

# NodeJS

## środowisko i technologia ServerSide

---

PAWEŁ ŁUKASZUK

# Basic rules

---



# About me


---

## Paweł Łukaszuk

backend developer, speaker, trainer

✉ [pawel@lukaszuk.net](mailto:pawel@lukaszuk.net)

 [linkedin.com/in/pawellukaszuk](https://linkedin.com/in/pawellukaszuk)

 [@Lukaszuk\\_Pawel](https://twitter.com/Lukaszuk_Pawel)

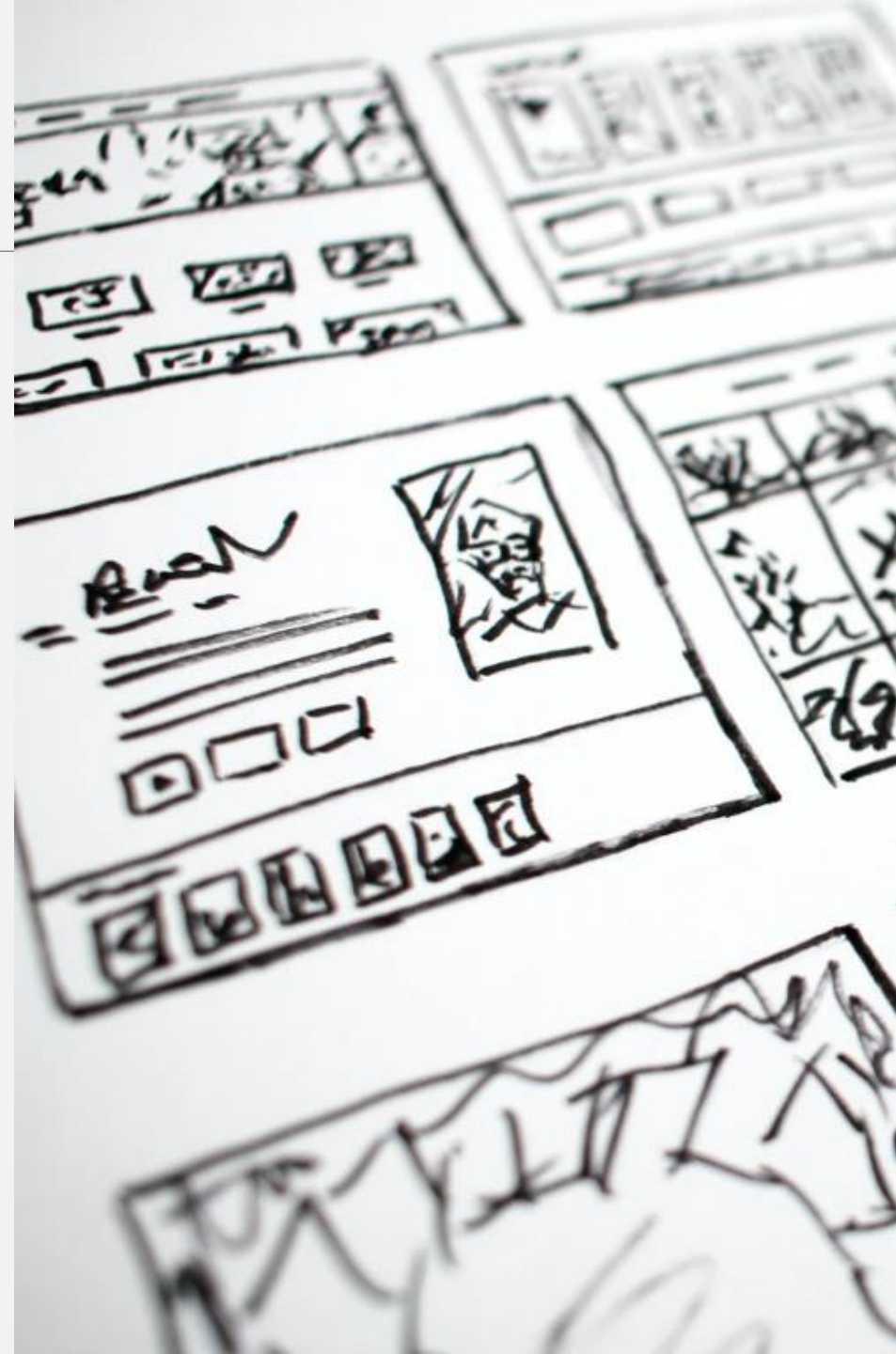
 [github.com/pawellukaszuk](https://github.com/pawellukaszuk)



# Plan

---

- Node.js environment - what it is and how it works
- Basics of programming in a Node.js environment
- Asynchronous programming
- Package and module - overview
- Creating your own package and module





# Library

---

Course repository: [github.com/pawellukaszuk/Nodejs20-21](https://github.com/pawellukaszuk/Nodejs20-21)

Node.js documentation: [nodejs.org/en/docs](https://nodejs.org/en/docs)

Node.js tutorial: [w3schools.com/nodejs](https://w3schools.com/nodejs)

NPM documentation: [docs.npmjs.com](https://docs.npmjs.com)

[github.com/kryz81/awesome-nodejs-learning](https://github.com/kryz81/awesome-nodejs-learning)

[github.com/sindresorhus/awesome-nodejs](https://github.com/sindresorhus/awesome-nodejs)

Mardan Azat, „Node.js w praktyce”, Helion 2015

David Herron: „Platforma Node.js Przewodnik webdevelopera” Helion 2017

Mike Cantelon, Marc Harter „Node.js w akcji”, Helion 2015

[nodeweekly.com](https://nodeweekly.com)



# Atwood's Law

---

“Any application that can be written in JavaScript, will eventually be written in JavaScript.”

Jeff Atwood

Cofounder of StackOverflow



# FrontEnd & BackEnd

---

Web browser

HTML

CSS

JavaScript

Angular

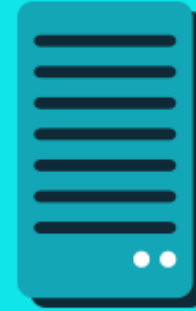
React

Vue

FRONT-END



BACK-END



Server

API

Database

Node.js

.NET

Java

PHP

# Node.js

---

As an asynchronous event-driven JavaScript runtime.

Node.js is designed to build scalable network applications.

Open-source cross-platform environment that executes JavaScript code outside a web browser.





# Why Node.js?

---

- multipurpose:
  - backend
  - desktop
  - tooling
- fast
- "lightweight" alternative
- vibrant ecosystem and platform support



# Where Node.js is mostly used?

---

- microservices and APIs
- serverless computing / function-as-a-service  
Azure Functions, AWS Lambda
- CLI applications
- desktop applications:  
electron framework (<https://www.electronjs.org/apps>)



# Where I can find Node.js?

---



# Why Node.js is for me?

---

- low entry cost - it's still JavaScript
- leverage your JavaScript skills
- huge library of code
- backend / fullstack / desktop developer  
one language everywhere
- create tools
- better understanding how web apps works  
by learning server side environment
- good for personal projects
- more convenient than PHP, cheaper than Java or .NET





# History

---

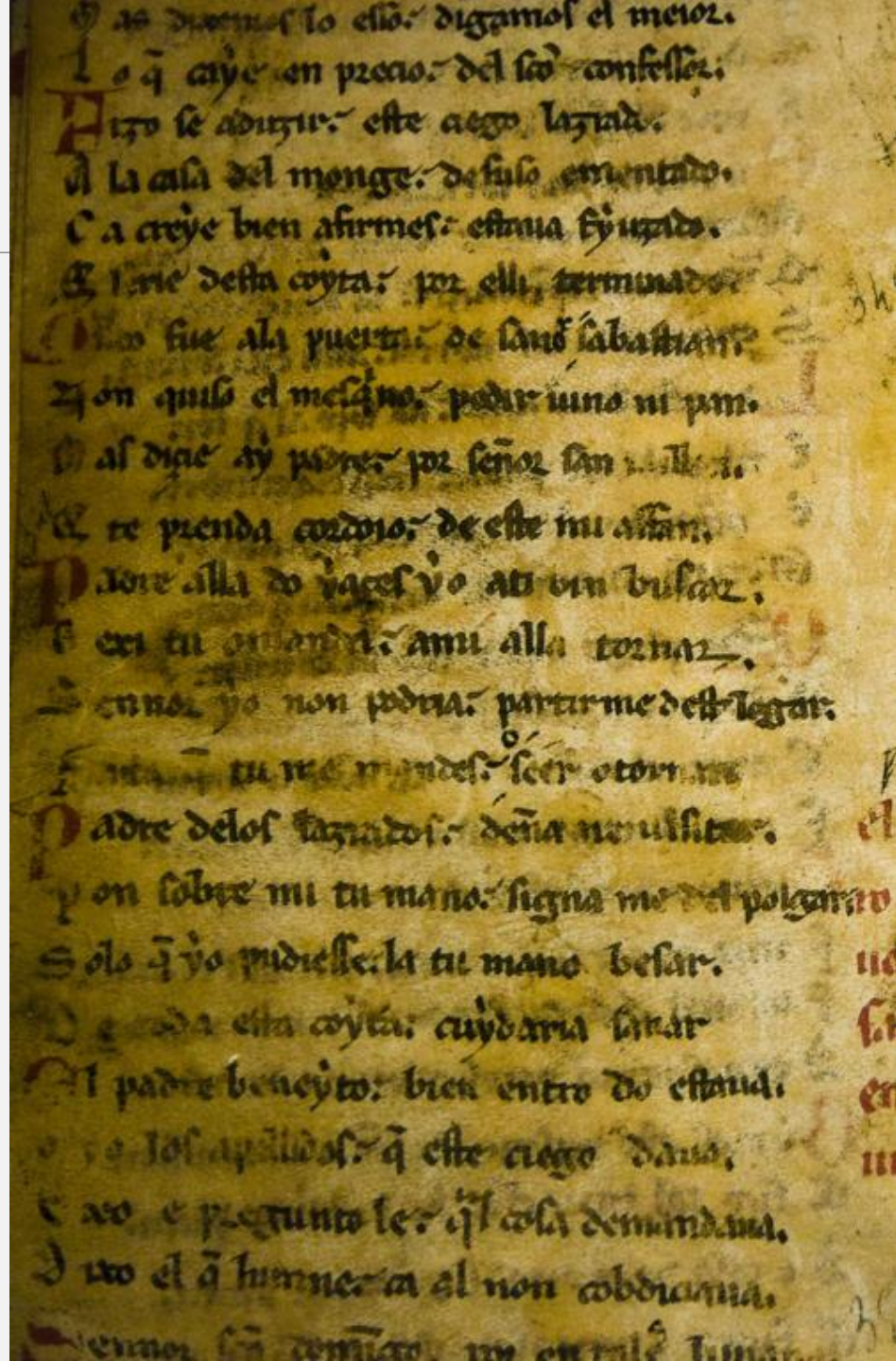
2009 Ryan Dahl demonstrates Node.js at jsconf.eu

2010 node package manager is released

2014 io.js forks Node.js

2015 io.js and Node.js merged, Node.js Foundation created

2019 JS Foundation and Node.js Foundation merged





# Open source

---

The OpenJS Foundation is comprised of many open source project communities which operate independently, but also collaborate together on the Cross Project Council (CPC).

Each project maintains their own communication channels, as does the OpenJS Foundation and the CPC.

<https://github.com/nodejs/node>



# Releasing strategy

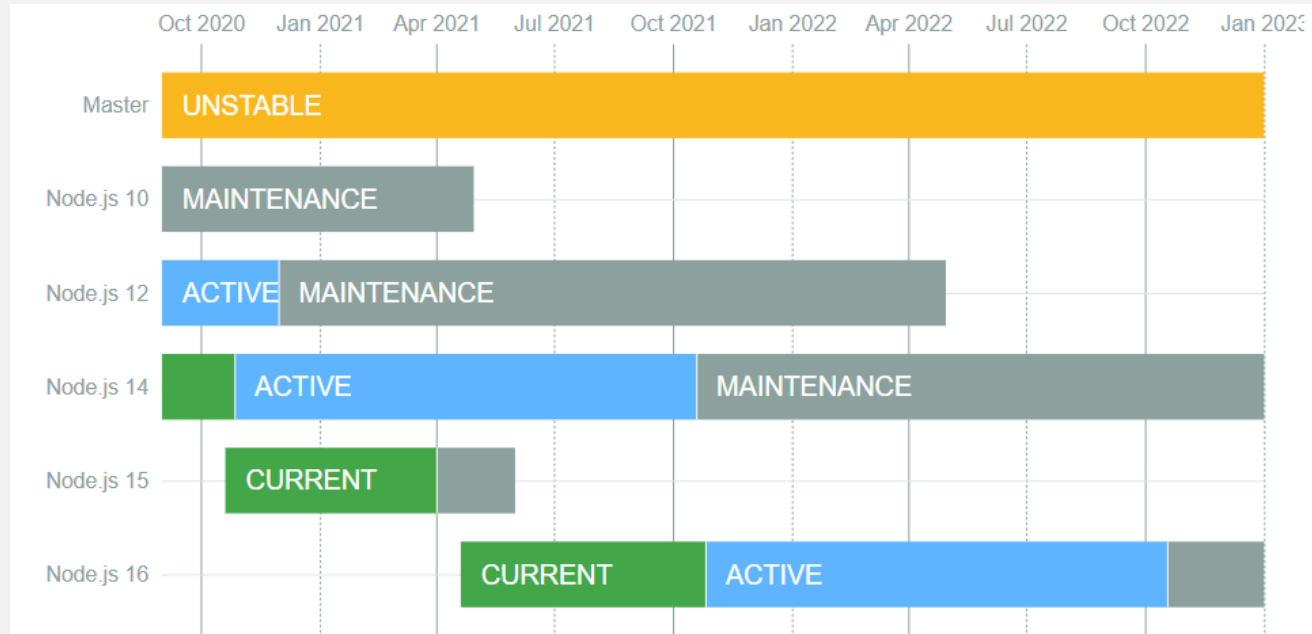
---

- new major version every 6 months
- even numbered releases covered by Long Time Support (LTS)
- no more than two active LTS releases at the time
- LTS version is recommended for production use

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



# Releases



Release	Status	Codename	Initial Release	Active LTS Start	Maintenance LTS Start	End-of-life
v10	Maintenance LTS	Dubnium	2018-04-24	2018-10-30	2020-05-19	2021-04-30
v12	Active LTS	Erbium	2019-04-23	2019-10-21	2020-11-30	2022-04-30
v14	Current		2020-04-21	2020-10-27	2021-10-19	2023-04-30
v15	Current		2020-10-20		2021-04-01	2021-06-01
v16	Pending		2021-04-20	2021-10-26	2022-10-18	2024-04-30

Dates are subject to change.

# JavaScript environment

---



# JavaScript runtime engine

---

- computer program that executes JavaScript code
- translates JavaScript code into more efficient machine code
- JavaScript engines are typically developed by web browser vendors





# Machine code

It is a set of processor commands in which a record of the program is expressed in the form of commands and their arguments.

The machine code is a form of a computer program (called an executable or binary form) intended for direct or almost direct execution by the processor.

```
1 sum: addc r30, r30, @FSaf
2      addc SAQ, r30, -56
3      addc SAQ, r30, -52
4      addc SAQ, r30, -48
5      addc ASAQ, r30, -44
6      mvut ASDQ, r5
7      addc ASAQ, r30, -40
8      mvut ASDQ, r4
9      addc ASAQ, r30, -36
10     mvut ASDQ, r3
11     mvut r2, CPQ
12     addc ALAQ, r30, b-@FSaf
13     mvfq r3, ALDQ
14     mvut r4, r2
15     br @L5
16 @L3: addc r4, r4, 1
17 @L5: cmpw r23, r3, r4, LT
18     brxnz r23, @L4
19     br @L3
20 @L4: addc LAQ, r30, -56
21     addc LAQ, r30, -52
22     addc LAQ, r30, -48
23     addc ALAQ, r30, -44
24     addc ALAQ, r30, -40
25     addc ALAQ, r30, -36
26     mvfq r5, ALDQ
27     mvfq r4, ALDQ
28     mvfq r3, ALDQ
29     subc r30, r30, @FSaf
30     ret

31 main: addc r30, r30, @FScm
32      addc SAQ, r30, 0-16
33      call sum
34      addc SAQ, 0, sol
35      subc r30, r30, @FScm
36      halt
```

# V8 engine

---

- Google's open source high-performance JavaScript
- used in Chrome and in Node.js, among others
- implements ECMAScript and WebAssembly, and runs on Windows 7 or later, macOS 10.12+, and Linux systems that use x64, IA-32, ARM, or MIPS processors
- can run standalone, or can be embedded into any C++ application



# What does it mean for me?

- no more browser incompatibilities or polyfills
- compatibility chart: <https://node.green>

Node.js ES2015 Support		Nightly!					
Learn more		16.0.0	15.0.1	14.5.0	13.14.0	13.1.0	12.10.0
		99% complete	99% complete	99% complete	99% complete	99% complete	99% complete
optimisation							
proper tail calls (tail call optimisation)							
direct recursion	?	Error	Error	Error	Error	Error	Error
mutual recursion	?	Error	Error	Error	Error	Error	Error
syntax							
default function parameters							
basic functionality	?	Yes	Yes	Yes	Yes	Yes	Yes
explicit undefined defers to the default	?	Yes	Yes	Yes	Yes	Yes	Yes
defaults can refer to previous params	?	Yes	Yes	Yes	Yes	Yes	Yes
arguments object interaction	?	Yes	Yes	Yes	Yes	Yes	Yes
temporal dead zone	?	Yes	Yes	Yes	Yes	Yes	Yes
separate scope	?	Yes	Yes	Yes	Yes	Yes	Yes
new Function() support	?	Yes	Yes	Yes	Yes	Yes	Yes
rest parameters							
basic functionality	?	Yes	Yes	Yes	Yes	Yes	Yes
function 'length' property	?	Yes	Yes	Yes	Yes	Yes	Yes
arguments object interaction	?	Yes	Yes	Yes	Yes	Yes	Yes
can't be used in setters	?	Yes	Yes	Yes	Yes	Yes	Yes
new Function() support	?	Yes	Yes	Yes	Yes	Yes	Yes

# What Node.js can do (today)

---

- C++ addons
- work with buffers
- create of child processes
- work with command line options
- use debugging console
- provide cryptographic functionality for OpenSSL's hash, HMAC etc.
- manipulate the filesystem
- handle input/output
- create our own http/https webserver
- creating stream-based TCP or IPC servers
- connect to databases
- use system-related utility methods and properties
- provide an API for implementing the stream interface
- scheduling functions to be called at some future period of time
- ...and many more

# Node.js vs javascript in browser

---

Node.js ↔ javascript in browser

global ↔ window

process ↔ document





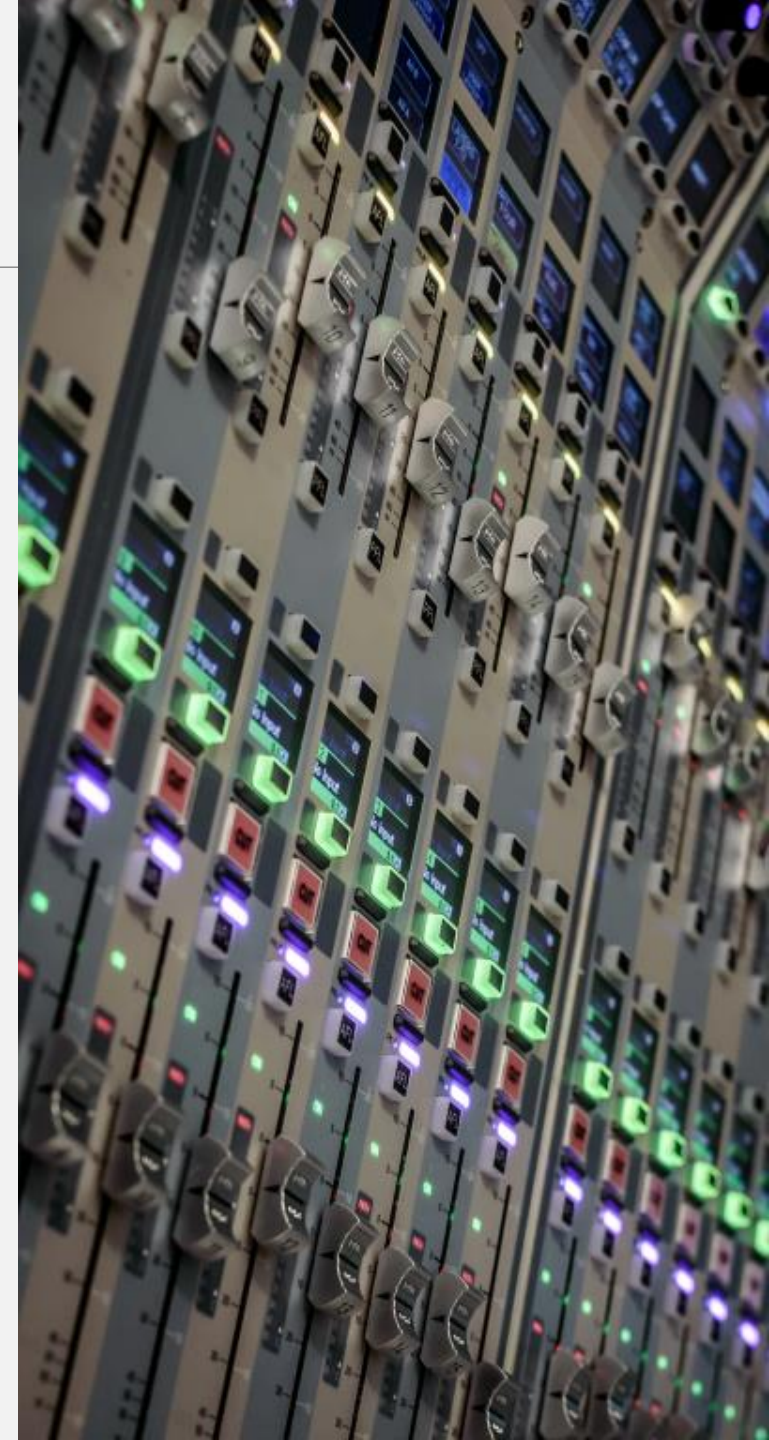
# Input/Output

---

It is something that every application does all the time – interaction with external entities.

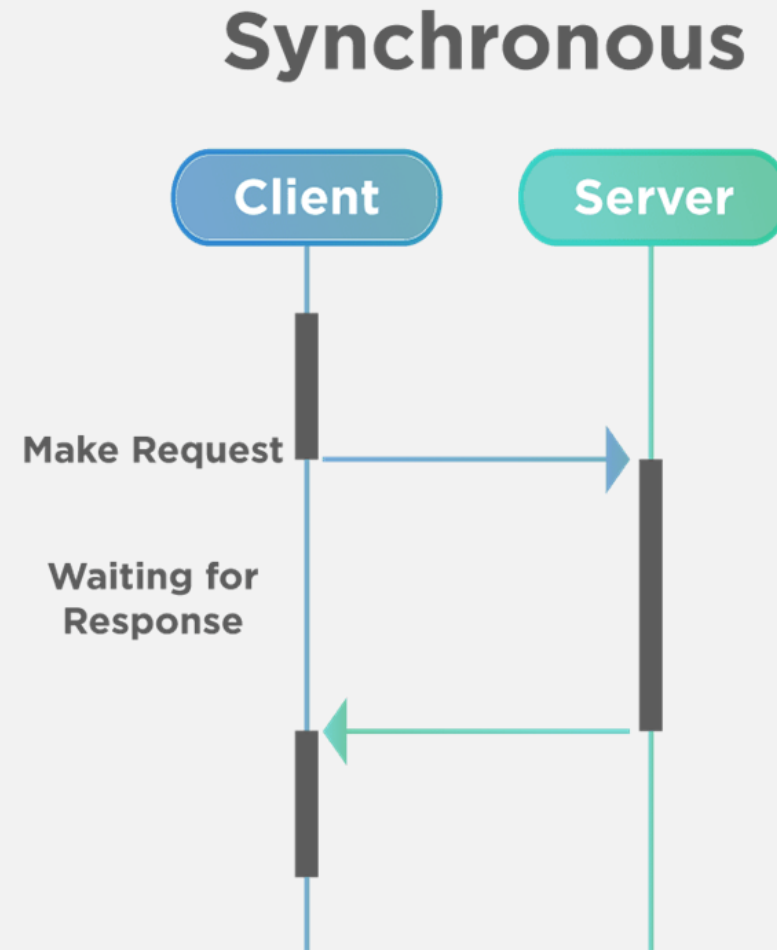
For example: database operations, sending and receiving HTTP requests, file system operations.

I/O operations take time!



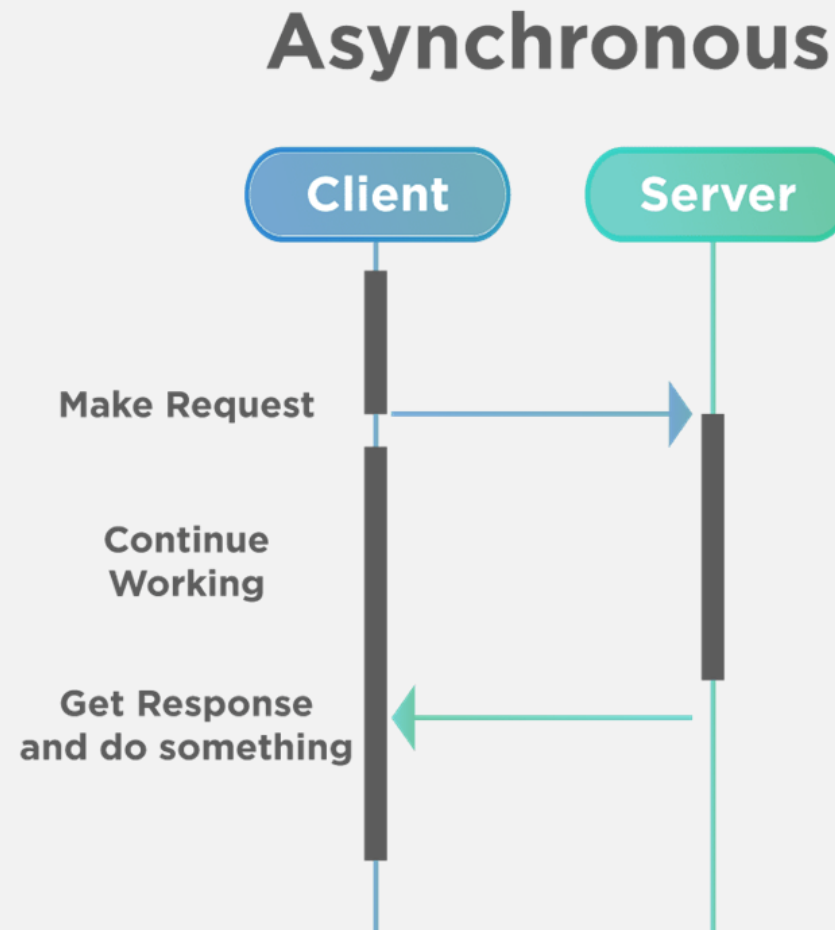
# Synchronous operations

---



# Asynchronous operations

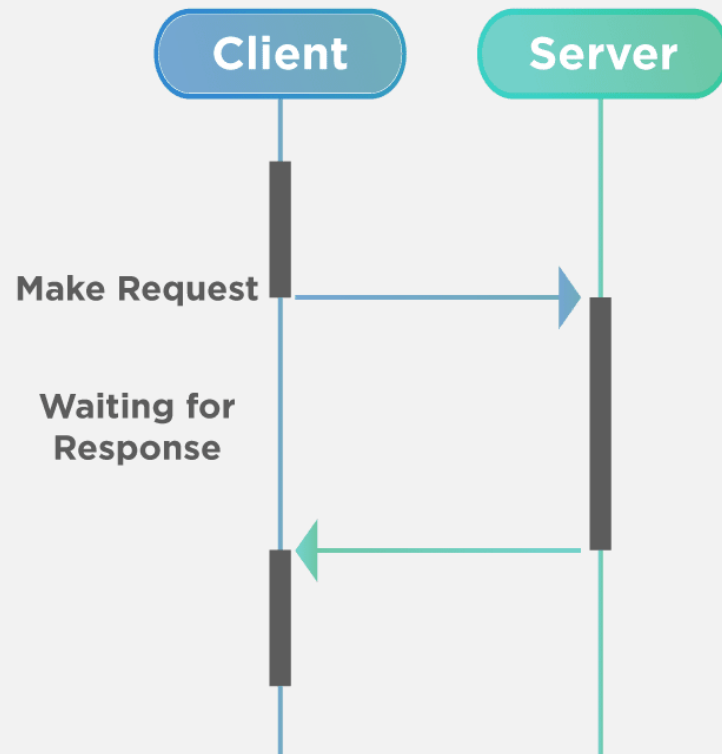
---



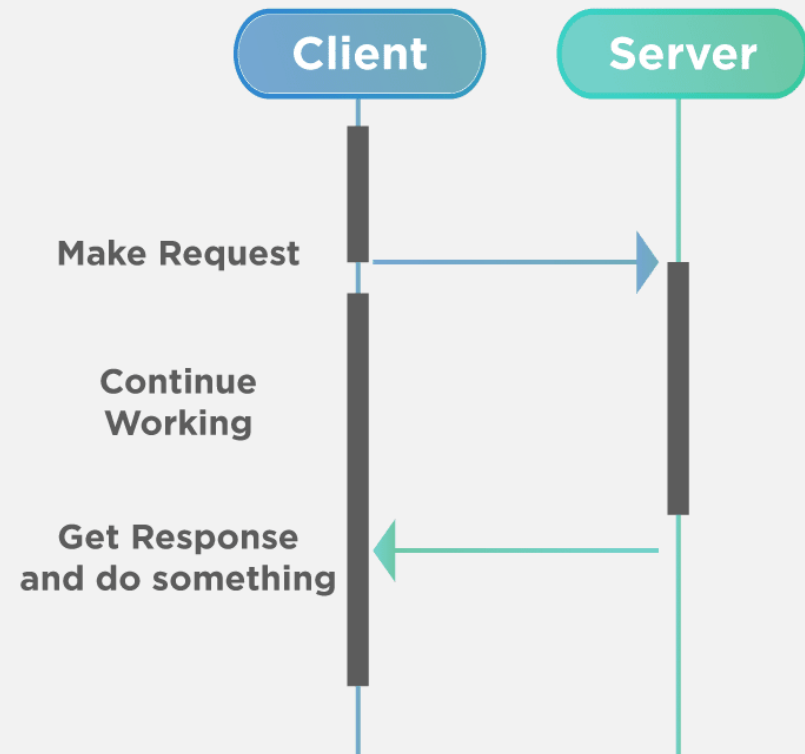
# Sync and Async operations

---

## Synchronous



## Asynchronous

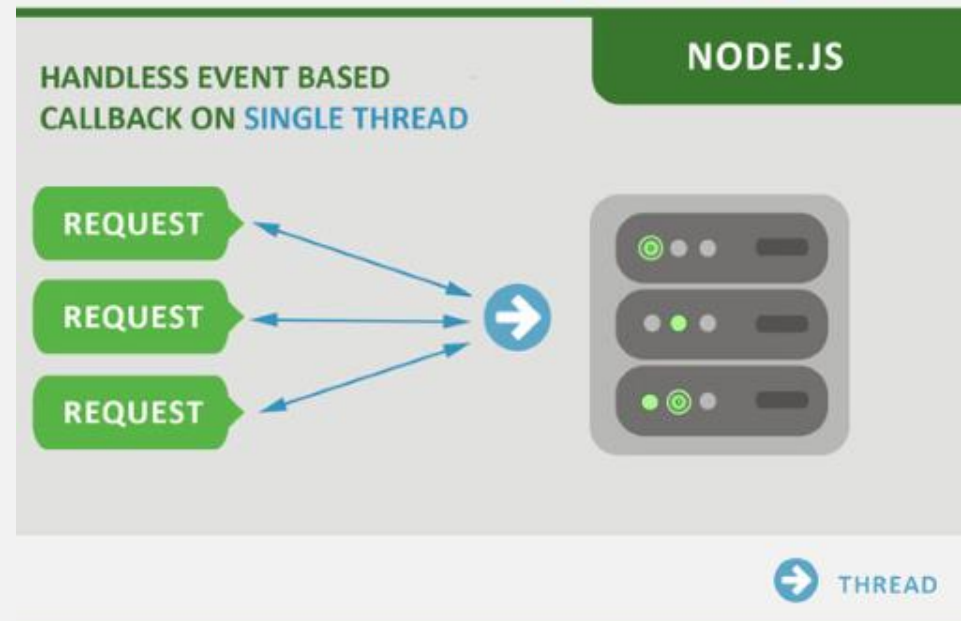
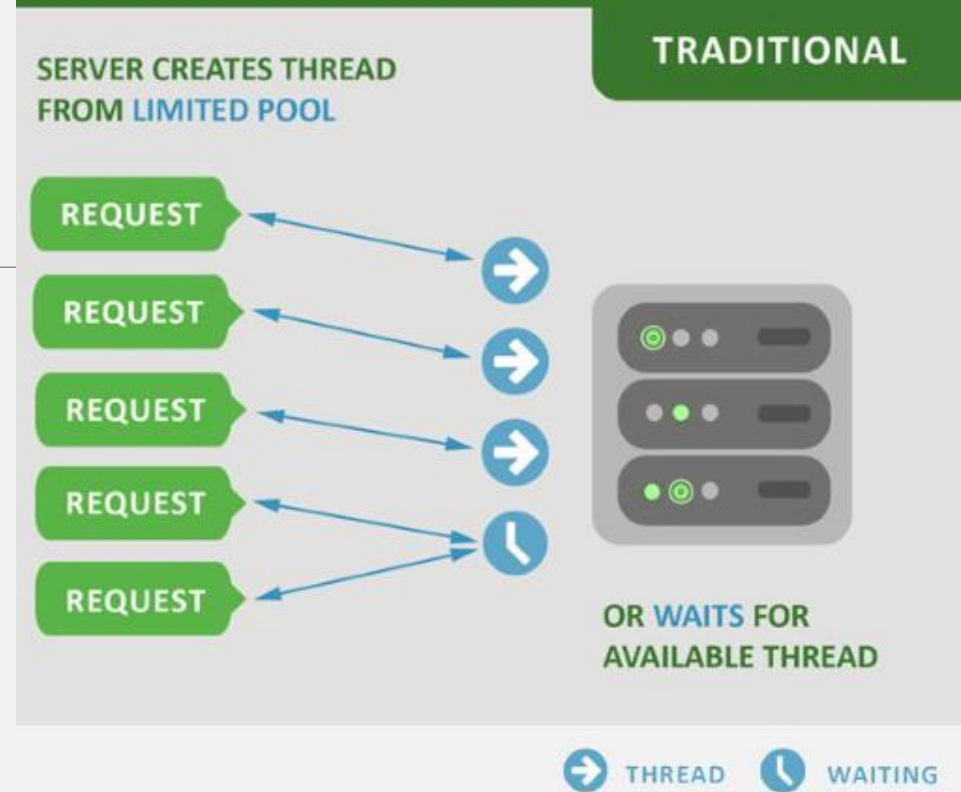


# Single/Multi threading

Multithreaded processes allow the execution of multiple parts of a program at the same time. These are lightweight processes available within the process.

Single threaded processes contain the execution of instructions in a single sequence. In other words, one command is processes at a time.

Node.js is singlethreaded.



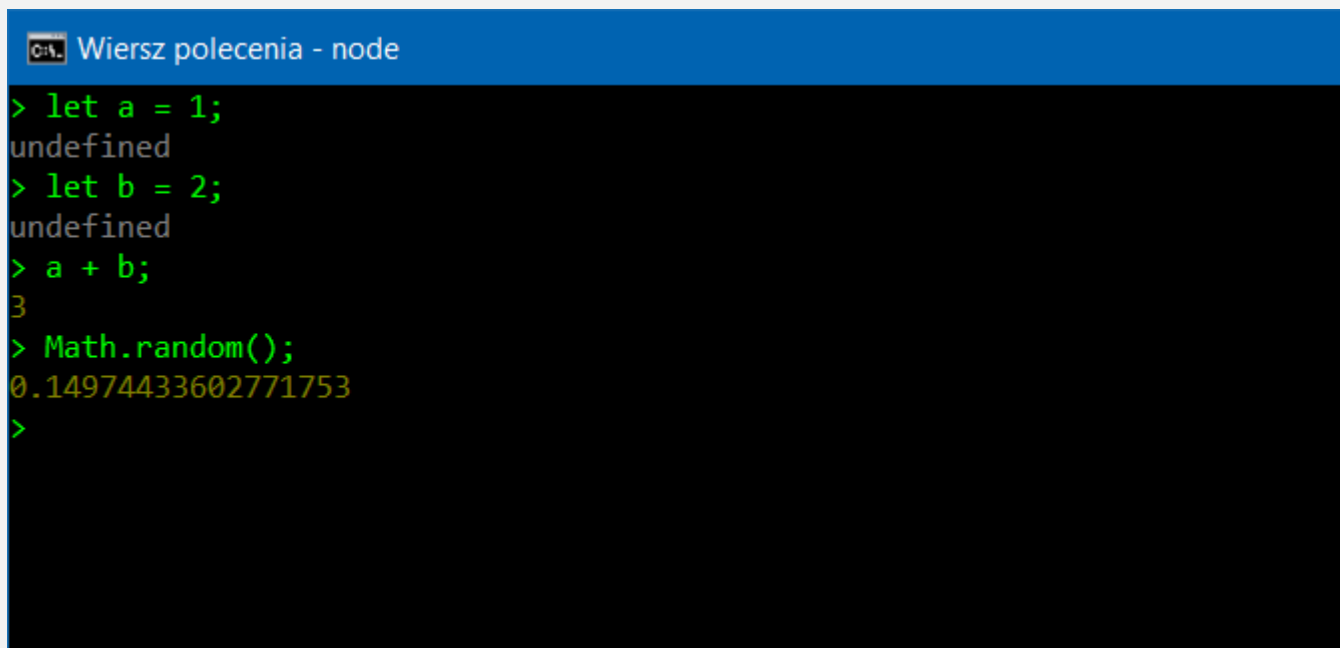


# CLI: REPL

---

Command-line interface processes commands to a computer program in the form of lines of text.

REPL: Read → Eval → Print → Loop

A screenshot of a Node.js REPL terminal window. The title bar is blue and contains the text "Wiersz polecenia - node". The terminal background is black with green text for input and yellow text for output. The input sequence is: `> let a = 1;`, `> let b = 2;`, `> a + b;`, and `> Math.random();`. The output sequence is: `undefined`, `undefined`, `3`, and `0.14974433602771753`.

```
Wiersz polecenia - node
> let a = 1;
undefined
> let b = 2;
undefined
> a + b;
3
> Math.random();
0.14974433602771753
>
```

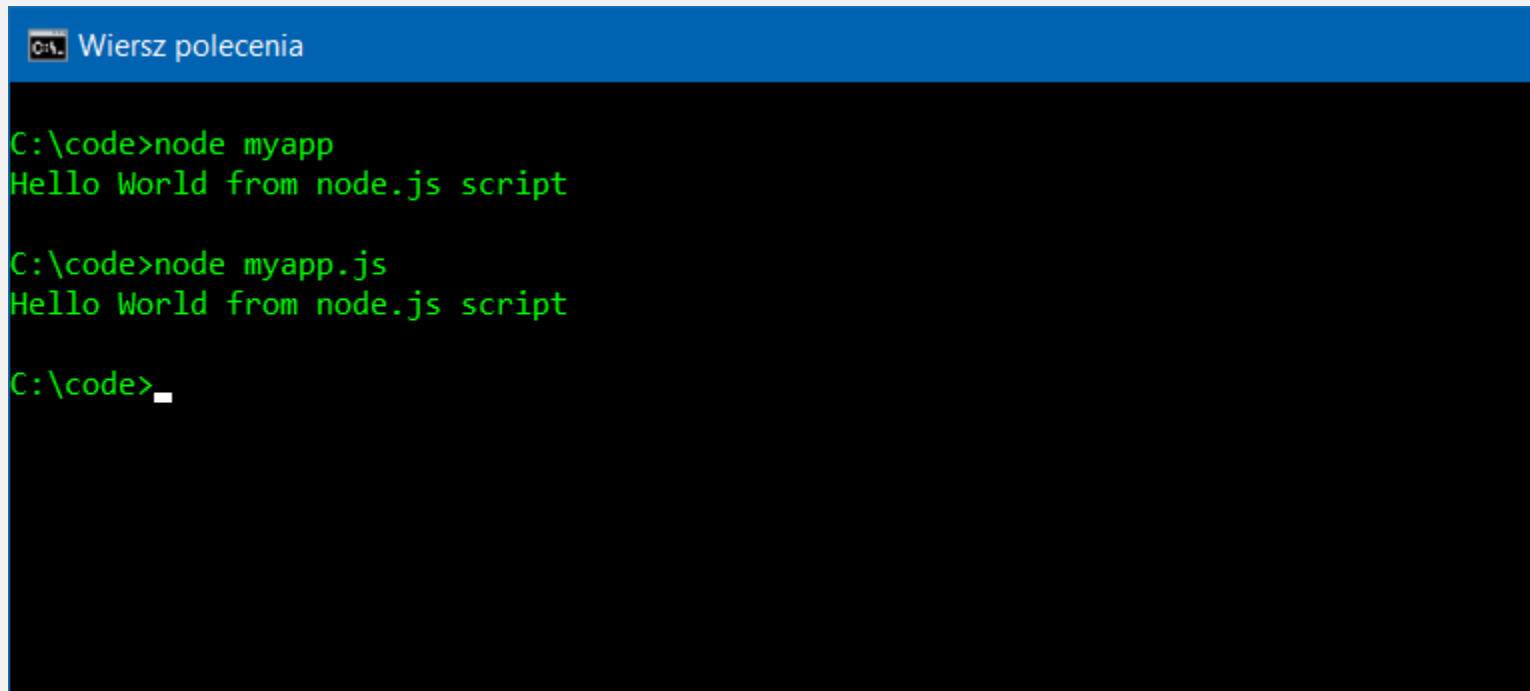
# CLI: script files

---

> node filename

or

> node filename.js



```
C:\code>node myapp
Hello World from node.js script

C:\code>node myapp.js
Hello World from node.js script

C:\code>_
```

# Module

---

Module in Node.js is a simple or complex functionality organized in single or multiple JavaScript files which can be reused throughout the Node.js application.

Each module in Node.js has its own context, so it cannot interfere with other modules or pollute global scope. Also, each module can be placed in a separate .js file under a separate folder.

Node.js implements CommonJS modules standard.

Node.js includes three types of modules:

- Core Modules
- Local Modules
- Third Party Modules



# Require

---

Function built into Node.js, allows us to read the module.

```
// core module
```

```
const fs = require('fs'); // absolute path
```

```
// local module
```

```
const user = require('./user'); // relative path
```

```
// third party module
```

```
const _ = require('lodash'); // absolute path
```

# Core Modules

---

Core modules are compiled into its binary distribution and load automatically when Node.js process starts.

You still need to import the core module before using it in your application!

```
console.log('start app');  
const os = require('os');  
const username = os.userInfo().username;  
console.log(username);  
const fs = require('fs');  
fs.writeFileSync('abc.txt', username);
```

<https://nodejs.org/dist/latest-v12.x/docs/api/>

# Local Modules

---

Local modules are modules created locally in your Node.js application.

These modules include different functionalities of your application in separate files and folders.

```
// file: app.js
console.log('start app');
const user = require('./user');
console.log(user.firstName);
console.log(user.lastName);
```

```
// file: user.js
module.exports = {
  firstName: 'Jan',
  lastName: 'Nowak',
};
module.exports.city = 'Bialystok';
```



# Third-party Modules

---

Third party modules can be downloaded using Node Package Manager (NPM).

Can be installed inside the project folder or globally.

```
// npm init
// npm install lodash
console.log('start app');
const _ = require('lodash');
console.log(_.isString('Ala ma kota'));
console.log(_.isString(123));
```

# Module.exports

---

Allows you to export the variable/function.

```
module.exports = 'Ala ma kota';
```

```
// or
```

```
module.exports.add = (a, b) => a + b;
```

```
module.exports.sub = (a, b) => a - b;
```

```
// or
```

```
module.exports = {
```

```
  add: (a, b) => a + b,
```

```
  sub: (a, b) => a - b,
```

```
};
```

```
// or
```

```
exports.add = (a, b) => a + b;
```

```
exports.sub = (a, b) => a - b;
```

```
// but not this way!
```

```
exports = {
```

```
  add: (a, b) => a + b,
```

```
  sub: (a, b) => a - b,
```

```
};
```

```
// and not this way!
```

```
exports = 'Ala ma kota';
```

# NPM – node package manager

---

NPM is a command-line application that helps you install modules available in the repository.

Node package manager, the default package manager for the NodeJS environment.

<https://www.npmjs.com>



# Tools

---

Node.js (LTS version)

[nodejs.org](https://nodejs.org)

Git

[git-scm.com](https://git-scm.com)

text editor

Visual Studio Code

[code.visualstudio.com](https://code.visualstudio.com)

console

file manager

Double Commander

[doublecmd.sourceforge.io](https://doublecmd.sourceforge.io)

