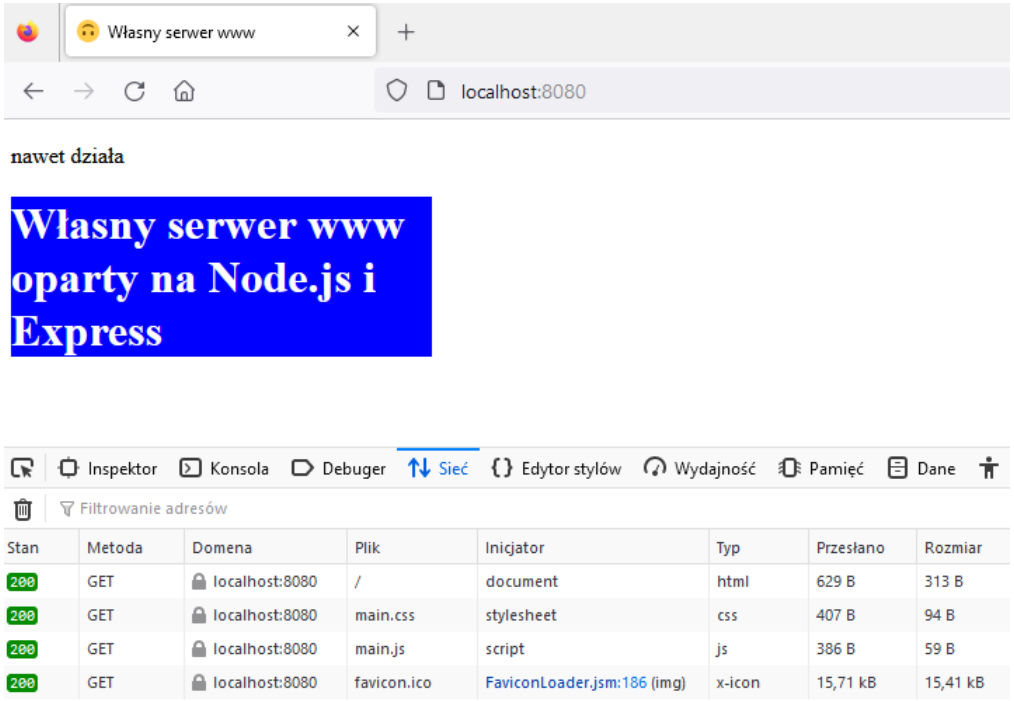
middleware static files

pliki statyczne w folderze strona



dzięki middleware express.static brak metod obsługujących żądania GET

middleware
static files



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

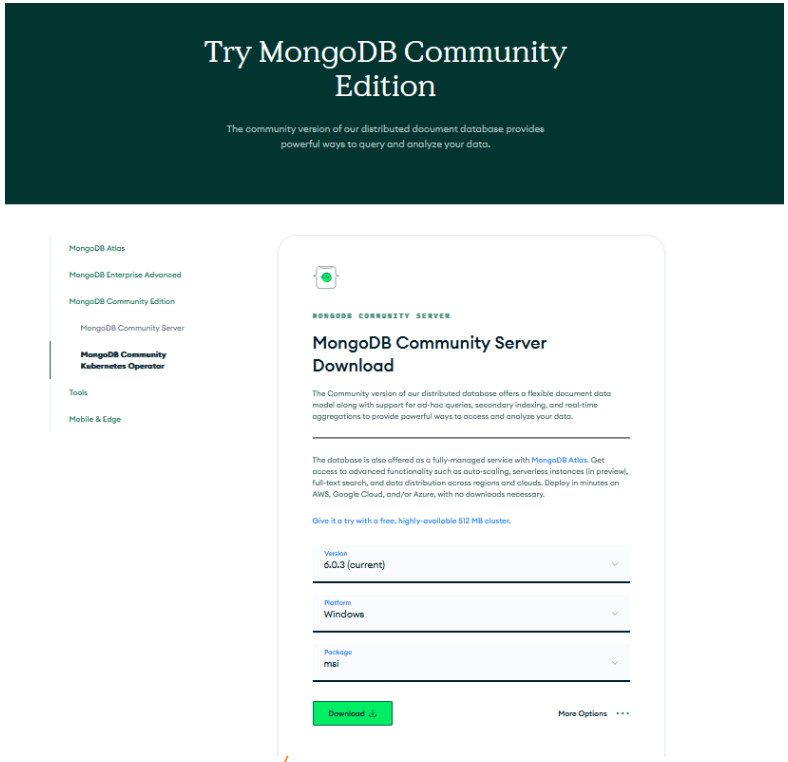
nierelacyjna (baza zawiera kolekcje,
w których przechowywane są dokumenty, baza typu noSQL, przykład: MongoDB)

https://www.mongodb.com/nosql-explained/nosql-vs-sql

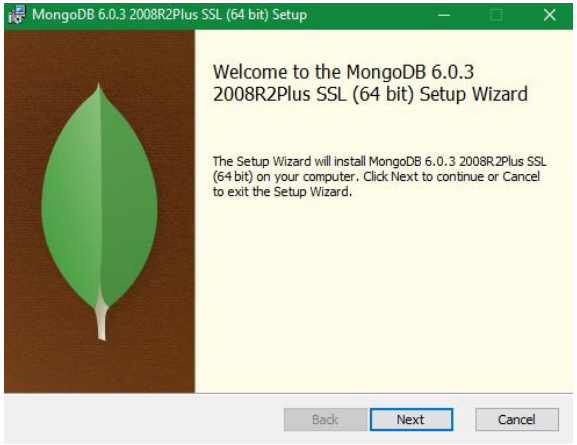
MongoDB

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



download MongoDB

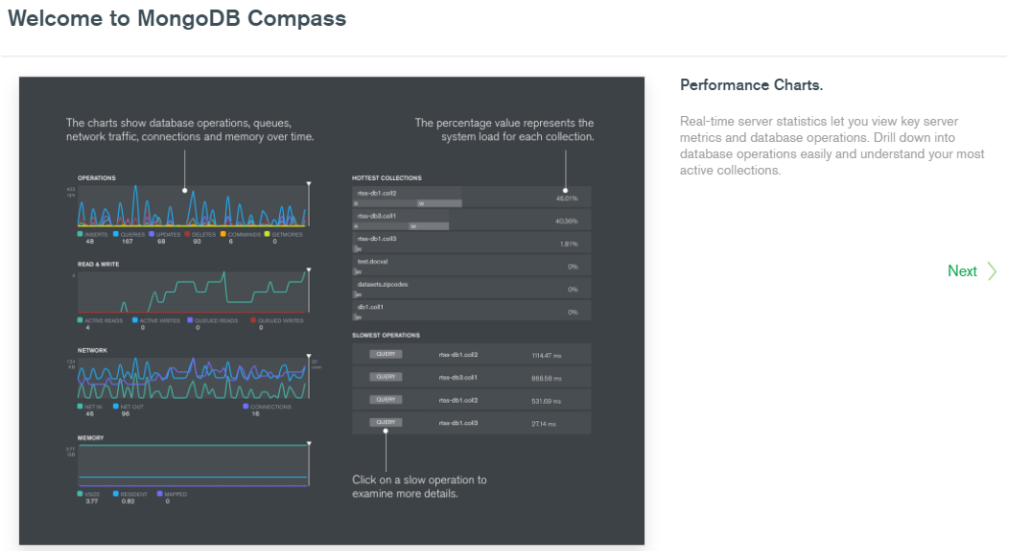


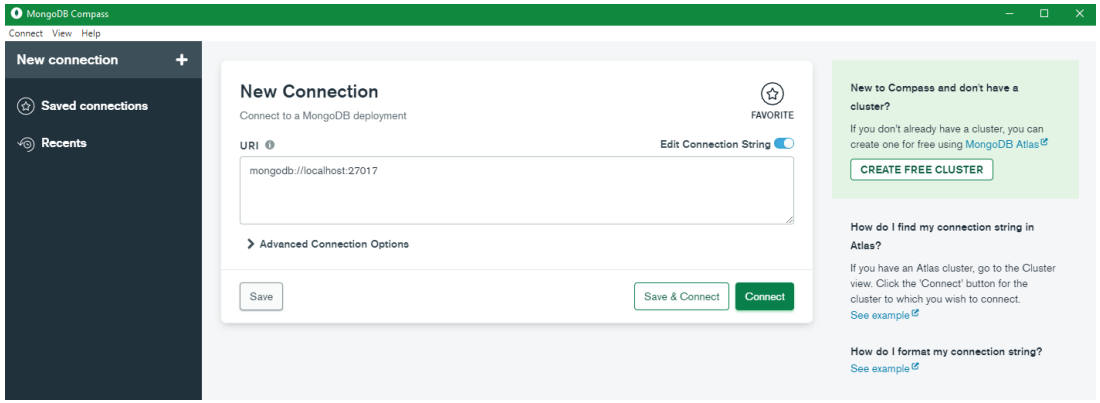
instalacja MongoDB

MongoDB nasłuchuje http://localhost:27017/

MongoDB

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

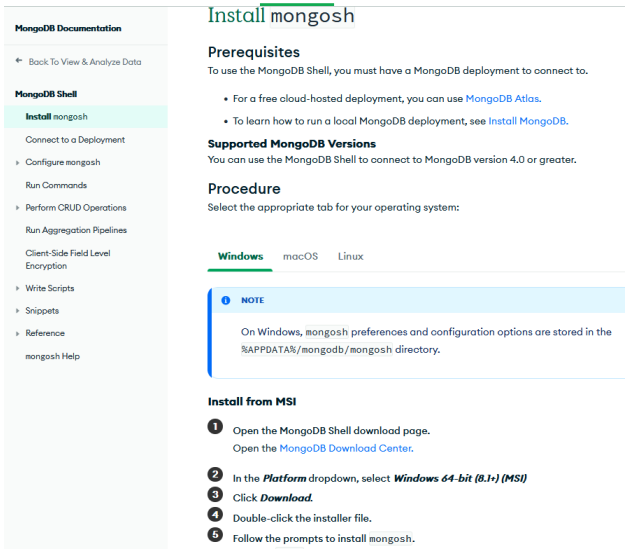


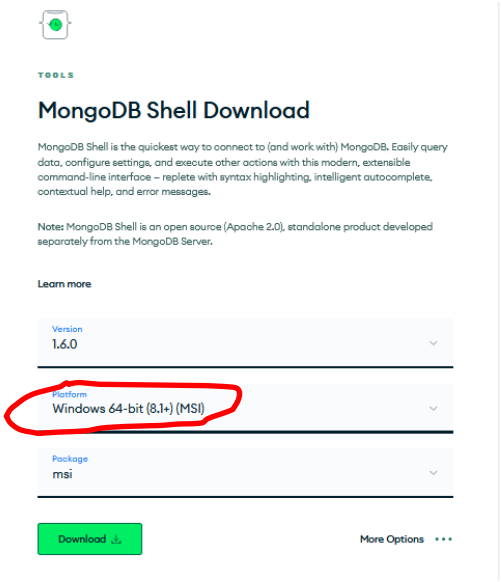


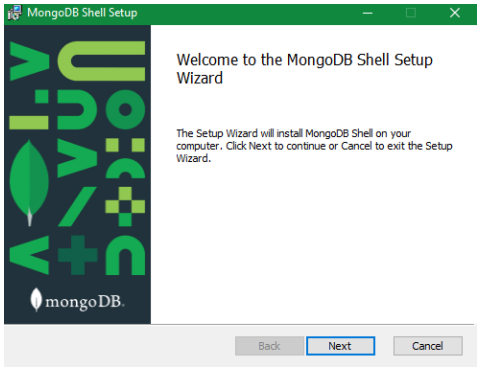
MongoDB shell

instalacja MongDB shell

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







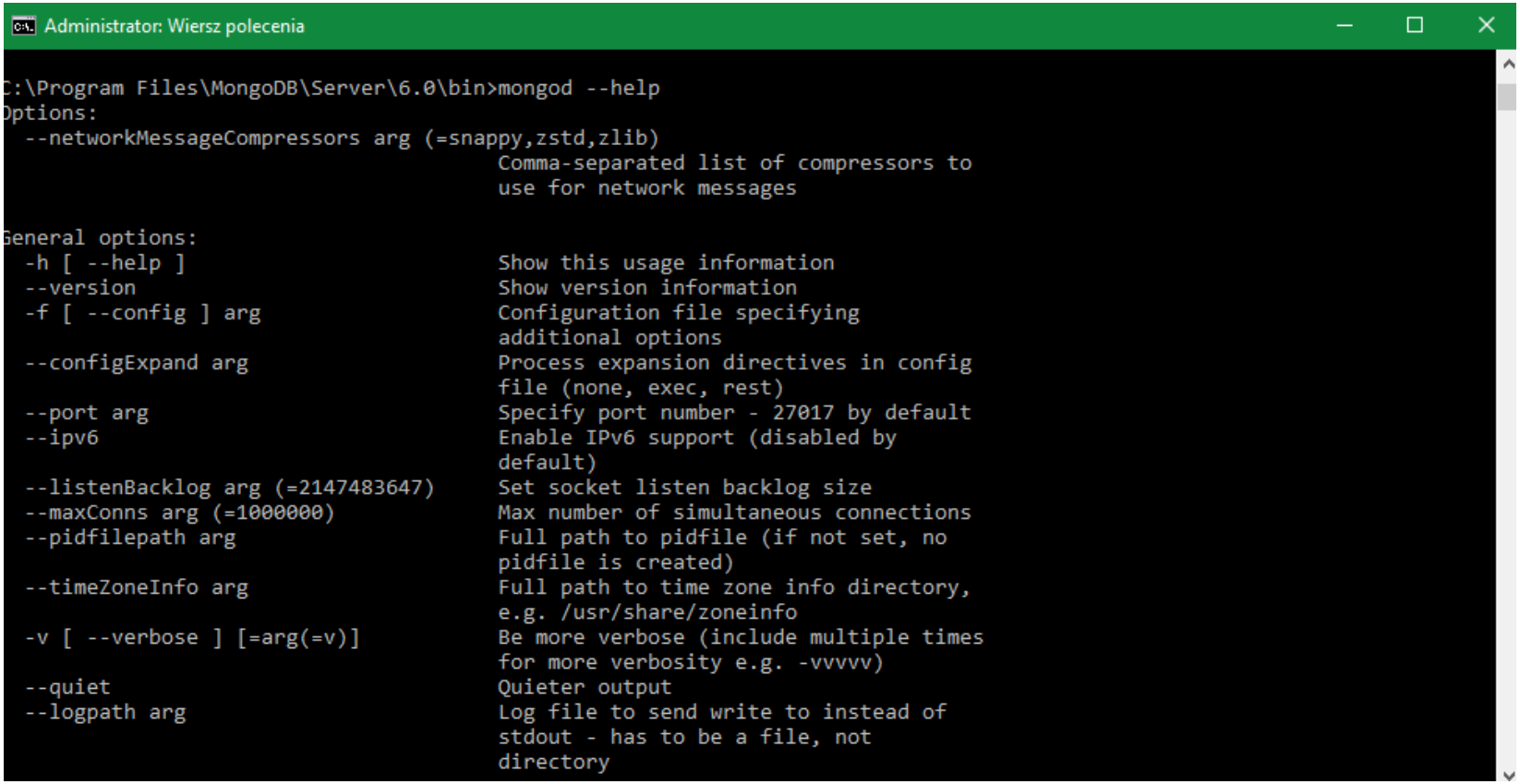
https://www.mongodb.com/docs/mongodb-shell/install/

defaultowa ścieżka instalacji shella

C:\Users\<user>\AppData\Local\Programs\mongosh\

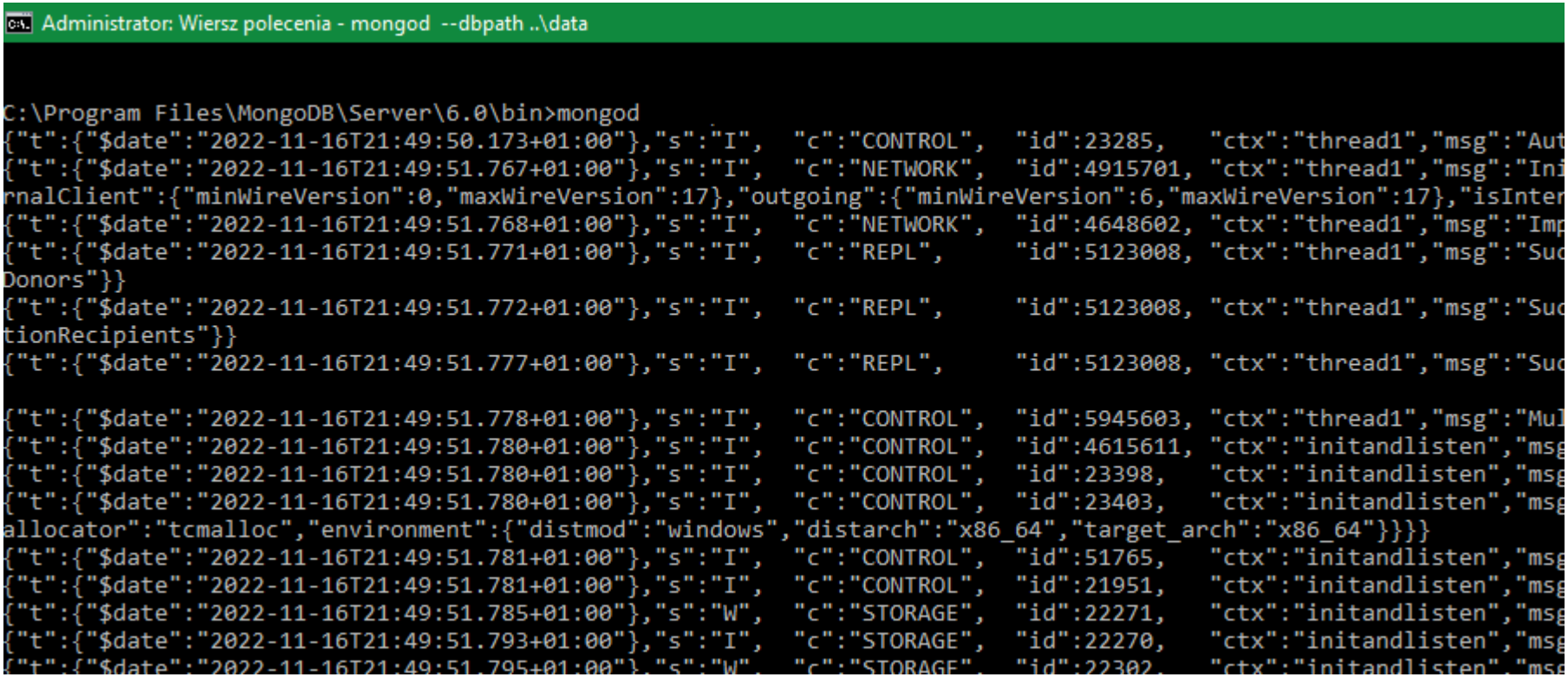
MongoDB help

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



MongoDB
uruchomienie serwera

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



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

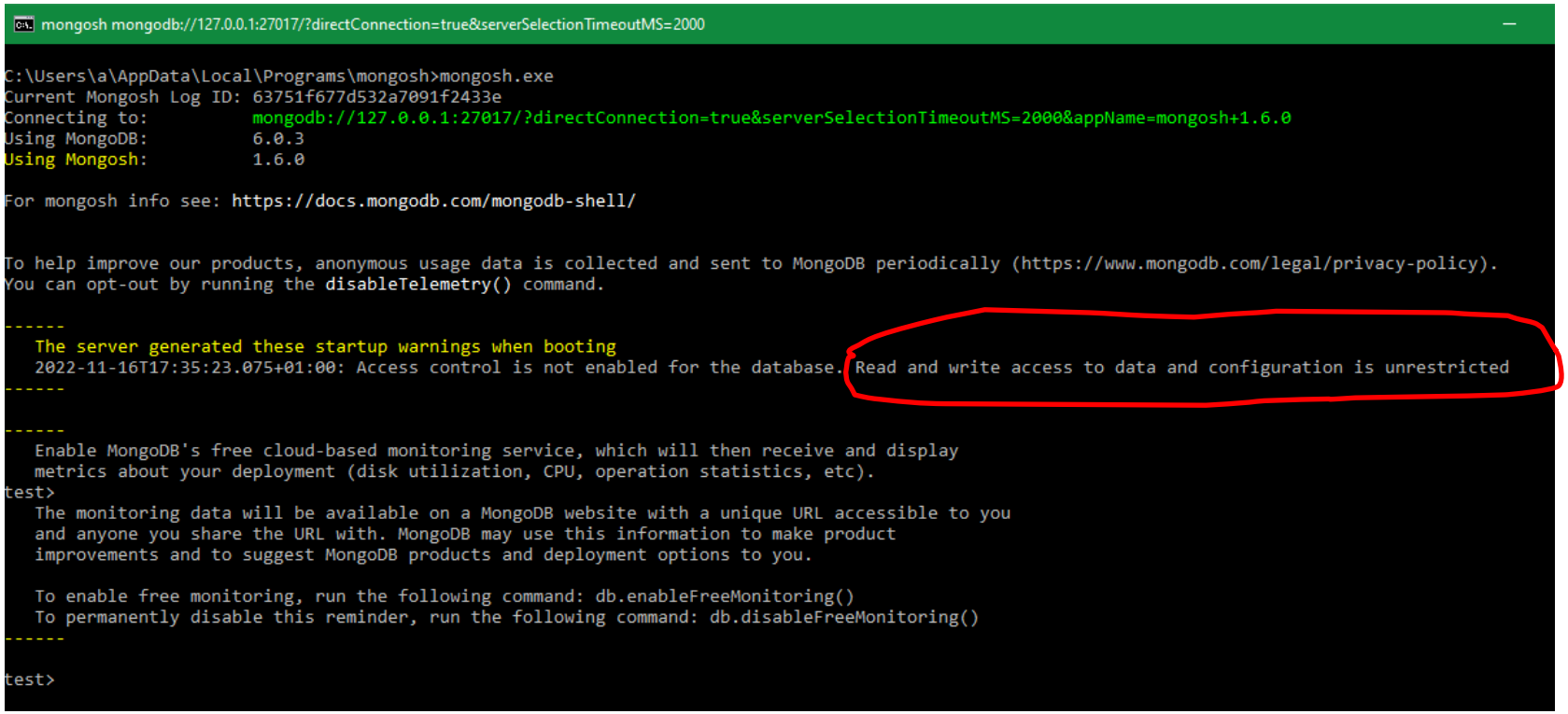
\

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

MongoDB shell

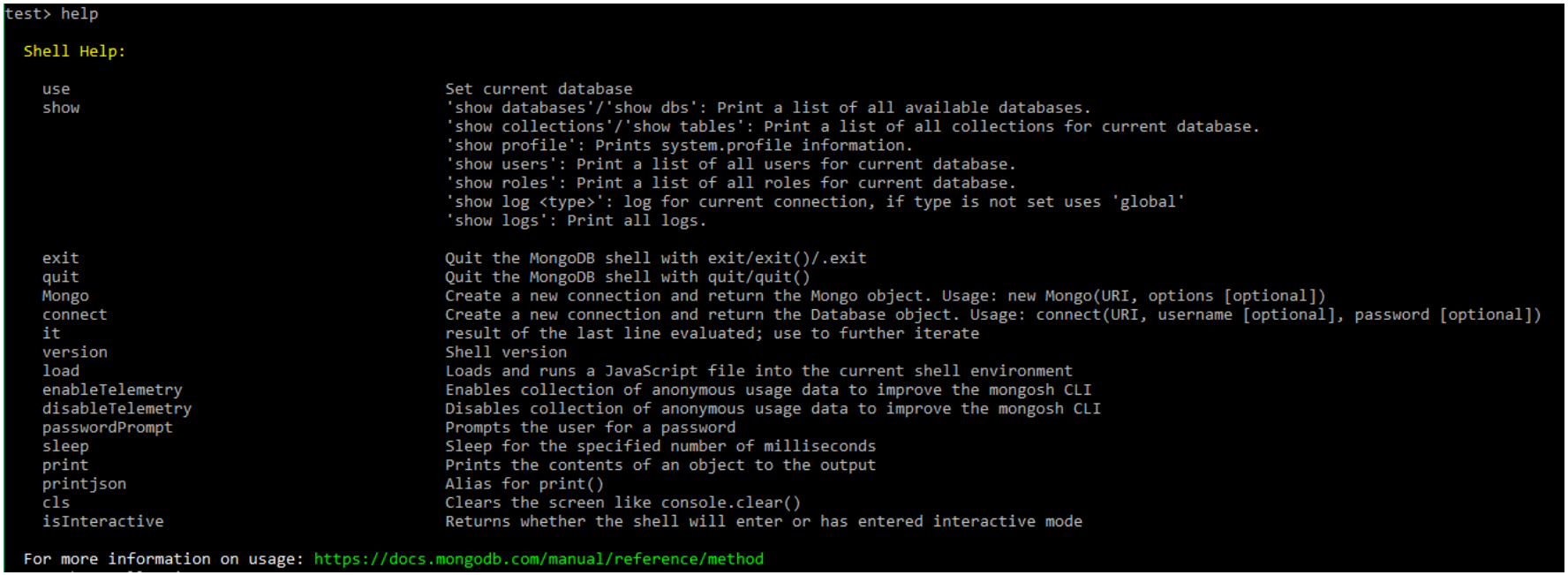
uruchomienie MongDB shella

C:\Users\<user>\AppData\Local\Programs\mongosh>mongosh.exe



MongoDB shell

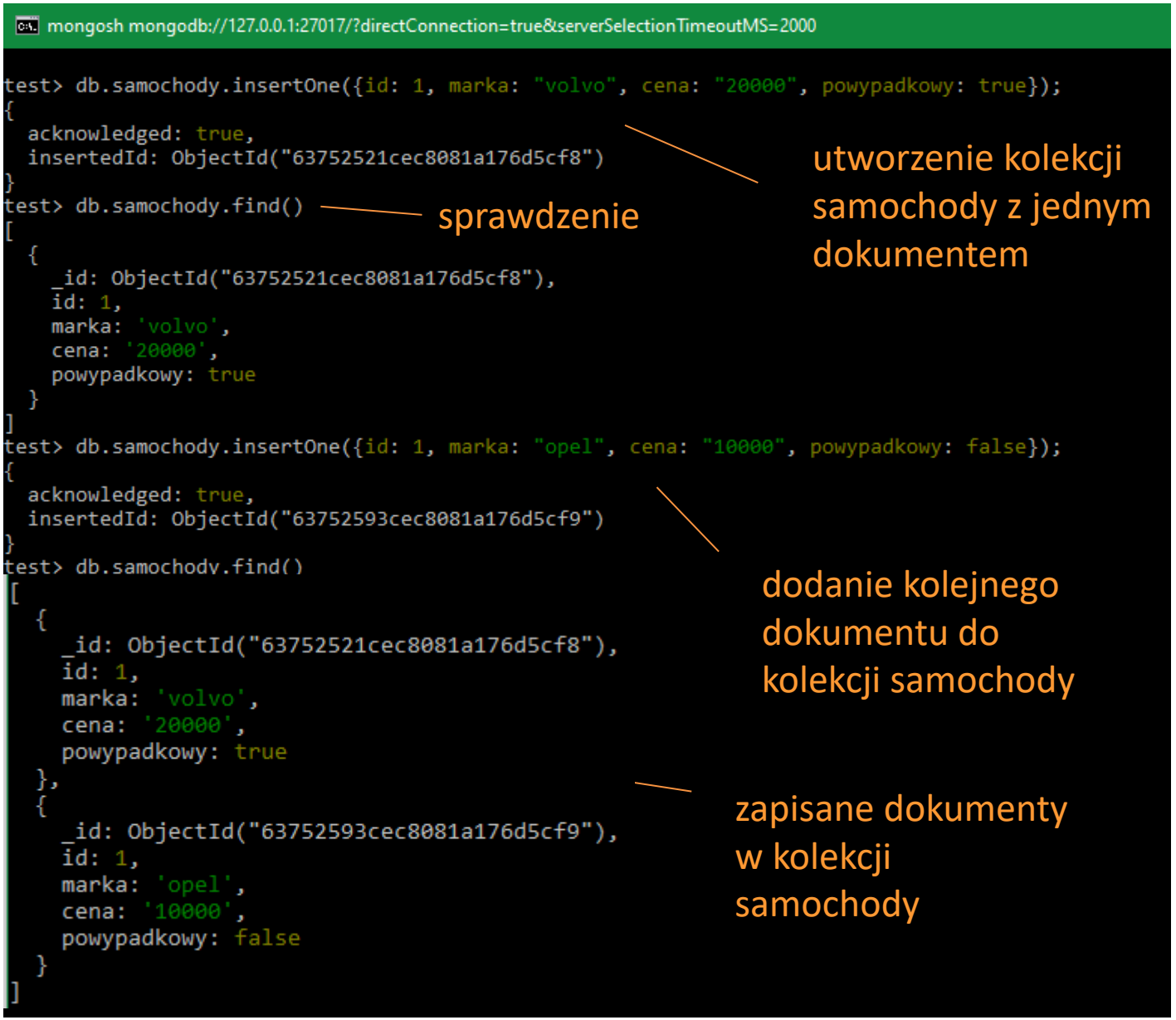
polecenie help



MongoDB shell

insertOne()

find()



sprawdzenie

utworzenie kolekcji samochody z jednym dokumentem

dodanie kolejnego dokumentu do kolekcji samochody

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

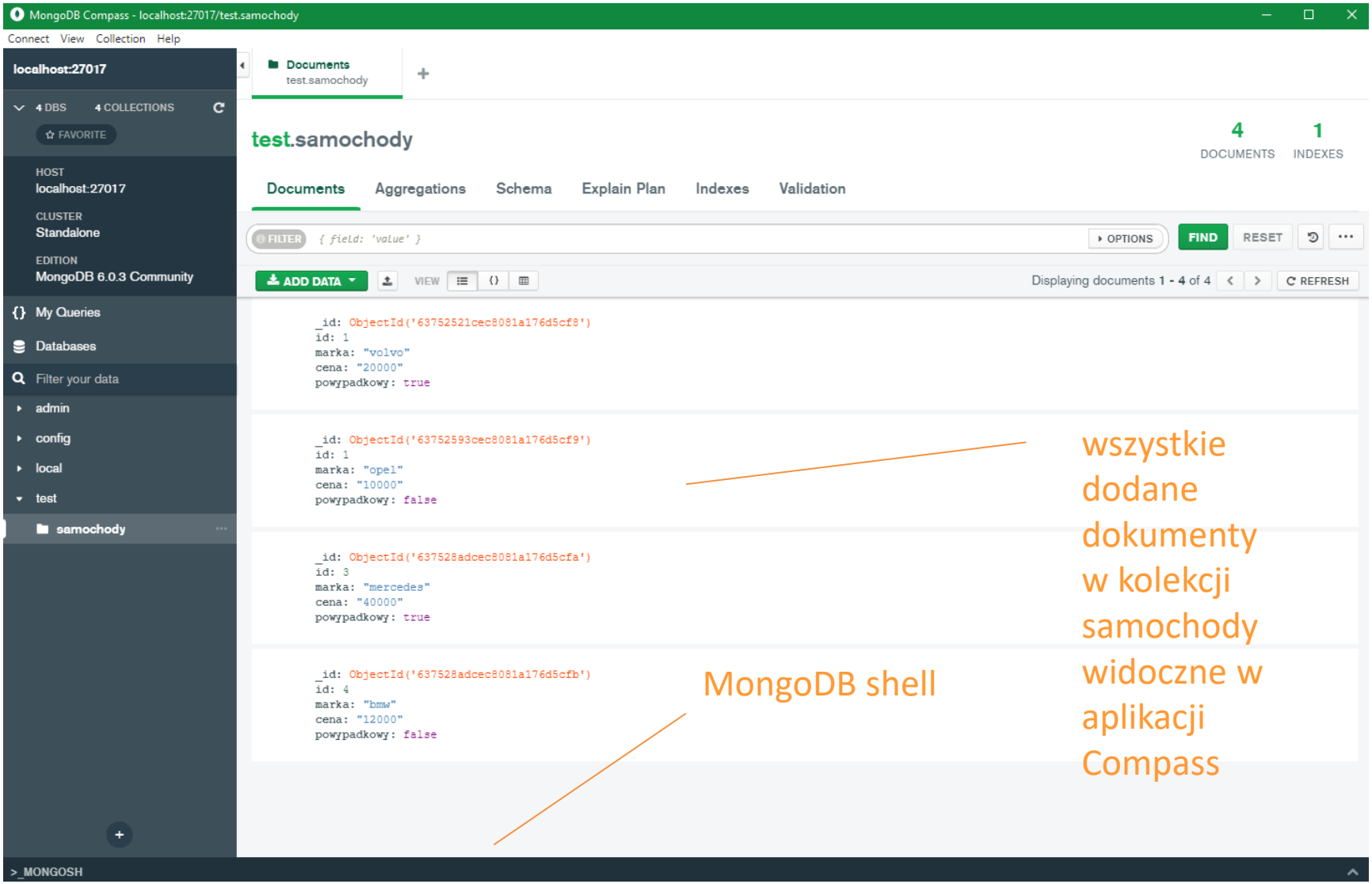
```
test> db.samochody.find({marka: {$in: ["bmw", "volvo"]}})
[
  {
    _id: ObjectId("63752521cec8081a176d5cf8"),
    id: 1,
    marka: 'volvo',
    cena: '20000',
    powypadkowy: true
  },
  {
    _id: ObjectId("637528adcec8081a176d5cfb"),
    id: 4,
    marka: 'bmw',
    cena: '12000',
    powypadkowy: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



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.

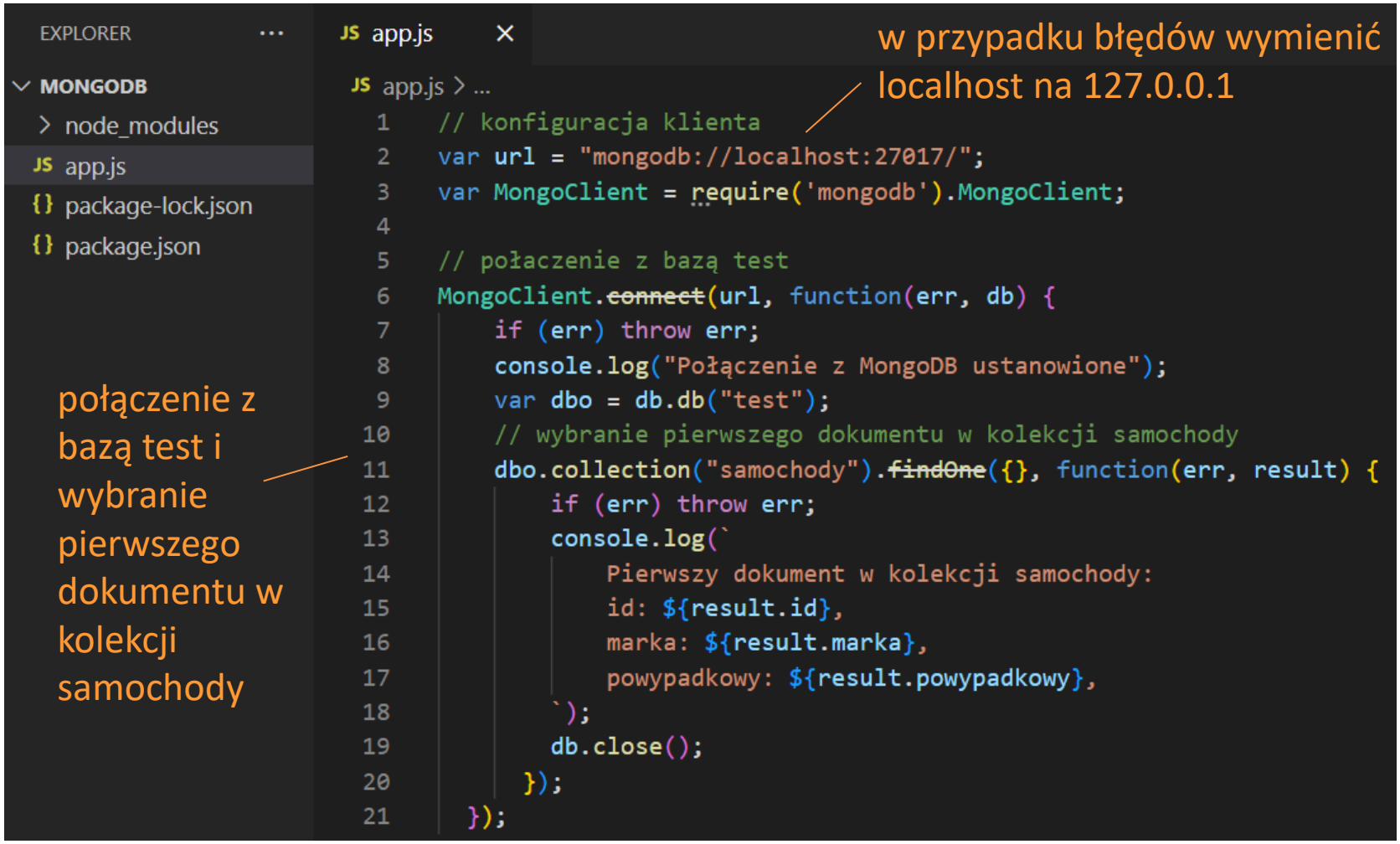
To download and install the official MongoDB driver, open the Command Terminal and execute the following:

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

Node.js MongoDB

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

instalacja sterownika mongodb



w przypadku błędów wymienić localhost na 127.0.0.1

połączenie z bazą test i wybranie pierwszego dokumentu w kolekcji samochody

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

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