# Node.js

Szybkie wprowadzenie do jednego ze środowisk działania JavaScript

Michał Jabłoński

# Charakterystyka Node.js

- Idea: „JavaScript Everyware” - Modułowość JavaScriptu
- Uruchamianie kodu JavaScript bezpośrednio
- Uruchamianie kodu JavaScript – z pliku
- Require oraz module.exports – łączenie plików JavaScript
- Zależności i plik **package.json**
- Package manager (**NPM**)
- Skrypty npm
- node_modules/ - „library root”
- INSTALACJA: **https://nodejs.org/en/download/**

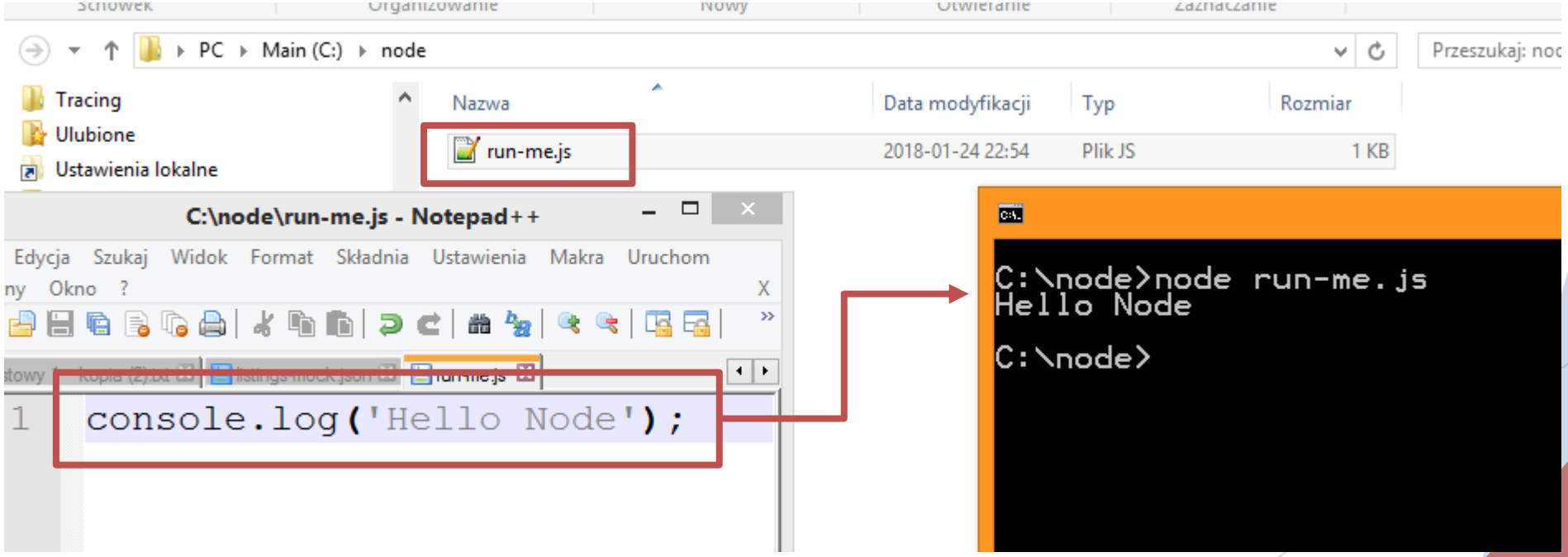
# Uruchamianie kodu JavaScript bezpośrednio

```
C:\Users\Michal>node
> const hello = 'Hello World';
undefined
> const two = 2;
undefined
> `result is: ${hello} ----> ${two}`
'result is: Hello World ----> 2'
>
```

# Środowisko – Node.js

https://nodejs.org/en/

- GAME – CHANGER !
- Idea:  commonJS + rozwój modularyzacji JavaScriptu
- JavaScript runtime | Server Side JavaScript



# Uruchamianie kodu JavaScript – z pliku

simple-file.js

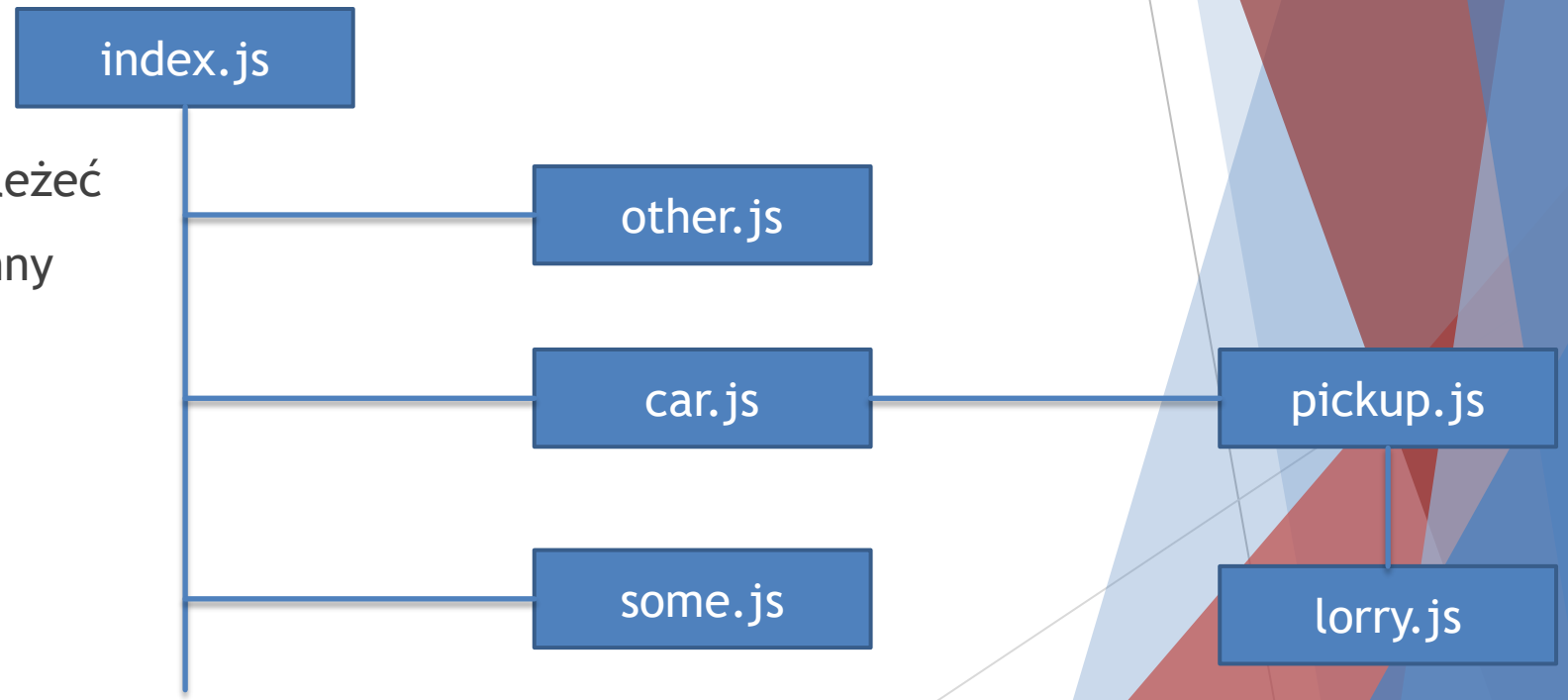
```
1
2   console.log('Hello World');
3
4   const twoPlusTwo = 2+2;
5
6   console.log('2+2=', twoPlusTwo);
```

```
> node .\simple-file.js
```

```
Hello World
2+2= 4
```

# Idea: „JavaScript Everyware” - Modułowość JavaScriptu

- Pliki mogą wzajemnie od siebie zależeć
- Jeden może „importować” to co inny „exportuje”
- Każdy plik ma swój zasięg i nie ma dostępu do zasięgu innego pliku
- Każdy plik traktowany jest jako oddzielny moduł



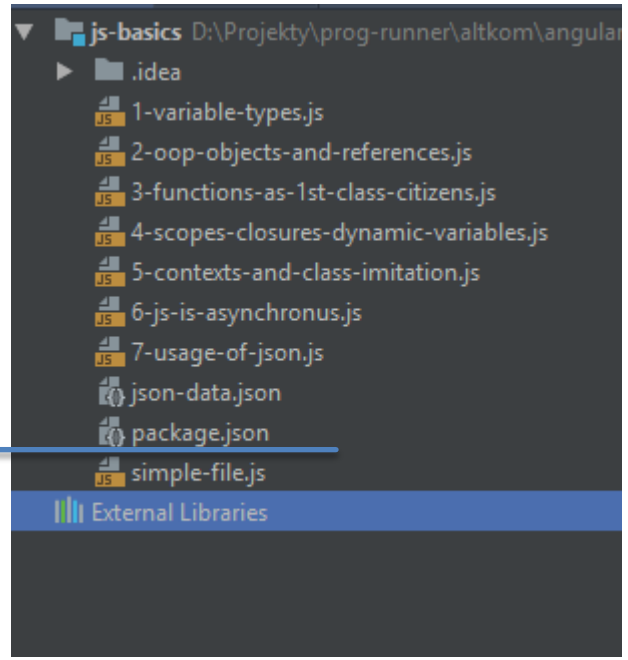
# Require – łączenie plików JavaScript

```
const camera = require('./camera');
const keyboard = require('./keyboard');

console.log({
    name: 'Laptop unit',
    camera: camera,
    keyboard: keyboard
});
```

```
{ name: 'Laptop unit',
  camera:
   { mpx: 12,
     status: 'NEW',
     brand: 'Logitech',
     getView: [Function: getView] },
  keyboard: { status: 'USED', brand: 'Razer', getWSAD: [Function: getWSAD] } }
```

# Zależności i plik **package.json**




```
"dependencies": {
  "@angular/animations": "^4.0.0",
  "@angular/common": "^4.0.0",
  "@angular/compiler": "^4.0.0",
  "@angular/compiler-cli": "^4.0.0",
  "@angular/core": "^4.0.0",
  "@angular/forms": "^4.0.0",
  "@angular/http": "^4.0.0",
  "@angular/platform-browser": "^4.0.0",
  "@angular/platform-browser-dynamic": "^4.0.0",
  "@angular/platform-server": "^4.0.0",
  "@angular/router": "^4.0.0",
  "bootstrap": "^3.3.7",
  "core-js": "^2.4.1",
},
"devDependencies": {
  "@angular/cli": "1.0.0",
  "@angular/compiler-cli": "4.0.0",
  "@types/jasmine": "2.5.38",
  "@types/node": "~6.0.60",
  "codelyzer": "~2.0.0",
  "jasmine-core": "~2.5.2",
  "jasmine-spec-reporter": "~3.2.0",
```

# Node Package Manager

- domyślny manager pakietów dla środowiska Node.js
- **https://www.npmjs.com/**


- Można instalować zależności „dependencies” | bibilioteki JavaScript
  - Przykłady:
  - webpack, angular, reactjs, jquery, bootstrap….
  - przykładowe wywołanie:

```
> npm install webpack -D
```

# Nowy projekt w Node.js

- Przygotowujemy dedykowany pusty katalog np.

```
my-project
```

- W środku uruchamiamy komendę:

```
npm init -y
```

- Przygotuje ona nam plik „pavckage.json” – opcja „-y” to „yes to all” (Przyjmujemy standardowe wartości pliku package.json”

- wg. konwencji – przygotowujemy w środku folder „src” w którym będą znajdowały się pliki źródłowe naszej aplikacji

# Dodawanie zależności devDependencies

- W katalogu z projektem – wykonujemy kolejne polecenia za pomocą node package managera

```
npm install –D eslint
```

# Własne skrypty w package.json

- Dzięki temu iż pakiet eslint posiada swoje „cli” (command line interface) możemy używać skryptów w odniesieniu do nowo powstałego katalogu „node_modules”
- Przykładowo do package.json możemy dodać w sekcji „scripts” zapis JSON:

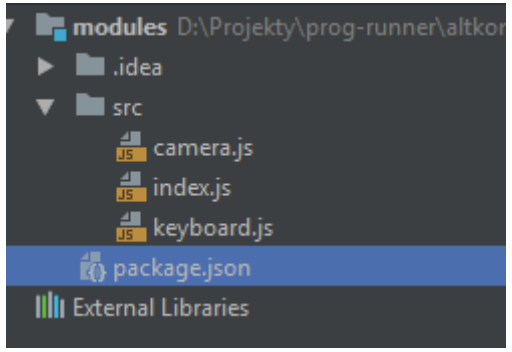
```
"scripts": {
   "lint-init" : "eslint --init"
}
```

- W tym układzie „lint-init” to nazwa którą musimy wowołać poleceniem:

```
npm run lint-init
```

- Natomiast kod „eslint –-init” to treść naszego skyptu – polecenie, które zostanie wykoanne

# Skrypty npm



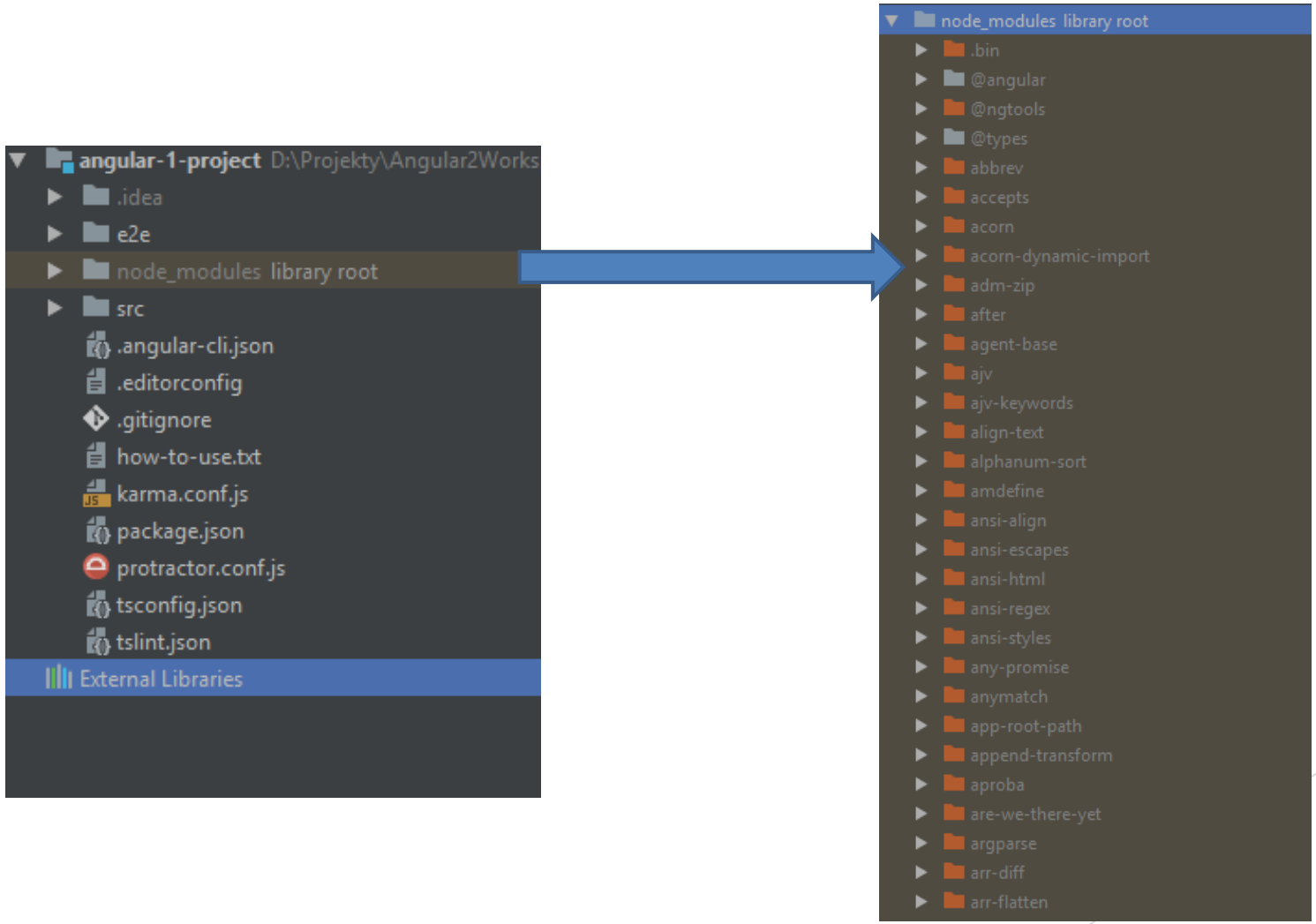

```
{
  "name": "modules",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node src/index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

```
> npm run start
```

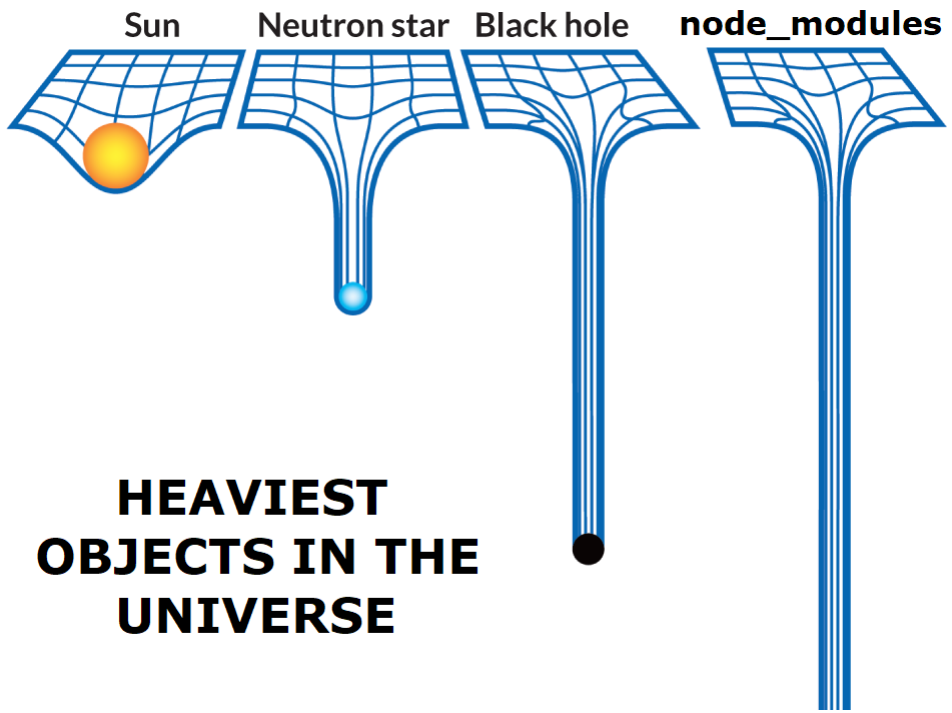
Wyjątkowo zadziała również:

```
> npm start
```

# node_modules/ - „library root”



# node_modules
# - nie wysyłamy na repozytorium kodu ☺




Źródło: https://i.redd.it/tfugj4n3l6ez.png