



# WYKŁAD 9

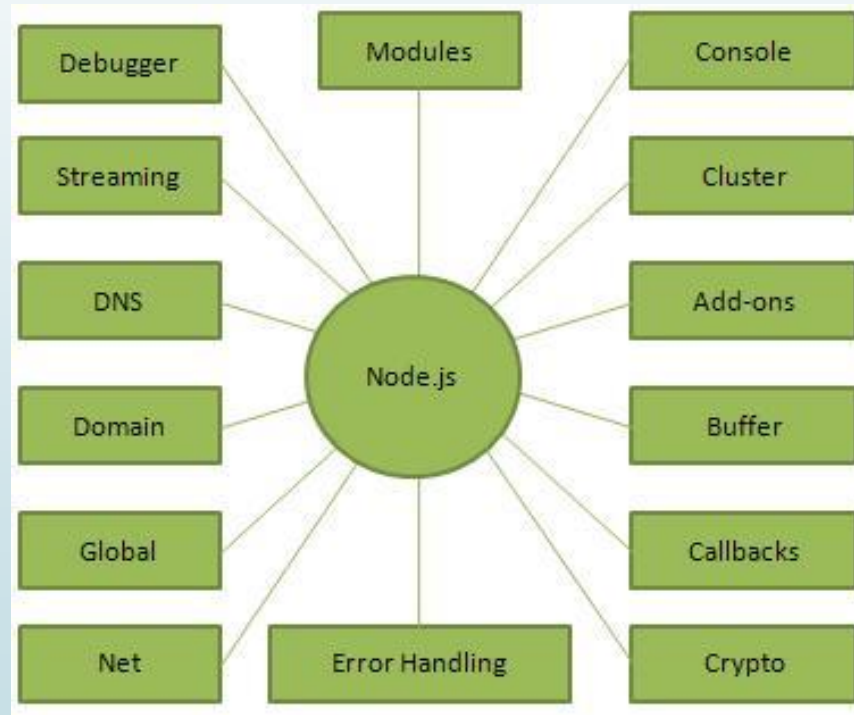
## Node.js c.d.

dr inż. Jacek Paluszak

# Node.js

## Wykorzystanie i przykład

`node [options] [V8 options] [script.js | -e "script" | - ] [arguments]`





## Wykorzystanie i przykład

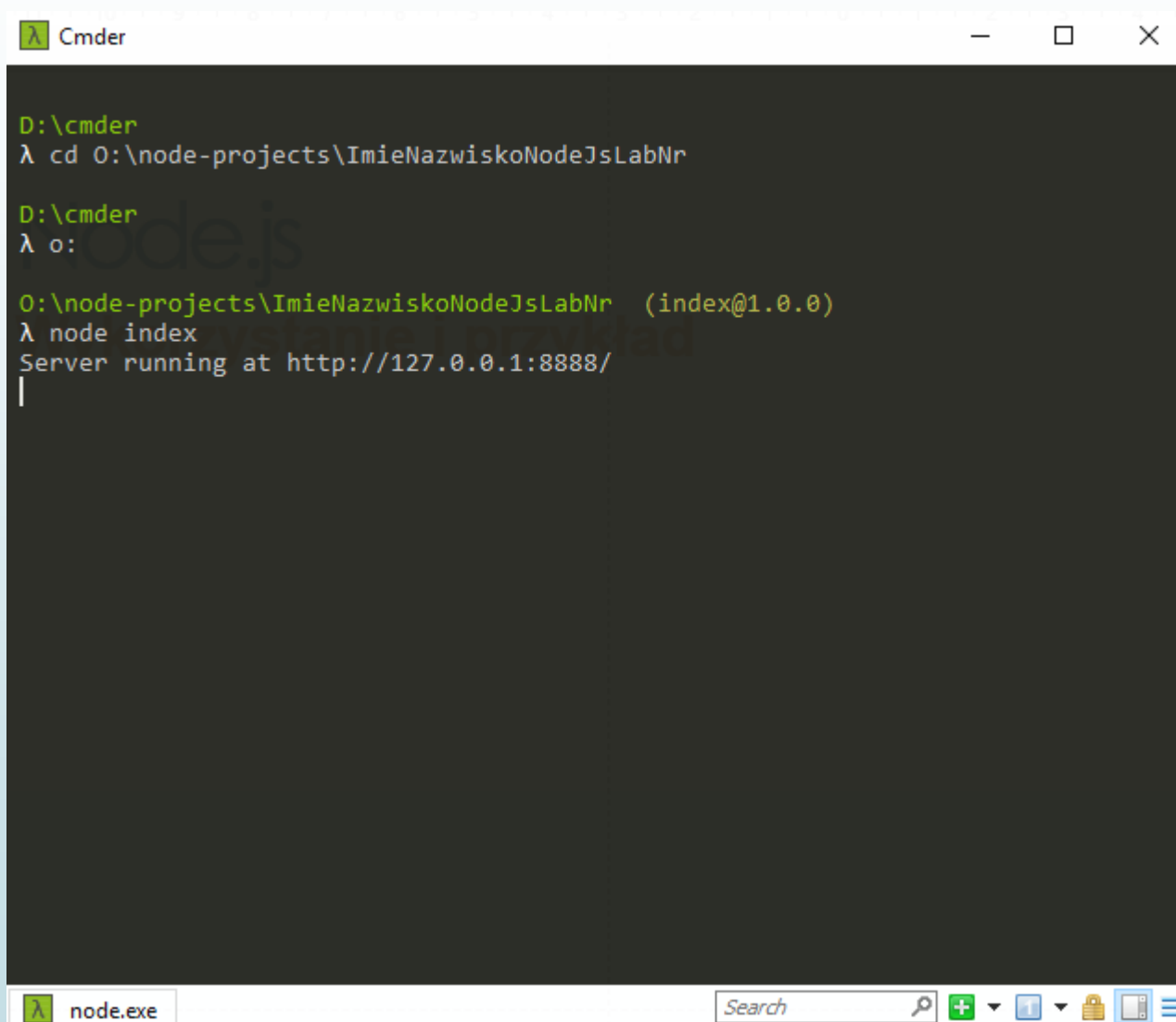
Obszary, w których Node.js jest najczęściej wykorzystywany:

- Aplikacje I/O
- Aplikacje do strumieniowego przesyłania danych
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

**Płynne połączenie backendu z frontendem.**

# Node.js

## Wykorzystanie i przykład



```
λ Cmder
D:\cmdr
λ cd 0:\node-projects\ImieNazwiskoNodeJsLabNr

D:\cmdr
λ o:

0:\node-projects\ImieNazwiskoNodeJsLabNr (index@1.0.0)
λ node index
Server running at http://127.0.0.1:8888/
|
```

The screenshot shows a Windows Command Prompt window titled "Cmdr". The user navigates to the directory "0:\node-projects\ImieNazwiskoNodeJsLabNr" and runs the command "node index". The output shows the server is running at "http://127.0.0.1:8888/". The taskbar at the bottom shows "node.exe" running.

# | Node.js

## Wykorzystanie i przykład

- parametr uruchomienia  
`node hello.js`

- REPL - Read-Eval-Print-Loop

```
$ node
```

```
> console.log(`Hello ${process.env.USER}!`);
```

# | Node.js

## Wykorzystanie i przykład

Aplikacje pisane w Node.js posiadają **modułową** architekturę.

Każdy plik **.js** stanowi swój odrębny moduł, a zmienne, które w nim deklarujemy posiadają **scope** w obrębie tego modułu.

W odróżnieniu od czystego **javascriptu** w przeglądarkach, gdzie

```
var zmienna = 5
```

znajduje w **scopie globalnym**

# | Node.js

## Wykorzystanie i przykład

Node.js również posiada **obiekty globalne**, a zdefiniowane w nich metody dostępne są dla programistów w każdym miejscu tworzonej aplikacji.

Ich lista znajduje się w oficjalnej dokumentacji

<https://nodejs.org/api/globals.html>

np.

```
console.log()
```

# | Node.js

## Wykorzystanie i przykład

Jedną z ważniejszych funkcji dostępnych globalnie jest:

`require()`

Używa się jej do importowania modułów Node.js:

```
var path = require("path")
```

**Path** to moduł dostępny natywnie w Node.js od razu po jego zainstalowaniu i zgodnie ze swoją nazwą – zajmuje się zarządzaniem ścieżkami’.



# | Node.js

## Wykorzystanie i przykład

```
var path = require("path");  
console.log(path.basename(__filename));
```

**basename()** to metoda w module **path**, która w połączeniu z parametrem **\_\_filename** wyrzuca nazwę pliku (z którego została wywołana).

Jeśli chcemy uzyskać nazwę aktualnego katalogu, możemy użyć parametru **\_\_dirname**.

```
var path = require("path");  
console.log(path.basename(__dirname));
```

```
//pełna ścieżka dostępu do katalogu:  
console.log(path.join(__dirname, 'uploads', 'images'));
```

# | Node.js

## Wykorzystanie i przykład

Innym modułem dostępnym natywnie w Node.js jest moduł **util**.

```
var util = require("util");
```

Moduł ten posiada sporo **funkcji-helperów** ułatwiających proces tworzenia aplikacji, a szczególnie własnych modułów.

```
np.  
log()
```

Możemy jej użyć **zamiast** javascriptowej standardowej **console.log()**.

```
util.log(path.join(__dirname, 'uploads', 'images'));
```

oprócz ścieżki dostępu mamy też czas wykonania np. 04 września o 10:56.

# Node.js

## Wykorzystanie i przykład

Jako, że Node.js jest ,zasilany' preprocesorem **v8** z przeglądarki Google Chrome, możemy użyć tego modułu i jego metod do np. **sprawdzenia aktualnego zużycia pamięci serwera.**

```
getHeapStatistics()
```

np.

```
const util = require("util");
```

```
// importujemy moduł v8 do stałej v8:  
const v8 = require("v8");
```

```
// wyświetlamy informacje o aktualnym zużyciu pamięci:  
util.log(v8.getHeapStatistics());
```

# | Node.js

## Menadżer pakietów – npm (Node Package Manager)

npm to menedżer pakietów JavaScript współdziałający z rejestrem

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

instalowanie, zarządzanie zależnościami

```
npm install [ package [ @version ] ]  
npm install -g <package [ @version ]>  
npm install --save <package [ @version ]>  
npm install --save -dev <package [ @version ]>
```

**-g** instalujemy moduł globalnie

**--save** stosujemy dla zależności, które są niezbędne dla uruchomienia projektu.

**--save -dev** – dla zależności, które są niezbędne do jego tworzenia

# | Node.js

## Menadżer pakietów – npm (Node Package Manager)

```
npm install nodemon -g
```

umożliwia automatyczny restart serwera po zapisaniu zmian:

```
nodemon index.js
```

**nodemon** musi być instalowany globalnie, tak aby korzystać z niego mogła każda aplikacja napisana z wykorzystaniem Node.js

Będzie nasłuchiwał zmian wprowadzonych w pliku **index.js** i restartował serwer za każdym razem, gdy zapiszesz zmiany.

# | Node.js

## Menadżer pakietów – npm (Node Package Manager)

`npm init`

stworzy to nam plik

`package.json`

zawiera informacje na temat naszego takie jak:

- nazwa
- strona domowa
- opis
- autorzy itp.

czyli to co byłoby przydatne gdybyśmy chcieli nasz projekt opublikować tak by inni programiści mogli ściągnąć naszą paczkę za pomocą npm.

Plik ten zawiera informacje z jakich paczek nasz projekt będzie korzystał lub korzysta.

# Node.js

## Menadżer pakietów – npm (Node Package Manager)

packages.json

```
1 {  
2   "name": "index",  
3   "version": "1.0.0",  
4   "description": "Laboratorium Node.js",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "Imię Nazwisko",  
10  "license": "ISC",  
11  "keywords": [  
12    "node"  
13  ],  
14  "dependencies": {  
15    "nodemon": "^1.18.7"  
16  }  
17 }
```

```
{}  
S name: "index"  
S version: "1.0.0"  
S description: "Laboratorium Node.js"  
S main: "index.js"  
▲ {} scripts  
S test: "echo \"Error: no test specified\" && exit 1"  
S author: "Imię Nazwisko"  
S license: "ISC"  
▲ [] keywords  
S : "node"  
▲ {} dependencies  
S nodemon: "^1.18.7"
```

# | Node.js

## Menadżer pakietów – npm (Node Package Manager)

```
npm install
```

Polecenie to instaluje WSZYSTKIE paczki zadeklarowane w `package.json`.

```
"dependencies": {  
  "nodemon": "^1.18.7"  
}
```



DZIĘKUJĘ ZA UWAGĘ