Node.js

tutorial version 1.11

# 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)

**Node.js**

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

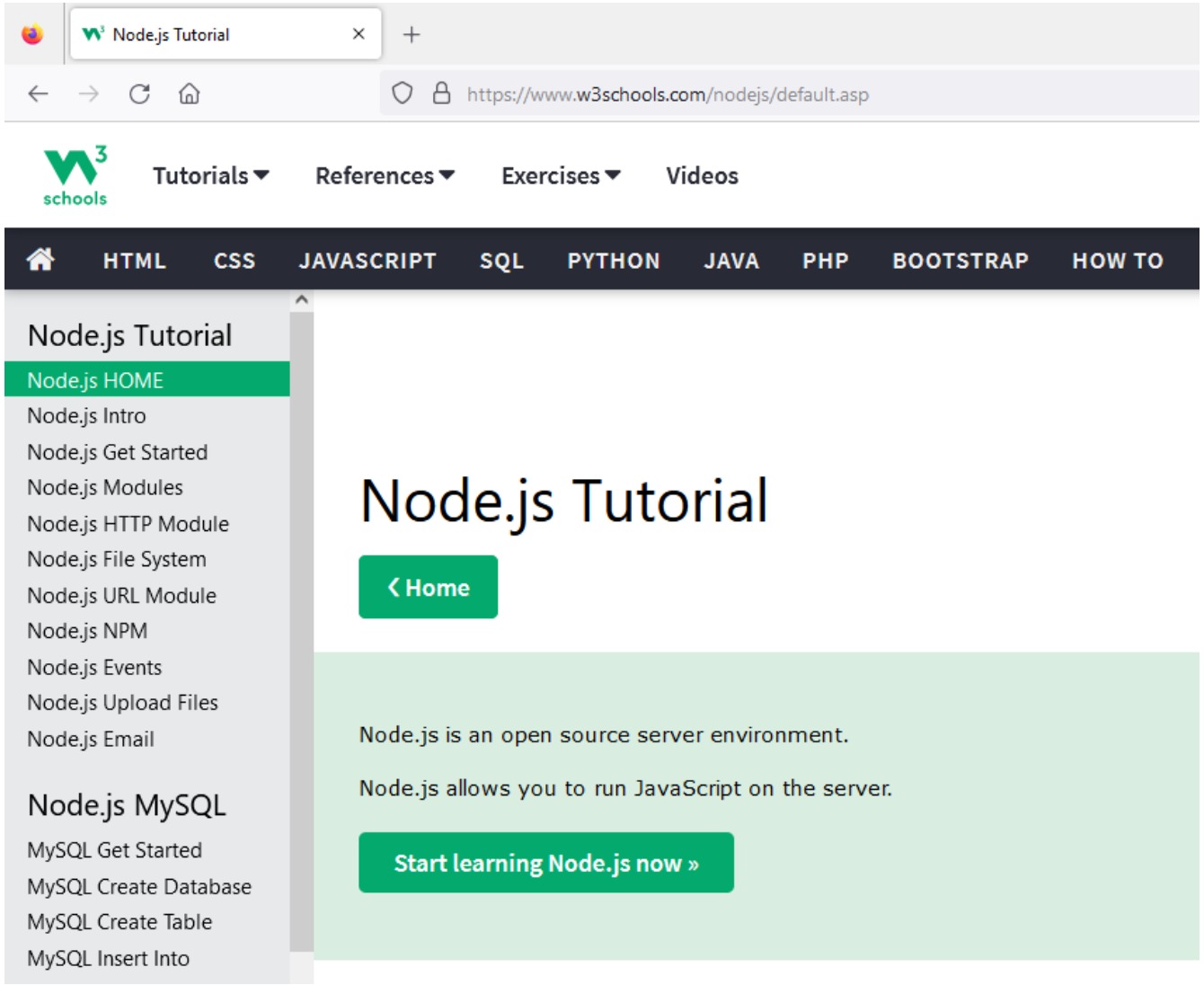
interpretacja kodu JS

środowisko uruchomieniowe kodu JS

kod JS

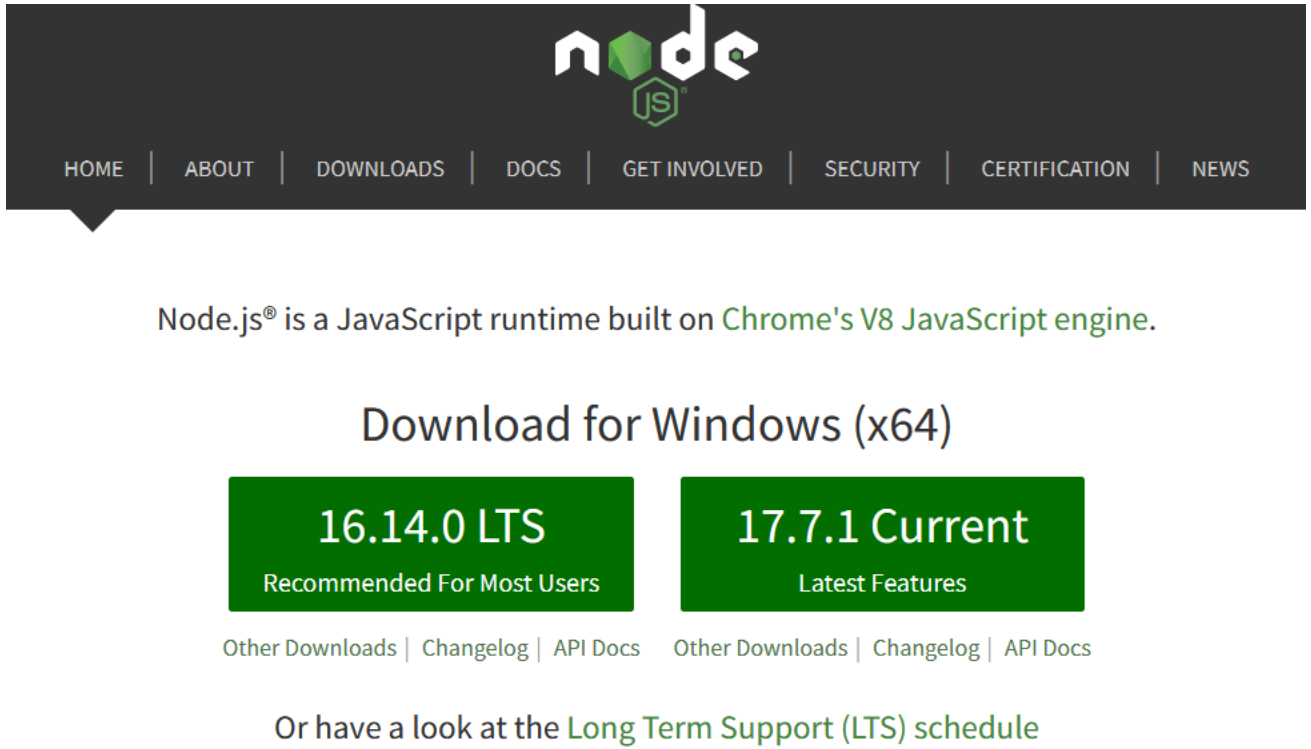
interpretacja kodu JS

# Node.js



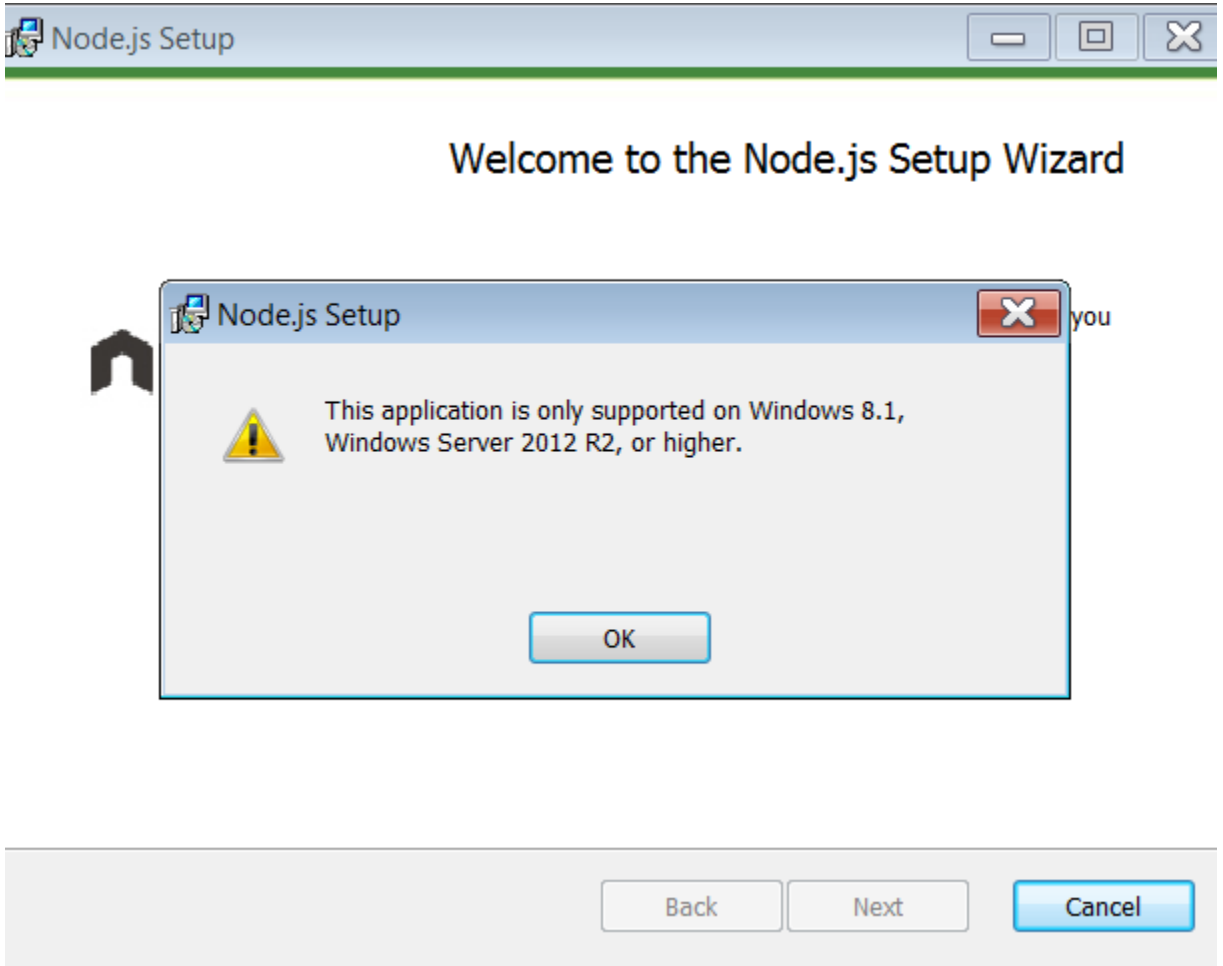

https://www.w3schools.com/nodejs/default.asp

# Node.js




https://nodejs.org/en/

# Node.js

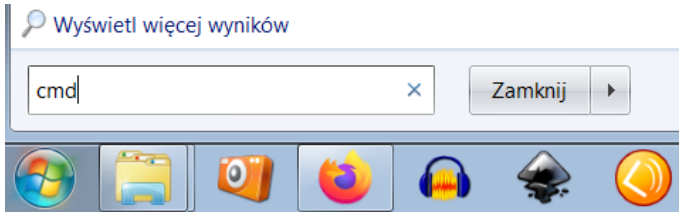



Zainstaluj Node.js na Win7 (wersja 13.14.0)

# 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

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         BigUint64Array        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               WeakSet               WebAssembly
_                     _error                assert                async_hooks           atob
btoa                  buffer                child_process         clearImmediate        clearInterval
clearTimeout          cluster               console               constants             crypto
decodeURI             decodeURIComponent    dgram                 diagnostics_channel   dns
domain                encodeURI             encodeURIComponent    escape                eval
events                fs                    global                globalThis            http
http2                 https                 inspector             isFinite              isNaN
module                net                   os                    parseFloat            parseInt
path                  perf_hooks            performance           process               punycode
querystring           queueMicrotask        readline              repl                  require
setImmediate          setInterval           setTimeout            stream                string_decoder
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 noda

# 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

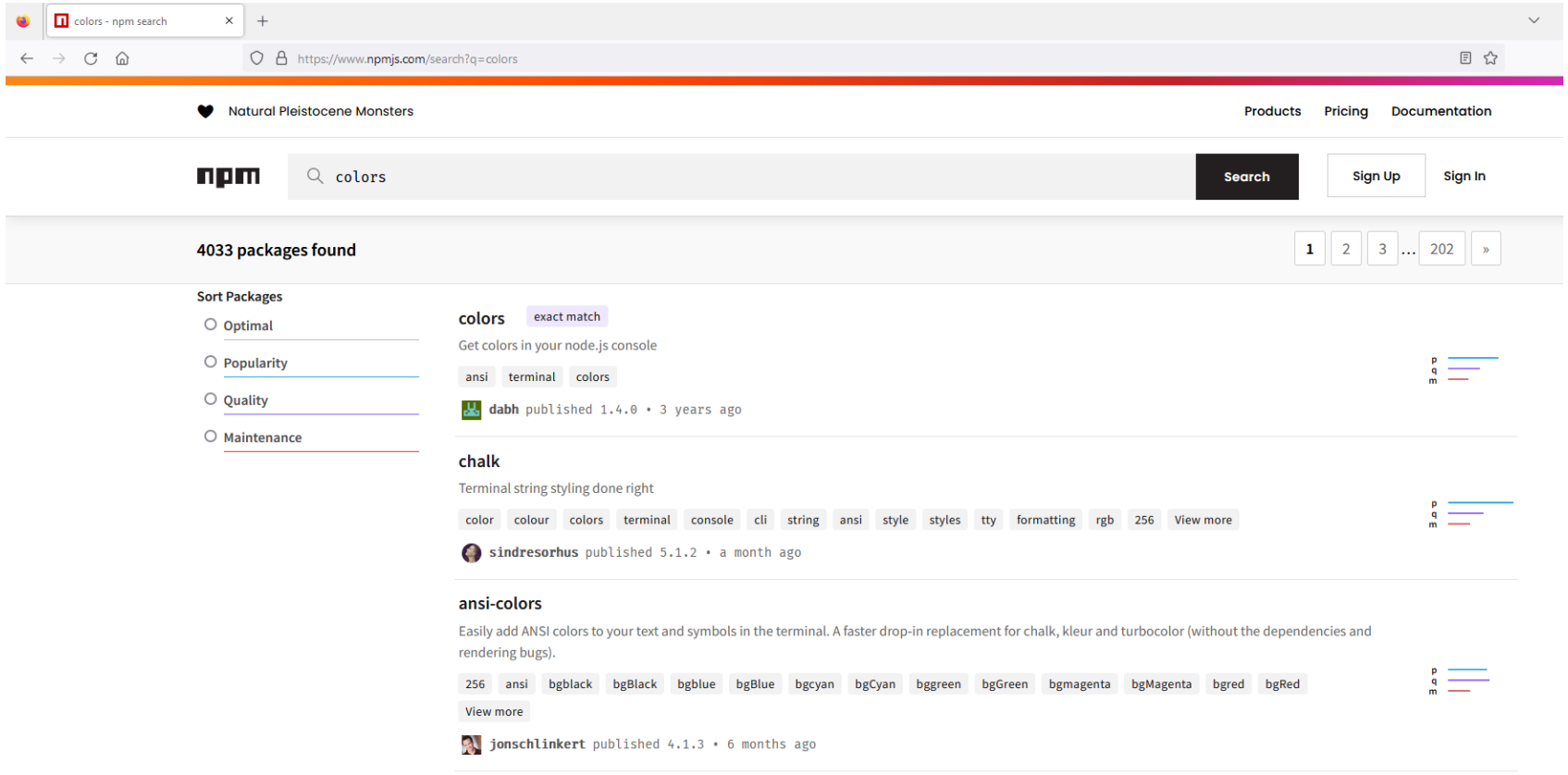
# Node.js

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 udostępniającego informacje o systemie

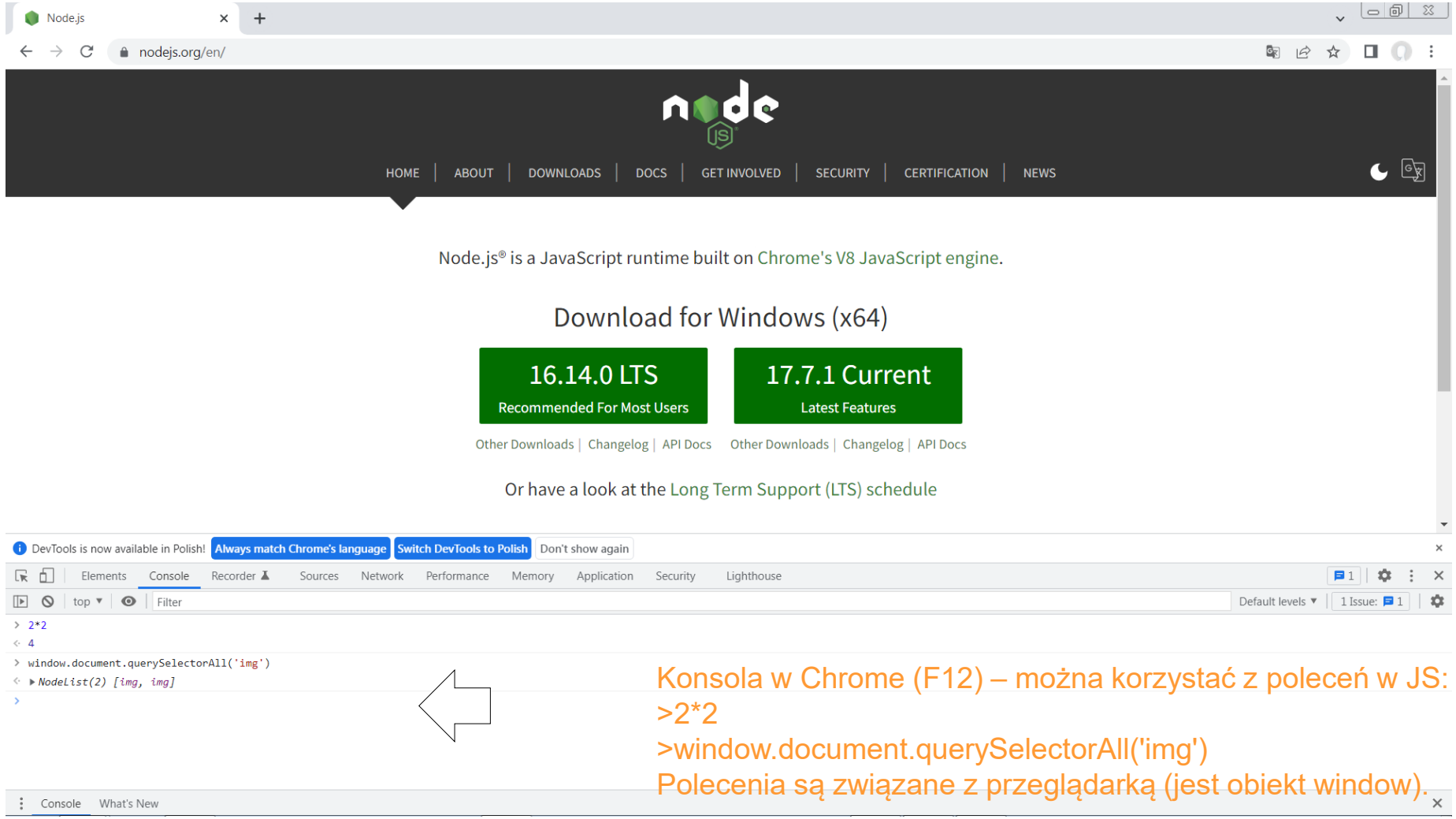
# Node.js




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

Run Terminal Help hello.js - Visual Studio Code

JS hello.js

D: > hello > JS hello.js

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

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>
```

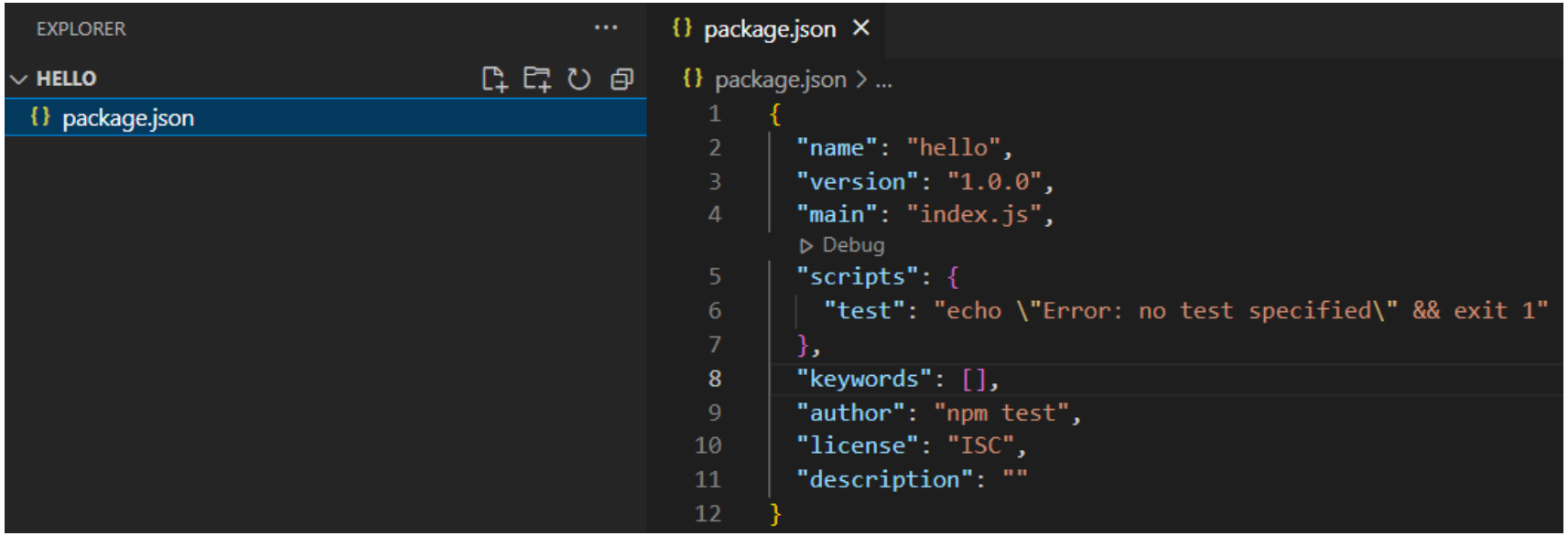
Aby uruchomić skrypt hello.js należy w konsoli przejść do katalogu z plikiem i wydać polecenie:
>node hello.js

# Node.js
# nowy projekt

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

```
npm -y init
```

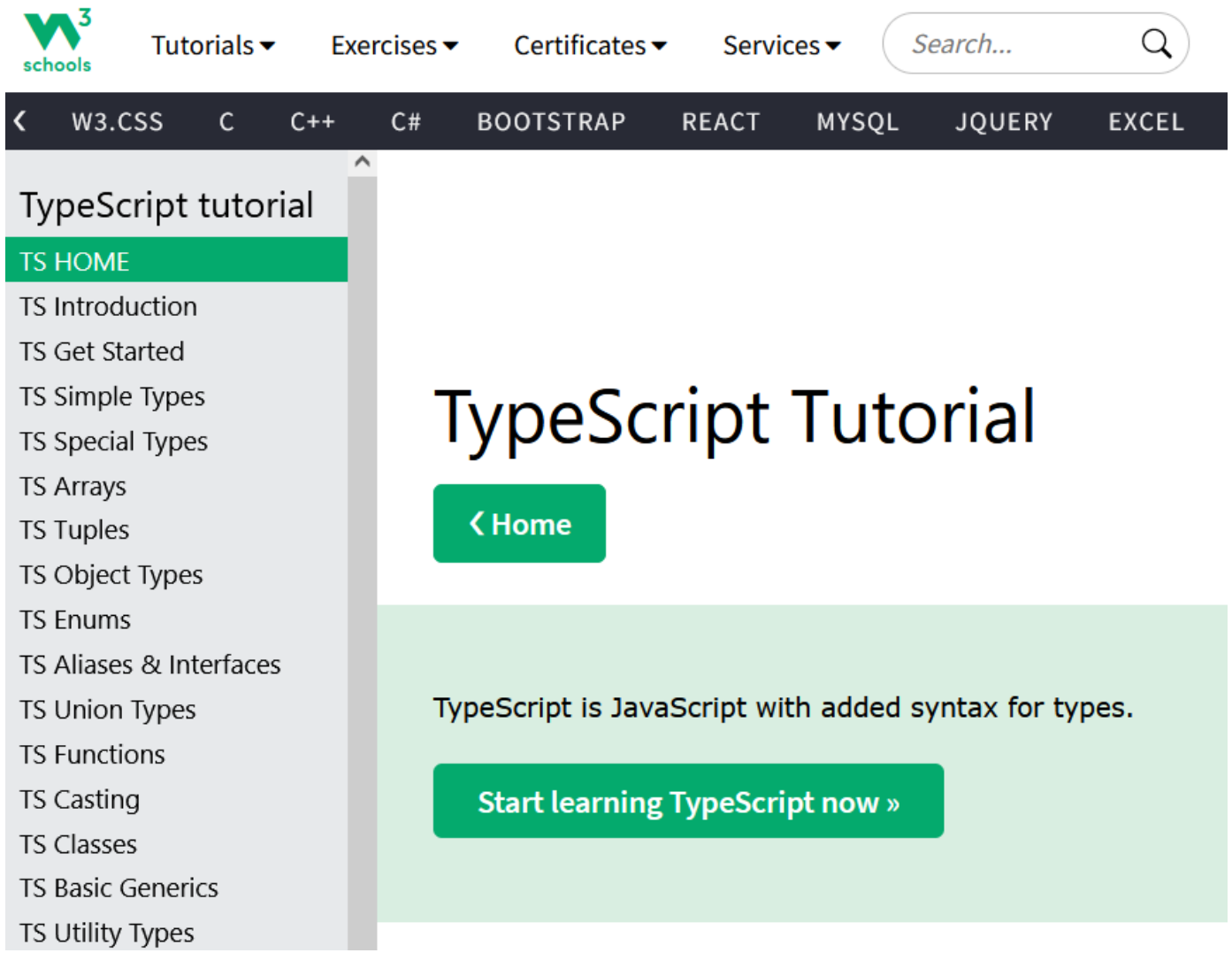
wygenerowany zostaje plik package.json z konfiguracją projektu



# TypeScript

https://www.typescriptlang.org/

# TypeScript



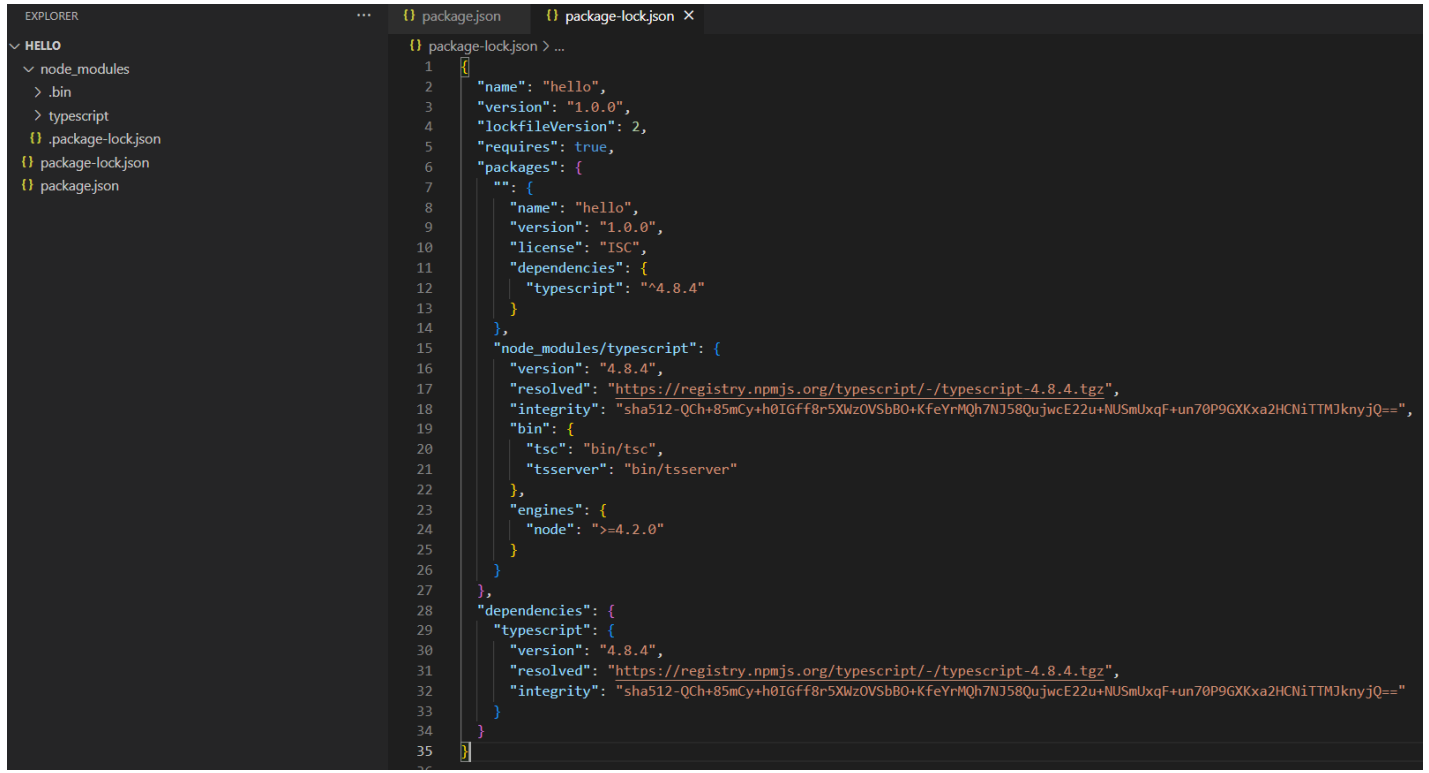

https://www.w3schools.com/typescript/

# 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



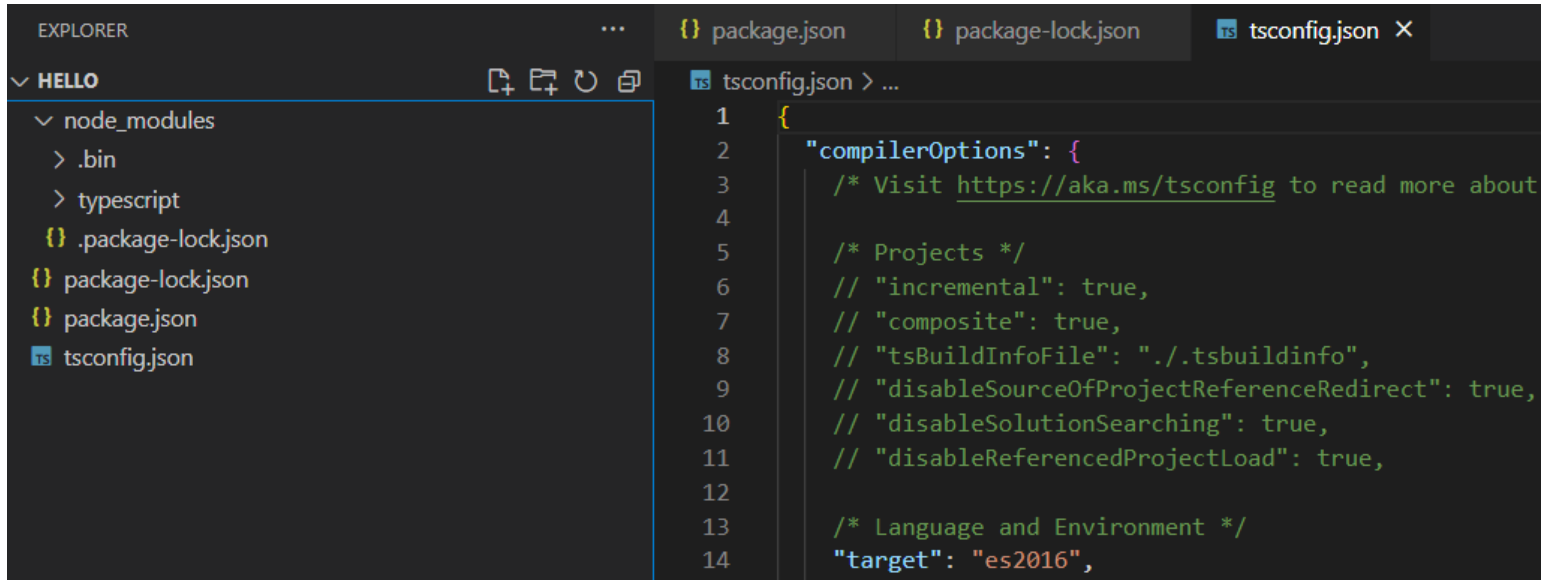
# Node.js
# inicjalizacja TypeScript

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

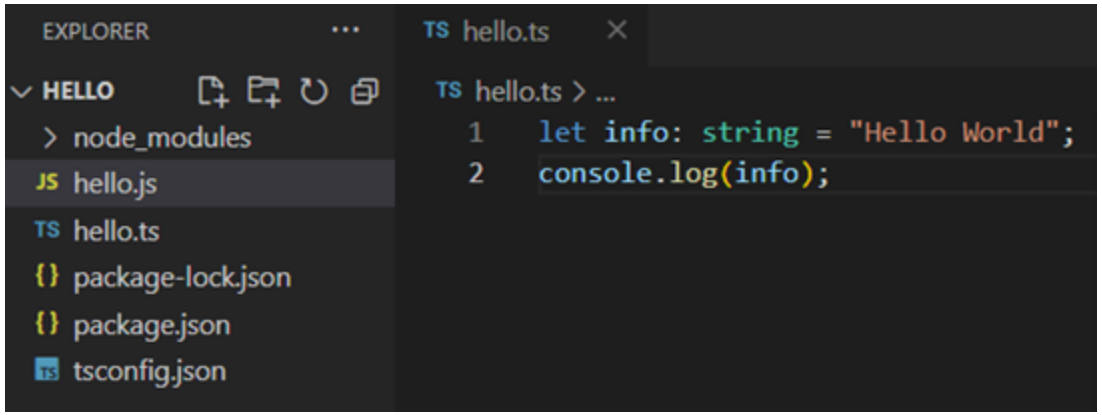
```
hello> npx tsc -init
```

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

TS hello.ts > ...

```typescript
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";    // bład - 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

hello.js

JS hello.js > ...

```javascript
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";    // bład - 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);
```

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

EXPLORER

HELLO
- node_modules
- asynchronicznosc.js
- asynchronicznosc.ts
- hello.js
- hello.ts
- package-lock.json
- package.json
- tsconfig.json

asynchronicznosc.ts > ...

```ts
let one = function(){
    setTimeout(()=>console.log("one"), 2000);
}
let two = function(){
    console.log("two");
}
let three = function(){
    console.log("three");
}

one();
two();
three();
```

asynchronicznosc.js > ...

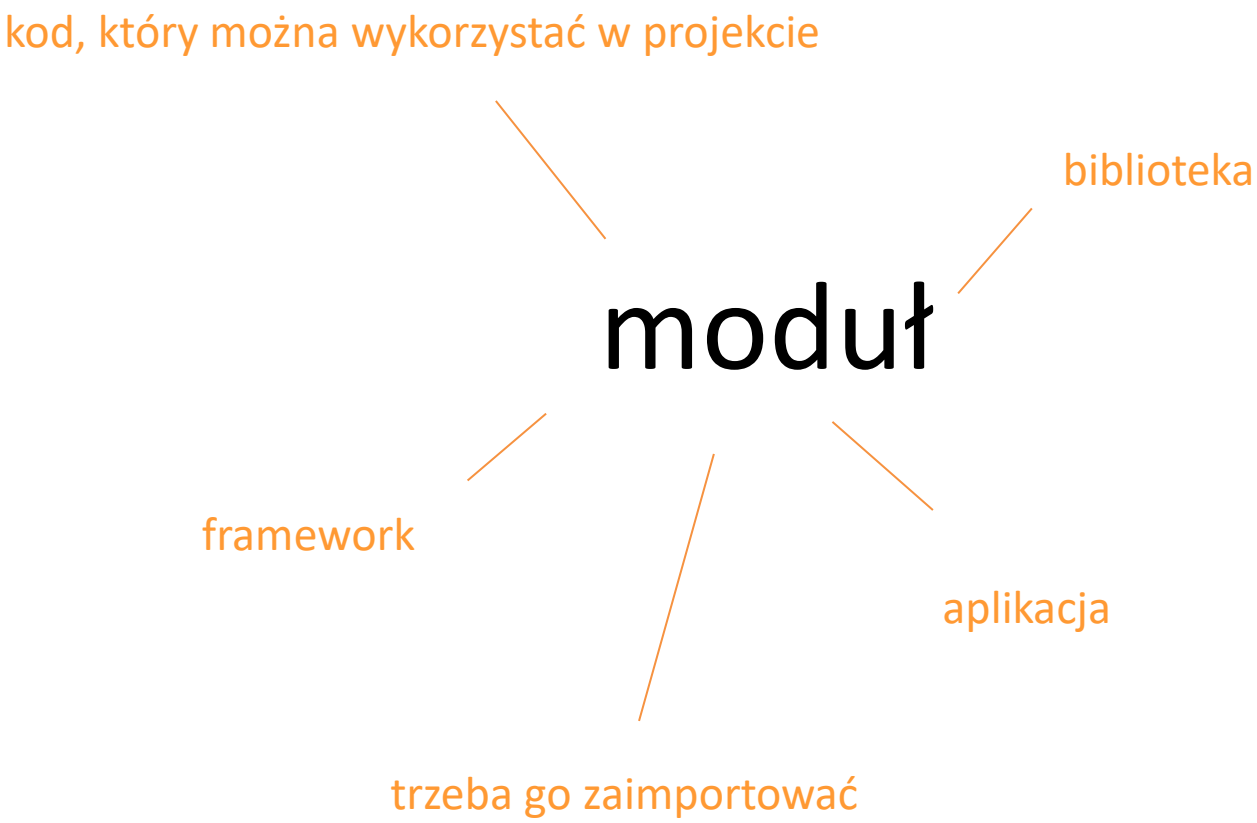
```js
var one = function () {
    setTimeout(function () { return console.log("one"); }, 2000);
};
var two = function () {
    console.log("two");
};
var three = function () {
    console.log("three");
};
one();
two();
three();
```

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

EXPLORER

MODULOS

app.js

app.js

app.js > ...

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

console.log(`
            version: ${os.version()}
            type: ${os.type()}
            uptime: ${os.uptime()}
            platform: ${os.platform()}
            release: ${os.release()}
            hostname: ${os.hostname()}
            totalmame: ${os.totalmem()}
`)
```

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
TS modulOs.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 > ...

```javascript
// import modułu url
const url = require('url');

// adres dp parsowania
const address = "http://pracownia.palacmlodziezy.lodz.pl/a/b/c/index.html"

const addr = url.parse(address, true)

console.log(`
            host: ${addr.host}
            href: ${addr.href}
            hostname: ${addr.hostname}
            path: ${addr.path}
            protocol: ${addr.protocol}
`)
```

```
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

File Edit Selection View Go Run Terminal Help app.js - modulFsReadDir - Vis

EXPLORER

MODULFSREADDIR

app.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
```

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

# moduł fs
# readfile

dostęp do plików

zawartość pliku

File Edit Selection View Go Run Terminal Help app.js - modulFsReadFile - Visual St

EXPLORER

MODULFSREAD...

- app.js
- plik.txt

app.js

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

// czytamy zawartość pliku plik.txt
fs.readFile('plik.txt', 'utf8', function(err, data){
    if (err) {
        return console.log("błąd");
    }
    // wypisujemy w logu zawartość pliku
    console.log(data);
});
```

obsługa błędów

plik.txt

```
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

EXPLORER

MODULFSWRITEFILE

- app.js
- plik.txt

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)

EXPLORER

MODULFSAPPENDFILE

- app.js
- plik.txt

plik.txt

plik.txt

```
1   Druga tajna informacja
```

app.js

app.js > ...

```js
// import modułu fs
const fs = require('fs');
// dane do zapisu
let data = "\nTrzecia tajna informacja";

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

plik.txt

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

MODULPATH

app.js

app.js > ...

```js
// import modułu path
const path = require('path');
// ścieżka do pliku
const filePath = '/a/b/main.js';
// wyciąga nazwa pliku z podanej ścieżki
const filename = path.basename(filePath);
// wyciąga nazwę katalogu z podanej ścieżki
const dirname = path.dirname(filePath);
// dokłada bieżący katalog do ścieżki
const join = path.join(__dirname, filePath);
// sprawdza czy podana ścieżka jest względna
// (Path From Content Root) (zwraca false)
// czy bezwzględna (Absolute Path) (zwraca true)
const isAbsolute = path.isAbsolute(join);

console.log(`
            filePath: ${filePath}
            filename: ${filename}
            dirname:  ${dirname}
            join:  ${join}
            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
```

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)

EXPLORER

MODULWLASNY
- polaFigur
  - index.js
- app.js

index.js

polaFigur > index.js > ...

```js
const wzory = [
{ figura: "kwadrat" , wzor: "p=a^2" },
{ figura: "prostokąt" , wzor: "p=a*b" },
{ figura: "trójkąt" , wzor: "p=a*h/2" },
];
// export metod modułu polaFigur
module.exports = {
    podajWzory(){
        console.log("\nWzory na pola figur:\n");
        for (let i = 0; i < wzory.length; i++) {
            const item = wzory[i];
            console.log(item.figura + ": " + item.wzor);
        }
        console.log("\n");
    },
    obliczPoleKwadratu(a){
        console.log("pole kadratu o boku " + a + " wynosi " + a*a);
    }
}
```

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

MODULWLASNY
- polaFigur
  - index.js
- app.js

app.js

```
// import modułu polaFigur
const wzory = require('./polaFigur');

// wywołanie metod modułu polaFigur
wzory.podajWzory();
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 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

# moduł http

prosty serwer http

serwer http

File Edit Selection View Go Run Terminal Help app.js - modulHttp - Visual Studio Cod

EXPLORER

MODULHTTP

app.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, resonse) {
6       resonse.writeHead(200, {'Content-Type': 'text/html'});
7       // wyświetlana strona
8       resonse.write(`
9                   <h1>Node.js</h1>
10                  <h2 style="color:red">Hello World!</h2>
11                  `);
12      resonse.end();
13  }).listen(port);
```

uruchomienie serwera www

```
node app.js
```

localhost:8080/

localhost:8080

# 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

Własny serwer www

localhost:8080

nawet działa

wynik działania skryptu w pliku main.js

**Własny serwer www**

styl umieszczony w odrębnym pliku main.css

Developer Tools F12

Inspektor | Konsola | Debuger | Sieć | Edytor stylów | Wydajność | Pamięć | Dane | Dostępność | Wątki | Adblock Plus

Filtrowanie adresów

| 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...

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

app.js > answer > fs.readFile() callback

```js
// 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;
```

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 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

```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>
```

EXPLORER

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

main.css

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

EXPLORER

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

main.js

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

https://favicon.io/

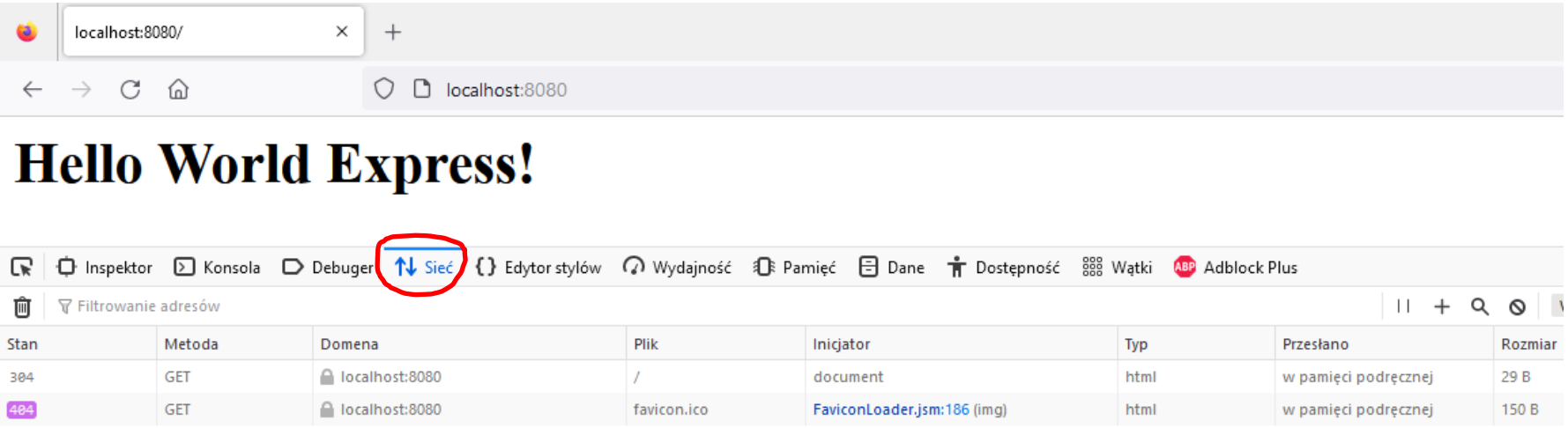
# Express
# serwer http

prosty serwer http

framework webowy

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

instalacja frameworka




serwer http w oparciu o framework Express

Developer Tools F12

# Express serwer http

prosty 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
```

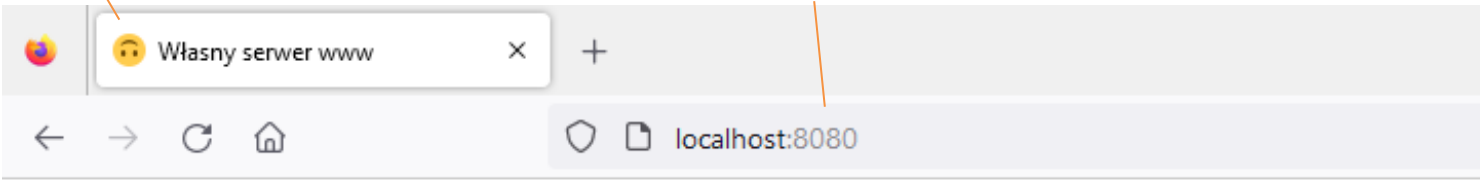
# Express serwer http

serwer http wyświetlający pliki strony

framework

favicon.ico https://favicon.io/

serwer nasłuchuje lokalnie na porcie 8080




nawet działa

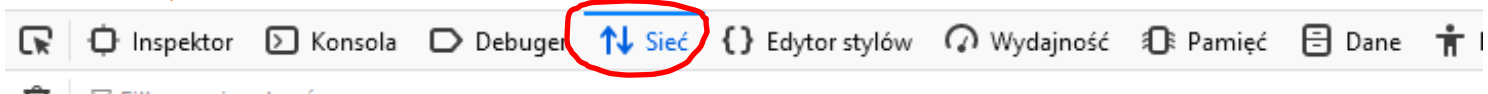
wynik działania skryptu w pliku main.js

**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

Developer Tools F12




| 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 |

# Express serwer http

EXPLORER

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

app.js

```
// import frameworka express
const express = require('express');
// import własnego modułu wyswietlającego logi
const serwerLogs = require('./serwerLogs');

// wywołanie frameworka
const app = express();
// port, na którym serwer nasłuchuje
const port = 8080;

// ścieżki do plików strony
// muszą być bezwzględne
// lub trzeba uzywać root
// w którym podajemy ścieżkę bezwzględną
// do folderu z plikami strony np.:
// res.sendFile(main, {root: ...})
const main = 'index.html';
const css = 'main.css';
const js = 'main.js';
const image = 'favicon.ico';


// obsługa żądania GET strony głównej
app.get('/', (req, res) => {
    // w root jest podana ścieżka bezwzględna do katalogu
    // z plikami strony
    res.sendFile(main,{ root: __dirname + '\\strona'});
    // wyświetlenie logów za pomocą własnego modułu
    serwerLogs.show(main,req);
});

// obsługa żądania GET pliku main.css
app.get('/main.css', (req, res) => {
    res.sendFile(css,{ root: __dirname + '\\strona'});
    serwerLogs.show(css,req);
});

// obsługa żądania GET pliku main.js
app.get('/main.js', (req, res) => {
    res.sendFile(js,{ root: __dirname + '\\strona'});
    serwerLogs.show(js,req);
});

// obsługa żądania GET pliku favicon.ico
app.get('/favicon.ico', (req, res) => {
    res.sendFile(image,{ root: __dirname + '\\strona'});
    serwerLogs.show(image,req);
});

// uruchomienie serwera nasłuchującego lokalnie na porcie 8080
app.listen(port, () => {
    console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
});
```

app.js

logi serwera

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

# Express serwer http

index.html

EXPLORER

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

index.html

strona > index.html > ...

```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 oparty na Node.js i Express</h1>
</body>
</html>
```

main.css

EXPLORER

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

main.css

strona > main.css > ...

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

main.js

EXPLORER

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

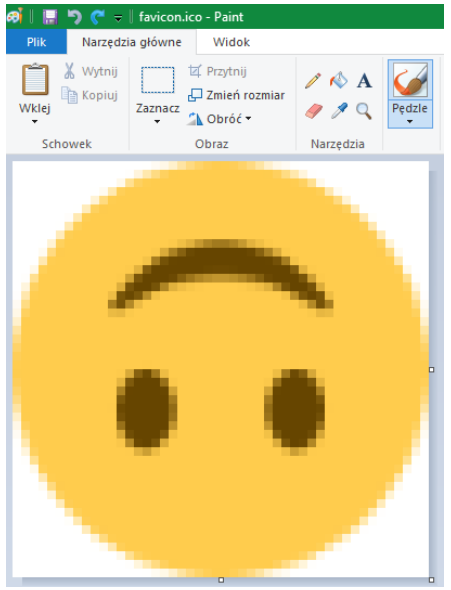
main.js

strona > main.js

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

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

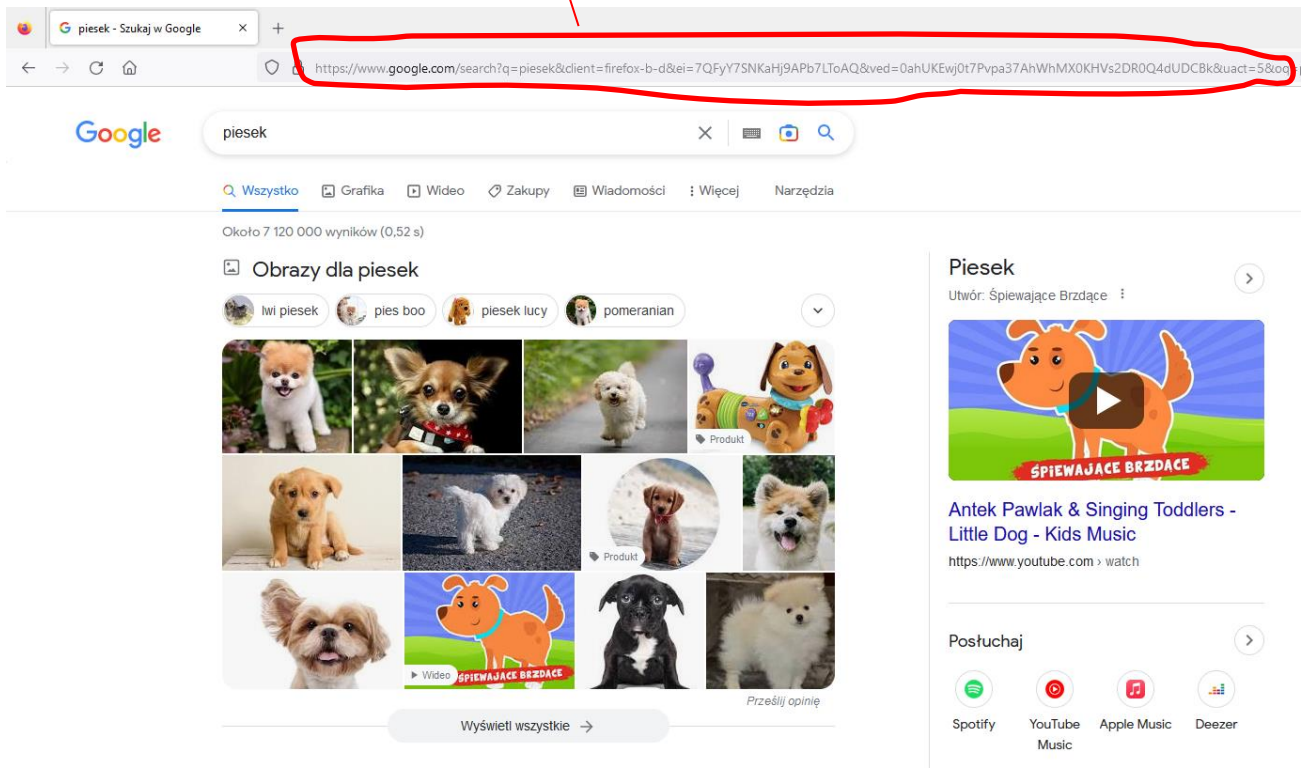
adres URL

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=Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIOCC4QgAQQsQMQgwEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAEELEDEIMBMggIABCxAxCDATILCC4QgAQQsQMQgwEyCwgAEIAEELEDEIMBMgsIABCABBCxAxCDATIFCC4QgAQ6CggAEEcQ1gQQsAM6BwgAELADEEM6DAguEMgDELADEEMYAToFCAAQgAQ6EAguELEDEIMBEMcBENEDEEM6CAguELEDEIMBOgQIABBDOgoILhDHARDRAxBDOgQILhBDOggILhCABBDUAjoKCC4QsQMQgwEQQzoHCC4Q1AIQQzoQCC4QgwEQ1AIQsQMQgAQQQzoHCAAQsQMQQzoLCC4QgAQQsQMQ1AI6DQguELEDEIMBENQCEENKBAhBGABKBAhGGAFQ_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

początek definicji parametrów

nazwa parametru

adres URL

wartość parametru

separator

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

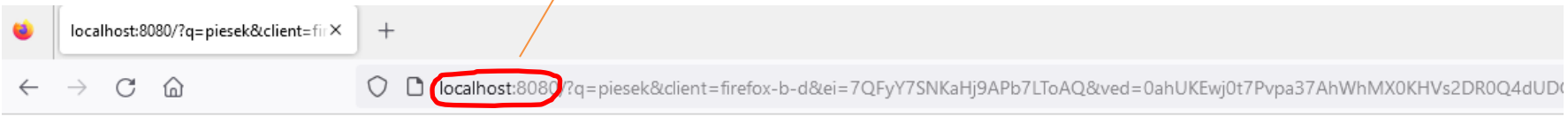
```
// obsługa żądania GET strony głównej
app.get('/', (req, res) => {

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

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

przesłane 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 (pary klucz-wartość)

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:
Cgxnd3Mtd2l6LXNlcnAQAzIKCAAQsQMQgwEQQzIOCC4QgAQQsQMQgwEQ1AIyCggAELEDEIMBEEMyCggAELEDEIMBEEMyCwgAEIAEELEDEIMB
ved: 0ahUKEwj0t7Pvpa37AhWhMX0KHVs2DR0Q4dUDCBk,
sclient: gws-wiz-serp

# Express przesyłanie informacji

EXPLORER

PRZESYŁANIEZMIENNYCH
- node_modules
- app.js
- package-lock.json
- package.json

app.js

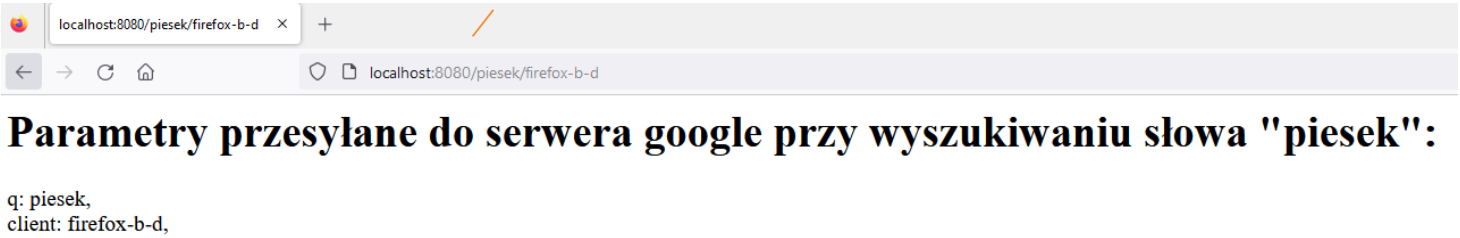
```
// import frameworka express
const express = require('express');

// wywołanie frameworka
const app = express();
const port = 8080;

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

    // pobieramy dane z adresu, który wyszukiwarka przesłała na serwer
    const { q, client, ei, wed, uact, oq, gs_lcp, ved, sclient} = req.query;
    res.send(`
            <h1>Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":</h1>
            q: ${q},<br>
            client: ${client},<br>
            ei: ${ei},<br>
            wed: ${wed},<br>
            uact: ${uact},<br>
            oq: ${oq},<br>
            gs_lcp: ${gs_lcp},<br>
            ved: ${ved},<br>
            sclient: ${sclient}
    `)
});

// uruchomienie serwera nasłuchującego lokalnie na porcie 8080
app.listen(port, () => {
    console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
});
```

# Express przesyłanie informacji

inny sposób doklejenia parametrów do adresu

wartości parametrów q oraz client

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




EXPLORER

PRZESYŁANIEPARAMETRO...
- node_modules
- app.js
- package-lock.json
- package.json

app.js > ...

```js
// import frameworka express
const express = require('express');

// wywołanie frameworka
const app = express();
const port = 8080;

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

    // pobieramy dane z adresu, który wyszukiwarka przesłała na serwer
    const q  = req.params.q;
    // alternatywnie
    const client = req.params['client'];
    res.send(`
            <h1>Parametry przesyłane do serwera google przy wyszukiwaniu słowa "piesek":</h1>
            q: ${q},<br>
            client: ${client},<br>
    `)
});

// uruchomienie serwera nasłuchującego lokalnie na porcie 8080
app.listen(port, () => {
    console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
```

# zadanie

utwórz stronę z zagadkami

formularz przesyła dane metodą get

Zagadki

localhost:8080

# Zagadki

zagadka

stolica Czech ?

Praga sprawdź

od początku

Zagadki

localhost:8080/?answer=Praga

# Zagadki

3! = ?

wpisz odpowiedź sprawdź

dobrze

od początku

weryfikacja odpowiedzi – gdy użytkownik zgadnie wyświetla się następna zagadka

Zagadki

localhost:8080/?answer=

# Zagadki

Pytania się skończyły

wpisz odpowiedź sprawdź

od początku

komunikat końcowy gdy pytania się wyczerpią

Zagadki

localhost:8080/?jeszczeRaz=1

# Zagadki

stolica Czech ?

wpisz odpowiedź sprawdź

od początku

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

# odpowiedź do zadania

EXPLORER

FORMZAGADKA
- node_modules
- app.js
- package-lock.json
- package.json

app.js 1/3

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

# 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

dane przesyłamy metodą GET, atrybut „answer” wraz z wartością zostanie doklejony do adresu

weryfikacja odpowiedzi na stronie głównej

wysłanie danych do przeglądarki

# cookie

formularz do zapisywania pliku cookie

Cookie

localhost:8080

## Wpisz zawartośc pliku cookie

node.js zapisz

Cookie

localhost:8080/?myCookie=node.js

## Wpisz zawartośc pliku cookie

wpisz plik cookie zapisz

Twoja informacja została zapisana w pliku cookie o nazwie myCookie

pokaż plik cookie

usuń plik cookie

localhost:8080/getMyCookie

localhost:8080/getMyCookie

JSON Nieprzetworzone dane Nagłówki

Zapisz Kopiuj Zwiń wszystkie Rozwiń wszystkie Filtruj JSON

myCookie: "node.js"

Inspektor Konsola Debuger Sieć Edytor stylów Wydajność Pamięć Dane Dostępność Wątki Adblock Plus

Filtruj elementy

Lokalna pamięć

Ciasteczka

http://localhost:8080

IndexedDB

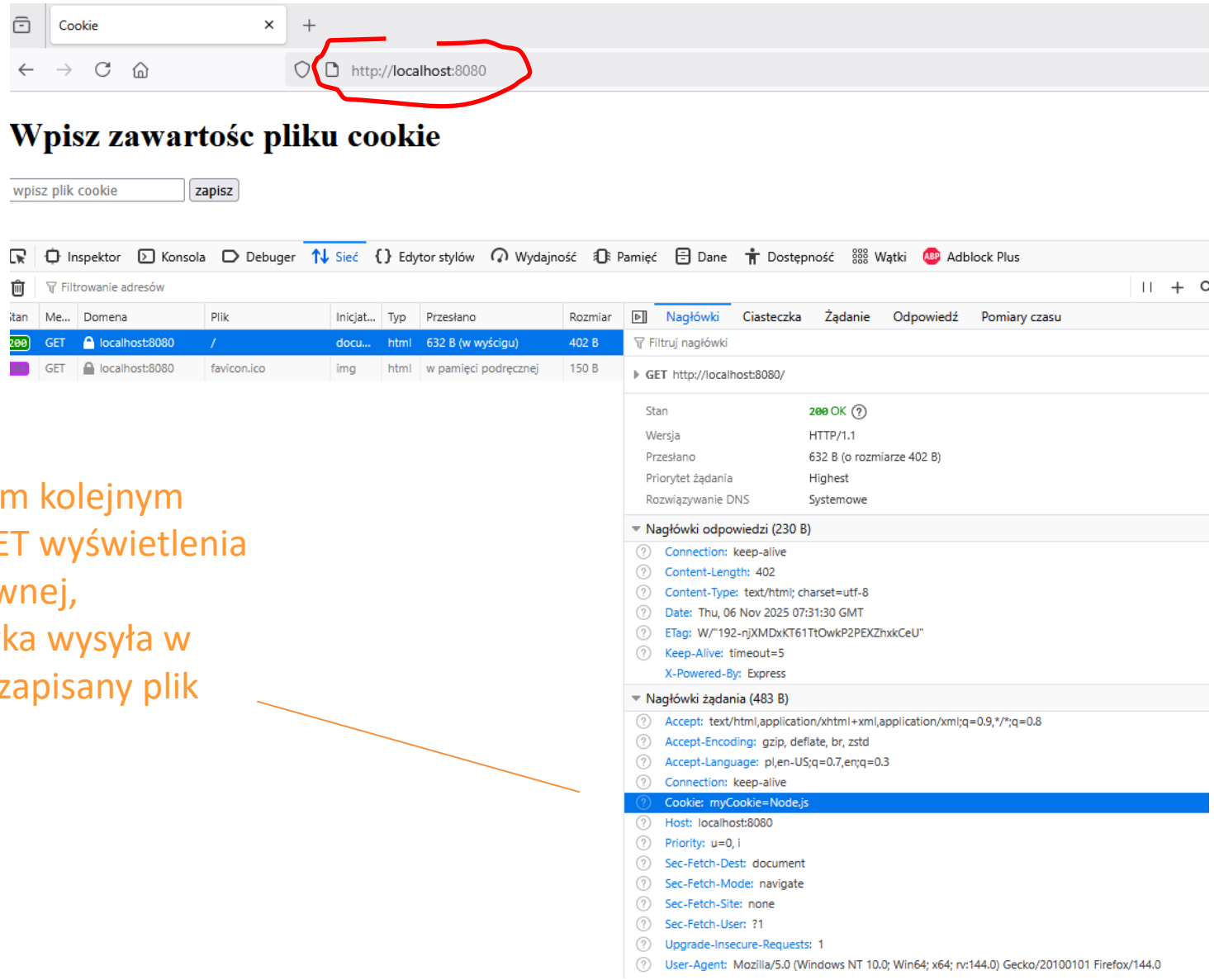
Pamięć podręczna

Pamięć sesji

| Nazwa | Wartość | Domain | Path | Wygasa / Max-Age |
|---|---|---|---|---|
| myCookie | node.js | localhost | / | Sesja |

zapisany plik cookie o nazwie myCookie z zawartością z formularza

# cookie




przy każdym kolejnym żądaniu GET wyświetlenia strony głównej, przeglądarka wysyła w nagłówku zapisany plik cookie

# cookie

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

instalacja modułu cookie-parser

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

cookie-parser to middleware (pośrednik) dla frameworka Express.js, który automatycznie odczytuje i rozkodowuje pliki cookie z żądań HTTP.
Dzięki niemu można w prosty sposób uzyskać dostęp do ciasteczek przez req.cookies zamiast ręcznie parsować nagłówek Cookie.

```
// 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 za pośrednictwem cookieParser

# cookie

EXPLORER

COOKIE
- node_modules
- app.js
- package-lock.json
- package.json

app.js > app.get('/') callback

```js
// import frameworka express
const express = require('express');
// import parsera cookie
const cookieParser = require('cookie-parser');
// wywołanie frameworka
const app = express();
// uruchomienie cookie parsera
app.use(cookieParser());
// port, na którym serwer nasłuchuje
const port = 8080;

// obsługa żądania GET strony głównej
app.get('/', (req, res) => {
    // informacja o zapisie pliku cookie
    cookieInfo = "";
    // link do pobrania cookie
    cookieLink = "";
    // link do usunięcia cookie
    cookieClear = "";

    // pobieramy dane wysłane z formularza
    // za pomocą metody get
    const myCookie = req.query.myCookie;

    // sprawdzamy czy użytkownik przesłał dane
    if (myCookie != "" && myCookie != undefined){
        // zapis plku cookie pod nazwą myCookie
        res.cookie('myCookie', myCookie);
        cookieInfo = `Twoja informacja została zapisana w pliku
                cookie o nazwie myCookie`;
        cookieLink = `<a href="http://localhost:${port}/getMyCookie">pokaż plik cookie</a>`
        cookieClear = `<a href="http://localhost:${port}/clearCookie">usuń plik cookie</a>`
    }

```

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

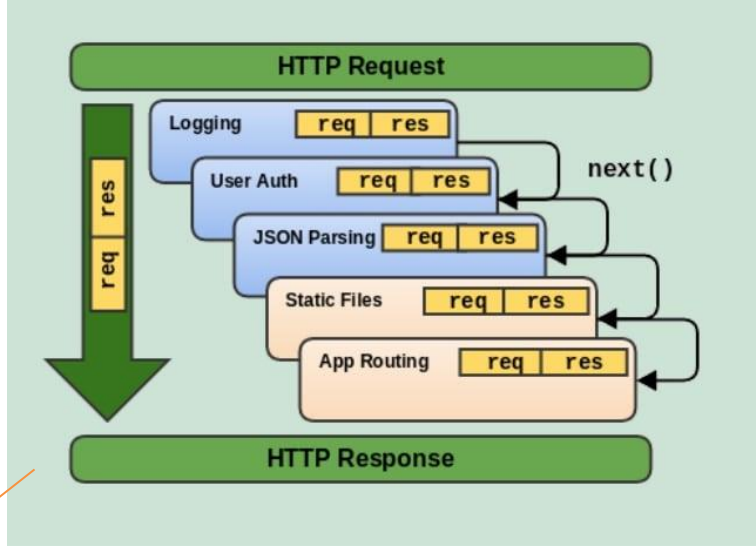
# middleware

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

MIDDLEWARE
- node_modules
- app.js
- package-lock.json
- package.json

app.js > ...

```javascript
// 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

EXPLORER

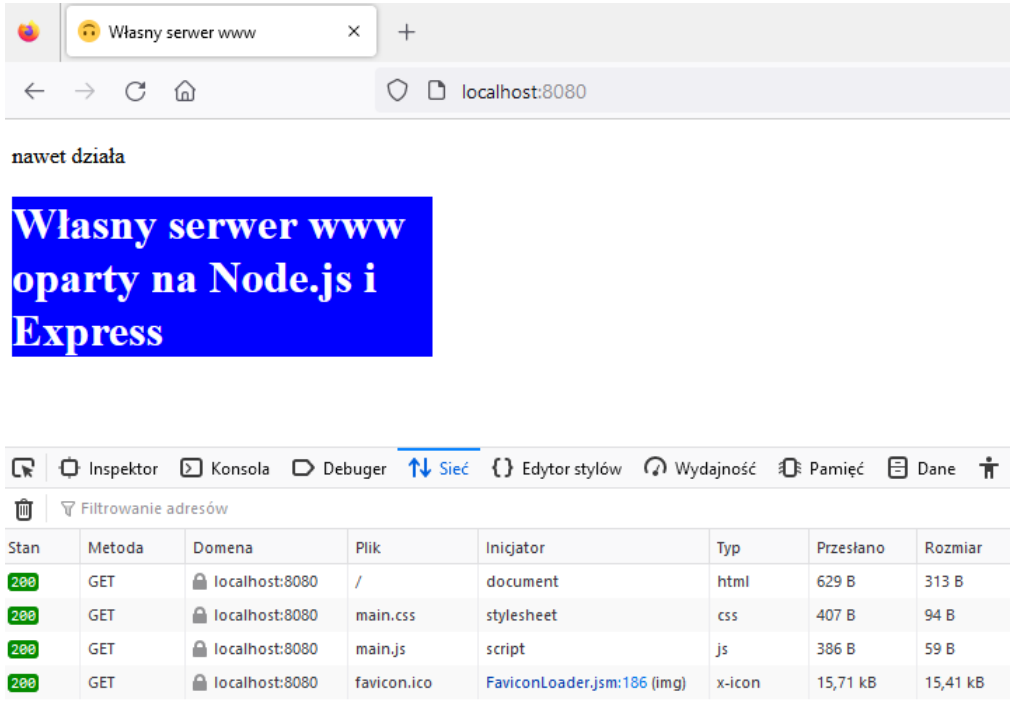
MIDDLEWARE2
- node_modules
- serwerLogs
- strona
  - favicon.ico
  - index.html
  - main.css
  - main.js
- app.js
- package-lock.json
- package.json

app.js

```
// import frameworka express
const express = require('express');
// import własnego modułu wyswietlającego logi
const serwerLogs = require('./serwerLogs');

// wywołanie frameworka
const app = express();
// port, na którym serwer nasłuchuje
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;
}

// middleware logujący czas żądania
const getTimeStampAndRequest = (req, res, next) =>{
    console.log(getTimeStamp() + ` żądanie: ${req.originalUrl}`);
    next();
};
// rejestracja pierwszego middleware
app.use(getTimeStampAndRequest);

// middleware do serwowanie (tzw. statycznych)
// plików znajdujących się w folderze strona
app.use(express.static('strona'));


// uruchomienie serwera nasłuchującego lokalnie na porcie 8080
app.listen(port, () => {
    console.log(`serwer nasłuchuje pod adresem: http://localhost:${port}`);
});
```

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

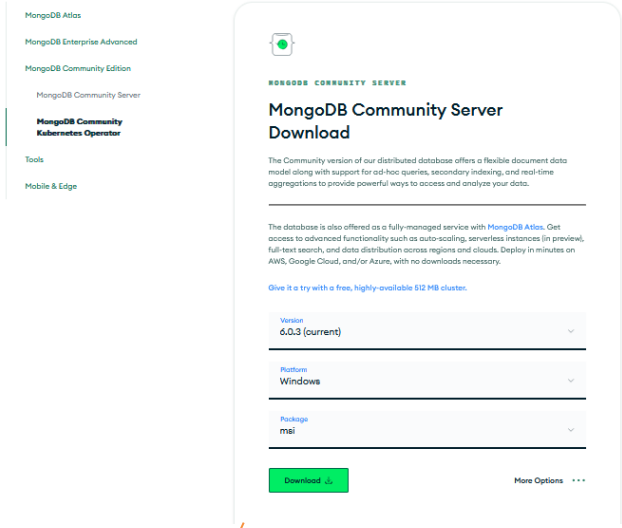
nierelacyjna (baza zawiera kolekcje,
w których przechowywane są dokumenty, baza typu noSQL, przykład: MongoDB)

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

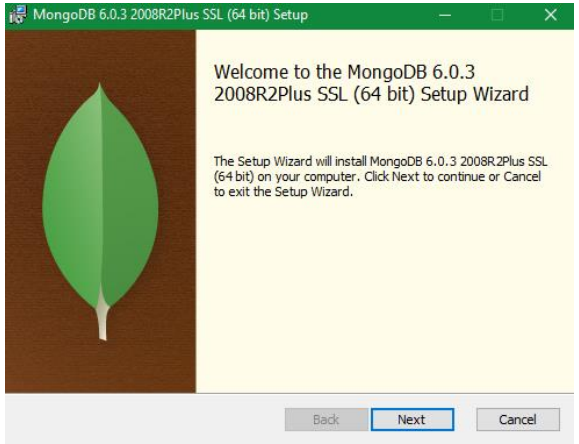
# MongoDB

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

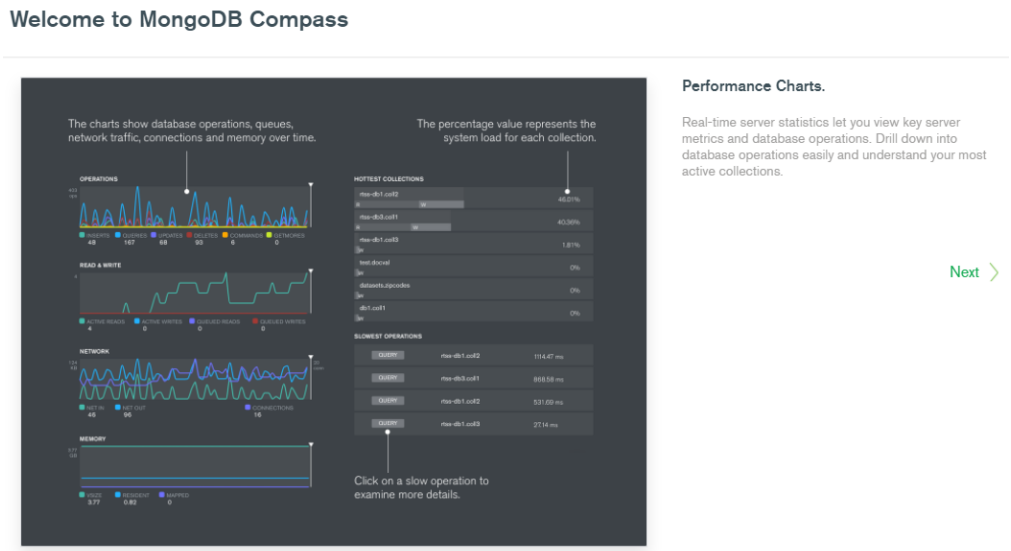


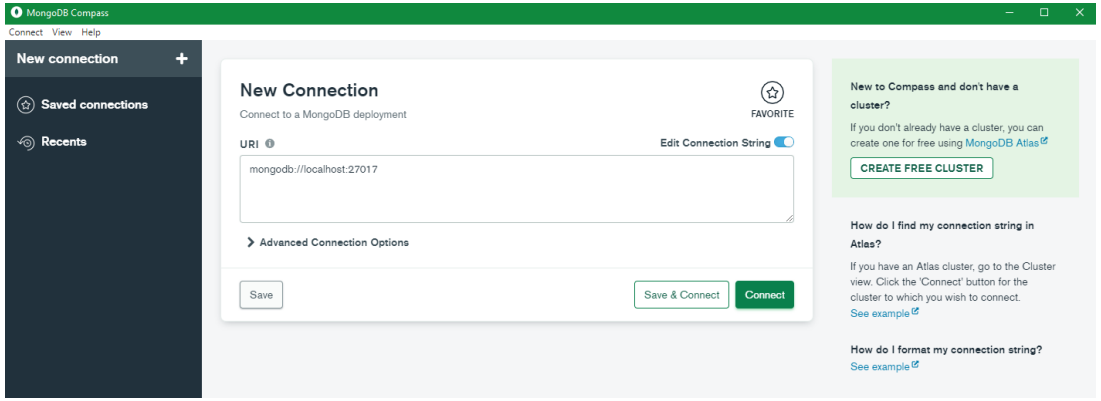

instalacja MongoDB

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)

MongoDB Documentation

Back To View & Analyze Data

MongoDB Shell

- Install mongosh
- Connect to a Deployment
- Configure mongosh
- Run Commands
- Perform CRUD Operations
- Run Aggregation Pipelines
- Client-Side Field Level Encryption
- Write Scripts
- Snippets
- Reference
- mongosh Help

## Install mongosh

### Prerequisites

To use the MongoDB Shell, you must have a MongoDB deployment to connect to.

- For a free cloud-hosted deployment, you can use MongoDB Atlas.
- To learn how to run a local MongoDB deployment, see Install MongoDB.

**Supported MongoDB Versions**

You can use the MongoDB Shell to connect to MongoDB version 4.0 or greater.

### Procedure

Select the appropriate tab for your operating system:

Windows | macOS | Linux

**NOTE**

On Windows, `mongosh` preferences and configuration options are stored in the `%APPDATA%/mongodb/mongosh` directory.

**Install from MSI**

1. Open the MongoDB Shell download page.
   Open the MongoDB Download Center.
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`.

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

TOOLS

## MongoDB Shell Download

MongoDB Shell is the quickest way to connect to (and work with) MongoDB. Easily query data, configure settings, and execute other actions with this modern, extensible command-line interface – replete with syntax highlighting, intelligent autocomplete, contextual help, and error messages.

Note: MongoDB Shell is an open source (Apache 2.0), standalone product developed separately from the MongoDB Server.

Learn more

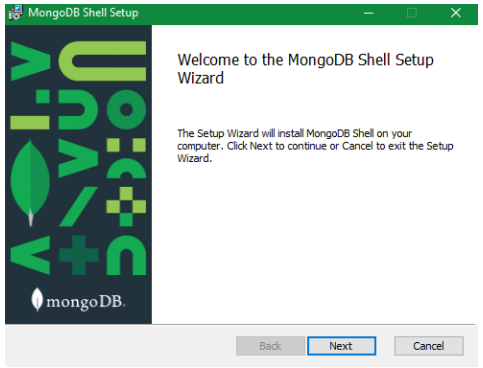
Version
1.6.0

Platform
Windows 64-bit (8.1+) (MSI)

Package
msi

Download

More Options




defaultowa ścieżka instalacji shella

C:\Users\<user>\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

Administrator: Wiersz polecenia - mongod --dbpath ..\data

```
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":"Ini
rnalClient":{"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 MongDB shella

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

```
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/mongodb-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

insertOne()

find()

```
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
  }
]
```

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

MongoDB Compass - localhost:27017/test.samochody

Connect View Collection Help

localhost:27017

4 DBS 4 COLLECTIONS

FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 6.0.3 Community

My Queries

Databases

Filter your data

admin

config

local

test

samochody

Documents
test.samochody

test.samochody

4 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

FILTER { field: 'value' } OPTIONS FIND RESET

ADD DATA VIEW

Displaying documents 1 - 4 of 4 REFRESH

```
_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
```

wszystkie dodane dokumenty w kolekcji samochody widoczne w aplikacji Compass

MongoDB shell

>_MONGOSH

# 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

OPEN EDITORS
- app.js

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

połączenie z bazą ***local*** i wybranie pierwszego dokumentu w kolekcji ***samochody***

app.js > ...

```
//https://www.mongodb.com/docs/get-started/?language=nodejs

const { MongoClient } = require('mongodb');

async function runGetStarted() {
  // Replace the uri string with your connection string
  const uri = "mongodb://127.0.0.1:27017/";
  const client = new MongoClient(uri);

  try {
    const database = client.db('local');
    const cars = database.collection('samochody');

    // Pierwszy dokument w kolekcji samochody:
    const query = {};
    const car = await cars.findOne(query);

    console.log(car);
  } finally {
    await client.close();
  }
}
runGetStarted().catch(console.dir);
```

//https://www.mongodb.com/docs/get-started/?language=nodejs

# 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

## Zadanie

Na kolejnych slajdach znajdują się kolejne funkcjonalności komunikacji z bazą danych MongoDB, ale niestety jest to starszy kod. Niektóre elementy są ~~przekreślone~~ czyli oznaczone w ten sposób jako *depricated (przestarzałe)*
Zadanie polega na uzyskaniu danej funkcjonalności (np. wpisaniu rekordu do bazy), ale za pomocą nowszej wersji kodu.
(wzór nowszej wersji kodu połączenia z bazą MongoDB znajduje się na slajdzie nr **82**)

# Node.js MongoDB

EXPLORER

OPEN EDITORS
- app.js
- app0.js

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

app.js | app0.js

app0.js > ...

```javascript
// konfiguracja klienta
var url = "mongodb://127.0.0.1:27017/";
var MongoClient = require('mongodb').MongoClient;

// połaczenie z bazą test
MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    console.log("Połączenie z MongoDB ustanowione");
    var dbo = db.db("local");
    // wybranie pierwszego dokumentu w kolekcji samochody
    dbo.collection("samochody").findOne({}, function(err, result) {
        if (err) throw err;
        console.log(`
            Pierwszy dokument w kolekcji samochody:
            id: ${result.id},
            marka: ${result.marka},
            powypadkowy: ${result.powypadkowy},
        `);
        db.close();
    });
});
```

połączenie z bazą ***local*** i wybranie pierwszego dokumentu w kolekcji ***samochody*** ***(depricated)***

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

# Node.js MongoDB

EXPLORER

OPEN EDITORS
- app.js
- app1.js

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

app.js | app1.js

app1.js > ...

```javascript
// konfiguracja klienta
var url = "mongodb://127.0.0.1:27017/";
var MongoClient = require('mongodb').MongoClient;

// połaczenie z bazą test
MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    console.log("Połączenie z MongoDB ustanowione");
    var dbo = db.db("local");
    // wybranie wszytkich dokumentów w kolekcji samochody
    dbo.collection("samochody").find({}).toArray(function(err, result) {
        if (err) throw err;
        console.log(result);
        db.close();
      });
  });
```

połączenie z bazą ***local*** i wybranie wszystkich dokumentów z kolekcji ***samochody*** ***(depricated)***

# Node.js MongoDB

EXPLORER

OPEN EDITORS
- app.js
- app1.js
- app2.js
- app3.js
- app4.js

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

app.js | app1.js | app2.js | app3.js | app4.js

app2.js > ...

```
// konfiguracja klienta
var url = "mongodb://127.0.0.1:27017/";
var MongoClient = require('mongodb').MongoClient;

// połaczenie z bazą test
MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    console.log("Połączenie z MongoDB ustanowione");
    var dbo = db.db("local");
    // wybranie dokumentów dla powypadkowych samochodów w kolekcji samochody
    dbo.collection("samochody").find({powypadkowy : false}).toArray(function(err, result)
        if (err) throw err;
        console.log(result);
        db.close();
      });
  });
```

połączenie z bazą ***local*** i wybranie dokumentów dla powypadkowych samochodów z kolekcji ***samochody (depricated)***

# Node.js MongoDB

EXPLORER

OPEN EDITORS
- app.js
- app1.js
- app2.js
- app3.js
- *app4.js*

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

wstawienie nowego dokumentu do kolekcji samochody ***(depricated)***

app.js | app1.js | app2.js | app3.js | *app4.js*

app3.js > ...

```js
// konfiguracja klienta
var url = "mongodb://127.0.0.1:27017/";
var MongoClient = require('mongodb').MongoClient;

// połaczenie z bazą test
MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    console.log("Połączenie z MongoDB ustanowione");
    var dbo = db.db("local");
    // wstawienie nowego dokumentu w kolekcji samochody
   dbo.collection("samochody").insertOne(
        { marka: "Opel", model: "Corsa", powypadkowy: false },
        function(err, res) {
            if (err) throw err;
            console.log("Dokument dodany");
            db.close();
        }
    );
  });
```

# Node.js MongoDB

EXPLORER

OPEN EDITORS
- app4.js

MONGODB
- node_modules
- app.js
- app0.js
- app1.js
- app2.js
- app3.js
- app4.js
- package-lock.json
- package.json
- samochody.txt

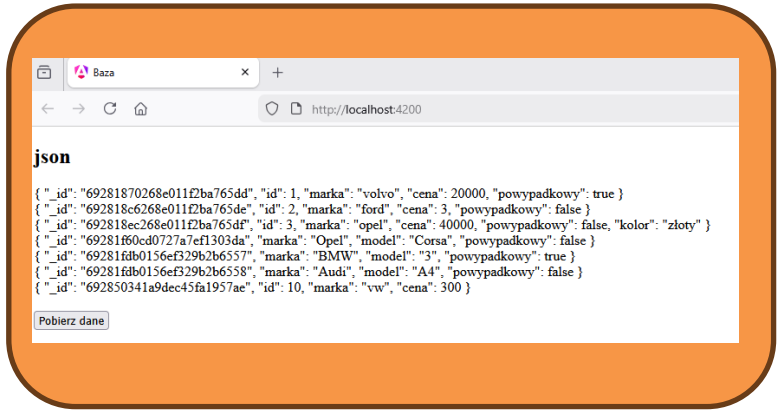
app4.js > ...

```js
// konfiguracja klienta
var url = "mongodb://127.0.0.1:27017/";
var MongoClient = require('mongodb').MongoClient;

// połaczenie z bazą test
MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    console.log("Połączenie z MongoDB ustanowione");
    var dbo = db.db("local");
    // wstawienie wielu dokumentów do kolekcji samochody
   dbo.collection("samochody").insertMany(
    [
        { marka: "BMW", model: "3", powypadkowy: true },
        { marka: "Audi", model: "A4", powypadkowy: false }
    ],
    function(err, res) {
        if (err) throw err;
        console.log("Dodano dokumentów: " + res.insertedCount);
        db.close();
    }
    );
  });
```

wstawienie dwóch nowych dokumentów do kolekcji samochody ***(depricated)***

# Połączenie z bazą danych MongoDB poprzez api

Angular (frontend)




komunikacja Angulara z bazą danych MongoDB poprzez server Node.js (backend)

response

request

response

request

http://localhost:3000/samochody

"/samochody" => Samochod.find();

baza danych „local”
kolekcja „samochody”

server Node.js Express nasłuchuje na http://localhost:3000 (endpoint)

# Połączenie z bazą danych MongoDB
# Node.js Express (backend)

EXPLORER

OPEN EDITORS
- index.js

NODESERVER
- node_modules
- index.js
- package-lock.json
- package.json

index.js > ...

```javascript
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');

const app = express();

app.use(cors());  // pozwala na wszystkie originy
                  // „Same Origin Policy”

mongoose.connect('mongodb://localhost:27017/local')
  .then(() => console.log("Połączono z MongoDB"))
  .catch(err => console.error("Błąd połączenia:", err));

// Schema
const SamochodSchema = new mongoose.Schema({
  id: Number,
  marka: String,
  cena: Number,
  powypadkowy: Boolean
});

// Model → kolekcja "samochody"
const Samochod = mongoose.model('Samochod', SamochodSchema, 'samochody');

// wypisz wszystkie dokumenty z kolekcji samochody
app.get('/samochody', async (req, res) => {
  try {
    const auta = await Samochod.find();
    res.json(auta);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

app.listen(3000, () => console.log("Serwer działa na http://localhost:3000"));
```

komendy w terminalu:

**npm init –y**
**npm install express**
**npm install express mongoose**
**npm install cors**

uruchomienie serwera:

**node index.js**

```
Serwer działa na http://localhost:3000
Połączono z MongoDB
```

# Połączenie z bazą danych MySQL
# Angular (frontend)

app.ts

EXPLORER

OPEN EDITORS

app.ts src\app M

BAZA

- .angular
- .vscode
- node_modules
- public
- src
  - app
    - app.config.ts M
    - app.css
    - app.html M
    - app.routes.ts
    - app.spec.ts
    - app.ts M
    - service.ts M
  - index.html
  - main.ts M
  - styles.css
- .editorconfig
- .gitignore
- angular.json
- package-lock.json
- package.json
- README.md
- tsconfig.app.json
- tsconfig.json
- tsconfig.spec.json

app.ts M

src > app > app.ts > ...

You, 56 seconds ago | 1 author (You)

```
1  import { Component } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { ApiService } from './service';
4
   You, 56 seconds ago | 1 author (You)
5  @Component({
6    selector: 'app-root',
7    imports: [CommonModule],
8    templateUrl: './app.html',
9    styleUrl: './app.css'
10 })
11 export class App {
12   dane: any;   // kontener na dane z bazy danych
13
14   // w konstruktorze klasy jako argument podajemy serwis
15   // łaczący naszą aplikację z endpointem
16   // (serwer Node.js)
17   constructor(private apiService: ApiService) {}
18
19   // metoda pobierająca dane z endpointu
20   pobierzDane() {
21     this.apiService.pobierzDane().subscribe({
22       next: (response) => {
23         this.dane = response;
24         console.log('Dane z API:', response);
25       },
26       error: (err) => {
27         console.error('Błąd przy pobieraniu danych:', err);
28       }
29     });
30   }
31 }
```

komenda w terminalu:

**ng new baza**

Angular ver.20

# Połączenie z bazą danych MongoDB
# Angular (frontend)

plik *service.ts*
tworzymy ręcznie
lub poleceniem:
**`ng g s nazwa_serwisu`**

service.ts serwis do połączenia z endpointem

EXPLORER

OPEN EDITORS
- service.ts baza\src\app M

ANGULARFRONTEND
- baza
  - .angular
  - .vscode
  - node_modules
  - public
  - src
    - app
      - app.config.ts M
      - app.css
      - app.html M
      - app.routes.ts
      - app.spec.ts
      - app.ts M
      - service.ts M

baza > src > app > service.ts > ...

You, 34 minutes ago | 1 author (You)

```
1  import { HttpClient } from '@angular/common/http';
2  import { Injectable } from '@angular/core';
3
4  // serwis łączący aplikację z serwerem Node.js
5  // nasłuchującym na porcie 3000 (endpoint)
6  // wystawiającym API /samochody
7  // które służy pobrania danych z bazy MongoDB
8  @Injectable({              // serwis można używać
9    providedIn: 'root'       // w całej angularowej aplikacji
10 })
11 export class ApiService {
12   private apiUrl = 'http://127.0.0.1:3000/samochody';
13
14   constructor(private http: HttpClient) { }
15
16   pobierzDane() {
17     return this.http.get(this.apiUrl);
18   }
19 }
```

adres API

Angular ver.20

# Połączenie z bazą danych MySQL
# Angular (frontend)

app.config.ts

EXPLORER

OPEN EDITORS

TS app.config.ts baza\src\app M

ANGULARFRONTEND

- baza
  - .angular
  - .vscode
  - node_modules
  - public
  - src
    - app
      - TS app.config.ts M
      - # app.css
      - app.html M

TS app.config.ts M

baza > src > app > TS app.config.ts > ...

You, 1 second ago | 1 author (You)

```
1  import { ApplicationConfig, provideBrowserGlobalErrorListeners, provideZoneChangeDetection } from '@angular/core';
2  import { provideRouter } from '@angular/router';
3  import { provideHttpClient } from '@angular/common/http';     // <====
4
5  import { routes } from './app.routes';
6
7  export const appConfig: ApplicationConfig = {
8    providers: [
9      provideBrowserGlobalErrorListeners(),
10     provideZoneChangeDetection({ eventCoalescing: true }),
11     provideRouter(routes),
12     provideHttpClient()   // <=======
13   ]
14 };
15
```

Angular ver.20

# Połączenie z bazą danych MySQL
# Angular (frontend)

app.html

EXPLORER

OPEN EDITORS

app.html baza\src\app M

ANGULARFRONTEND

- baza
  - .angular
  - .vscode
  - node_modules
  - public
  - src
    - app
      - app.config.ts M
      - app.css
      - app.html M

app.html M

baza > src > app > app.html > ...

Go to component | You, 1 second ago | 1 author (You)

```
<h2>json</h2>
<div *ngFor="let item of dane">{{ item | json }}</div>
<br />
<button (click)="pobierzDane()">Pobierz dane</button>
```

uruchomienie Angulara w terminalu:

ng serve

Angular ver.20

# Połączenie z bazą danych MongoDB
# baza MongoDB

MongoDB Compass - localhost:27017/local.samochody

Connections Edit View Collection Help

Compass

My Queries

Data Modeling

CONNECTIONS (8)

Search connections

- localhost:27017
  - admin
  - config
  - local
    - samochody
    - startup_log
- localhost:27017
- localhost:27017
- localhost:27017
- localhost:27017
- localhost:27017
- localhost:27017
- localhost:27017

samochody

localhost:27017 > local > samochody

Documents 7 | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

ADD DATA | EXPORT DATA | UPDATE | DELETE


```
{
  "_id": {
    "$oid": "69281870268e011f2ba765dd"
  },
  "id": 1,
  "marka": "volvo",
  "cena": 20000,
  "powypadkowy": true
}
{
  "_id": {
    "$oid": "692818c6268e011f2ba765de"
  },
  "id": 2,
  "marka": "ford",
  "cena": 3,
  "powypadkowy": false
}
{
  "_id": {
    "$oid": "692818ec268e011f2ba765df"
  },
  "id": 3,
  "marka": "opel",
  "cena": 40000,
  "powypadkowy": false,
  "kolor": "złoty"
}
```

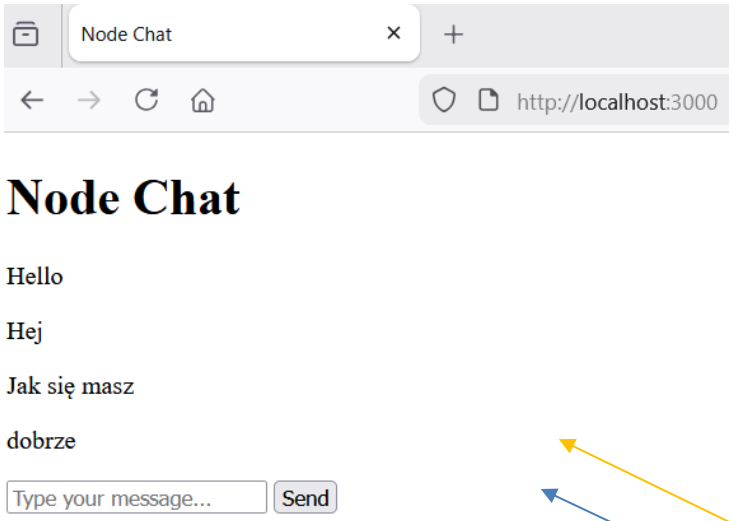

baza *local* kolekcja *samochody*

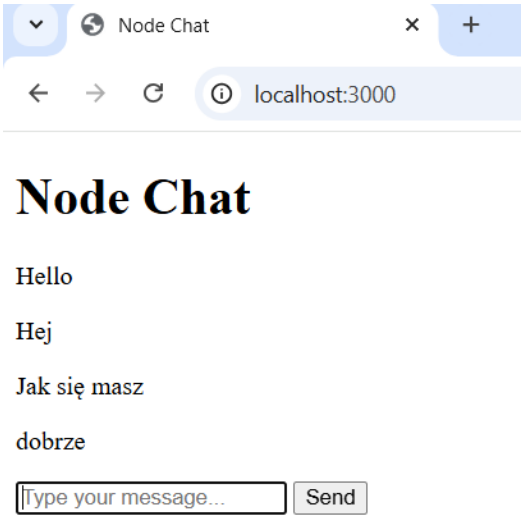
# Socket.IO

Socket.IO to biblioteka JavaScript, która umożliwia dwukierunkową, opartą na zdarzeniach komunikację w czasie rzeczywistym między klientami a serwerami internetowymi. Została zaprojektowana do działania na każdej platformie, przeglądarce i urządzeniu, kładąc nacisk na niezawodność i szybkość.

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

# Socket.IO




serwer po otrzymaniu wiadomości rozsyła ją do wszystkich podłączonych klientów (stron)

Serwer Node.js
http://localhost:3000

```
Server running on port 3000
Message received: Hello
Message received: Hej
Message received: Jak się masz
Message received: dobrze
```

# Socket.IO

app.js

EXPLORER

OPEN EDITORS
- app.js

SOCKET
- node_modules
- public
  - index.html
- app.js
- package-lock.json
- package.json

app.js > io.on('connection') callback

```javascript
const express = require('express');
const http = require('http');
const { Server } = require('socket.io');
const path = require('path');

const app = express();

const server = http.createServer(app);
const io = new Server(server);

// Serve static files
app.use(express.static(path.join(__dirname, 'public')));

// Simple route
app.get('/', (req, res) => {
;  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

```

komendy w terminalu:

**npm init –y**
**npm install express**
**npm install socket.io**
**npm install socket.io-client**

uruchomienie serwera:

**node index.js**

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

# Socket.IO

app.js

EXPLORER

OPEN EDITORS

app.js

SOCKET

node_modules

public

index.html

app.js

package-lock.json

package.json

app.js > io.on('connection') callback

```
18
19  // Socket.IO connection handler
20  io.on('connection', (socket) => {
21    console.log('A user connected');
22
23    // Handle new messages
24    socket.on('chat message', (msg) => {
25      console.log('Message received:', msg);
26      // Broadcast the message to all connected clients
27        io.emit('chat message', msg);
28      });
29
30    // Handle disconnection
31    socket.on('disconnect', () => {
32      console.log('A user disconnected');
33    });
34  });
35
36  const PORT = process.env.PORT || 3000;
37  server.listen(PORT, () => {
38    console.log(`Server running on port ${PORT}`);
39  });
```

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

index.html

# Socket.IO

EXPLORER

OPEN EDITORS

index.html public

SOCKET

node_modules

public

index.html

app.js

package-lock.json

package.json

index.html

public > index.html > html > body > script

```
1
2  <!DOCTYPE html>
3  <html>
4  <head>
5    <title>Node Chat</title>
6  </head>
7  <body>
8    <h1>Node Chat</h1>
9    <div id="messages"></div>
10   <form id="form" action="#">
11     <input id="input" autocomplete="off" placeholder="Type your message..." />
12    <button>Send</button>
13    </form>
14
15   <!--<script src="/socket.io/socket.io.js"></script> -->
16   <script src="https://cdn.socket.io/4.5.0/socket.io.min.js"></script>
17   <script>
18     const socket = io();
19     const form = document.getElementById('form');
20     const input = document.getElementById('input');
21     const messages = document.getElementById('messages');
22
```

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

index.html

# Socket.IO

EXPLORER

OPEN EDITORS

index.html public

SOCKET

node_modules

public

index.html

app.js

package-lock.json

package.json

index.html

public > index.html > ...

```html
<html>
<body>
  <script>
    // Handle form submission
    form.addEventListener('submit', (e) => {
        e.preventDefault();
        const message = input.value.trim();
        if (message) {
            // Emit the message to the server
            socket.emit('chat message', message);
                // Clear the input
                input.value = '';
            }
        });

        // Listen for incoming messages
        socket.on('chat message', (msg) => {
            const item = document.createElement('p');
            item.textContent = msg;
            messages.appendChild(item);
        });
    </script>
</body>
</html>
```

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

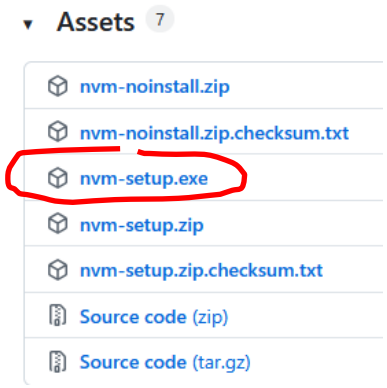
# help

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)

# Node.js upgrade to latest version (Windows)

1. install nvm (program zarządzający wersjami node.js)

   https://github.com/coreybutler/nvm-windows/releases




2. open cmd (Command Prompt)

3. write command:

```
nvm install latest
nvm use latest
```

4. more commands:

```
nvm
```

```
Running version 1.2.2.

Usage:

  nvm arch                     : Show if node is running
  nvm current                  : Display active version.
  nvm debug                    : Check the NVM4W process
  nvm install <version> [arch] : The version can be a spe
                                 most recent LTS version.
```