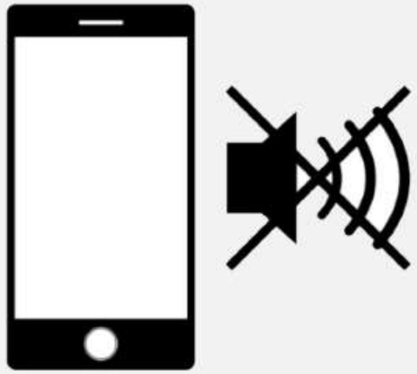
NodeJS
środowisko i technologia ServerSide

PAWEŁ ŁUKASZUK

Basic rules



About me

Paweł Łukaszuk

backend developer, speaker, trainer

pawel@lukaszuk.net

linkedin.com/in/pawellukaszuk

@Lukaszuk_Pawel

github.com/pawellukaszuk

Plan

- Node.js environment - what it is and how it works
- Basics of programming in a Node.js environment
- Modules and packages
- Asynchronous programming using different techniques
- Web protocols
- API types
- MongoDB database
- Testing
- My own WebApp

Library

Course repository:	github.com/pawellukaszuk/Nodejs21-22
Node.js documentation:	nodejs.org/en/docs
Node.js tutorial:	w3schools.com/nodejs
NPM documentation:	docs.npmjs.com

github.com/kryz81/awesome-nodejs-learning

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

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

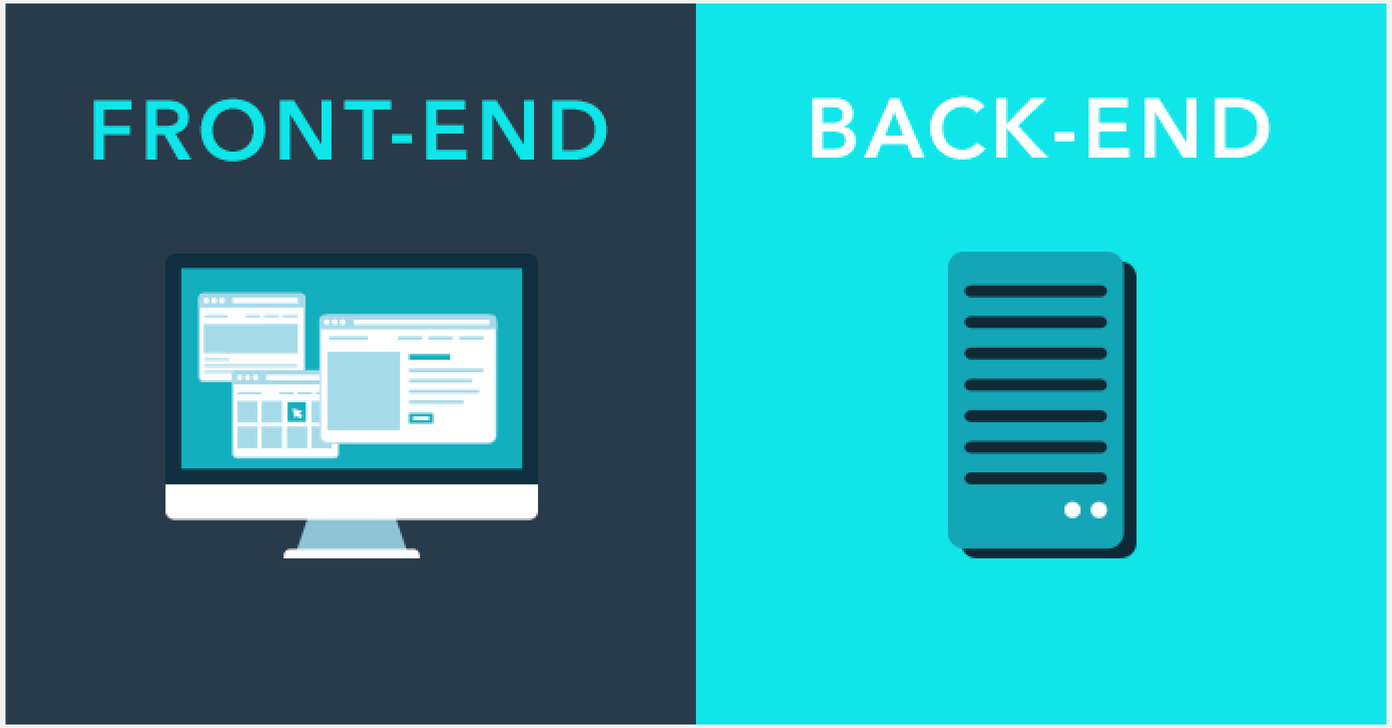
CSS

JavaScript

Angular

React

Vue



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

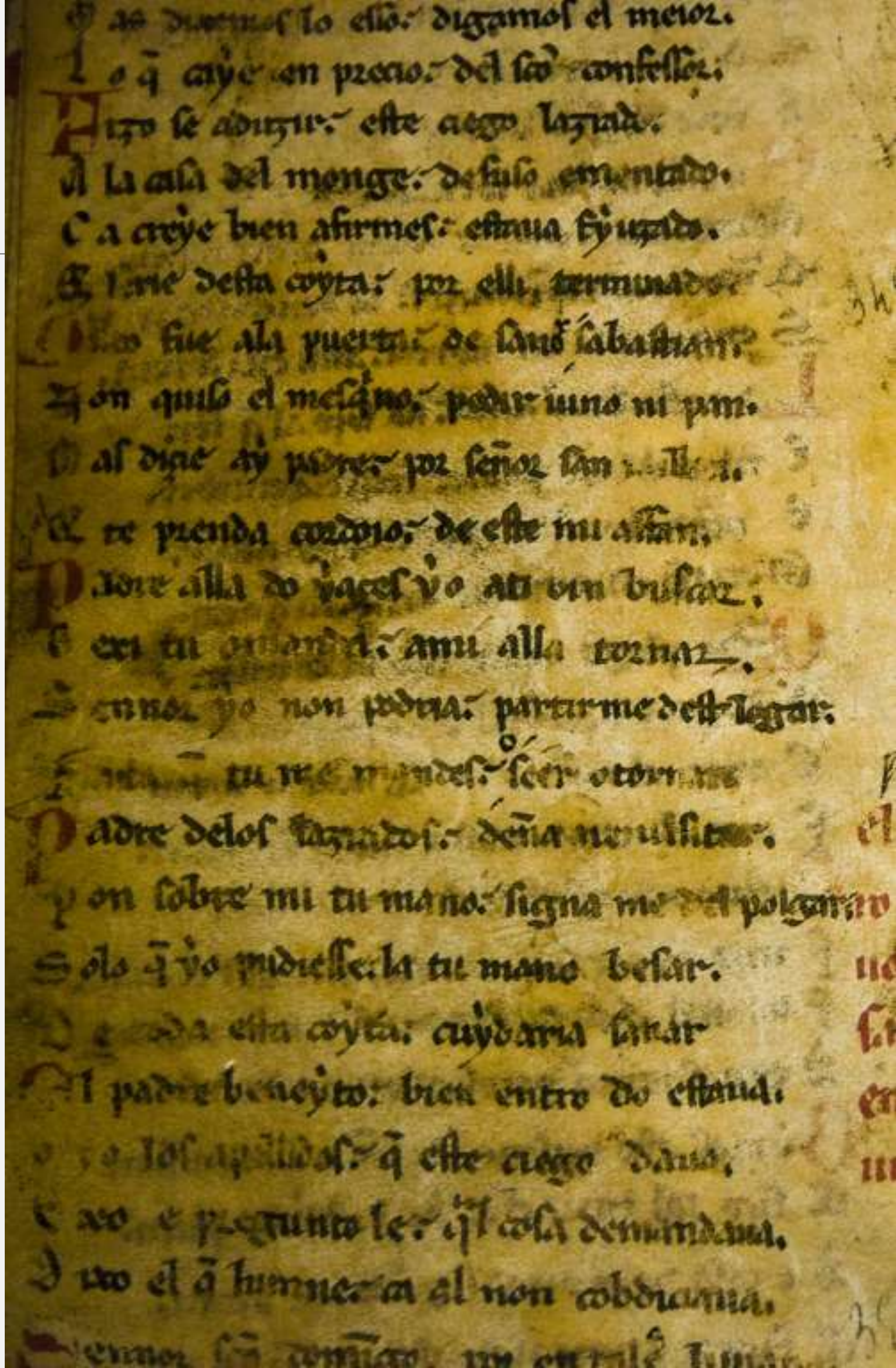
2009 Ryan Dahl demonstrates Node.js at jsconf.eu

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

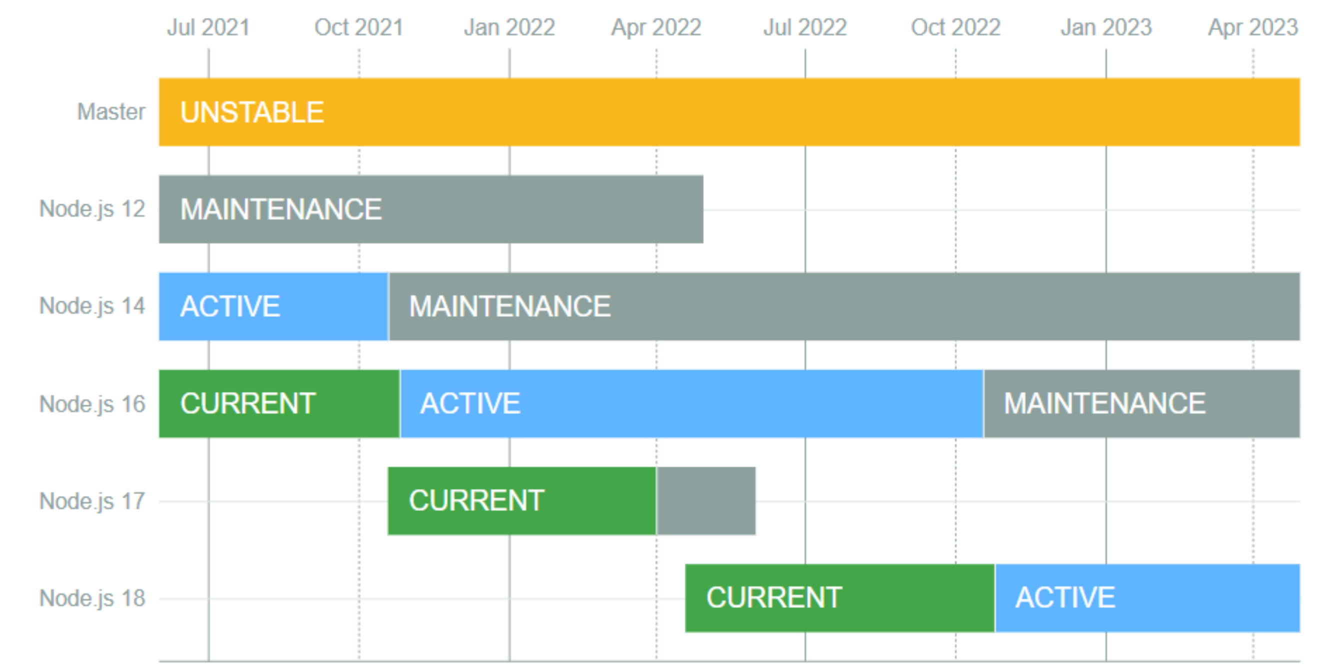
OpenJS Foundation

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
v12	Maintenance LTS	Erbium	2019-04-23	2019-10-21	2020-11-30	2022-04-30
v14	Active LTS	Fermium	2020-04-21	2020-10-27	2021-10-19	2023-04-30
v16	Current		2021-04-20	2021-10-26	2022-10-18	2024-04-30
v17	Pending		2021-10-19		2022-04-01	2022-06-01
v18	Pending		2022-04-19	2022-10-25	2023-10-18	2025-04-30

JavaScript environment

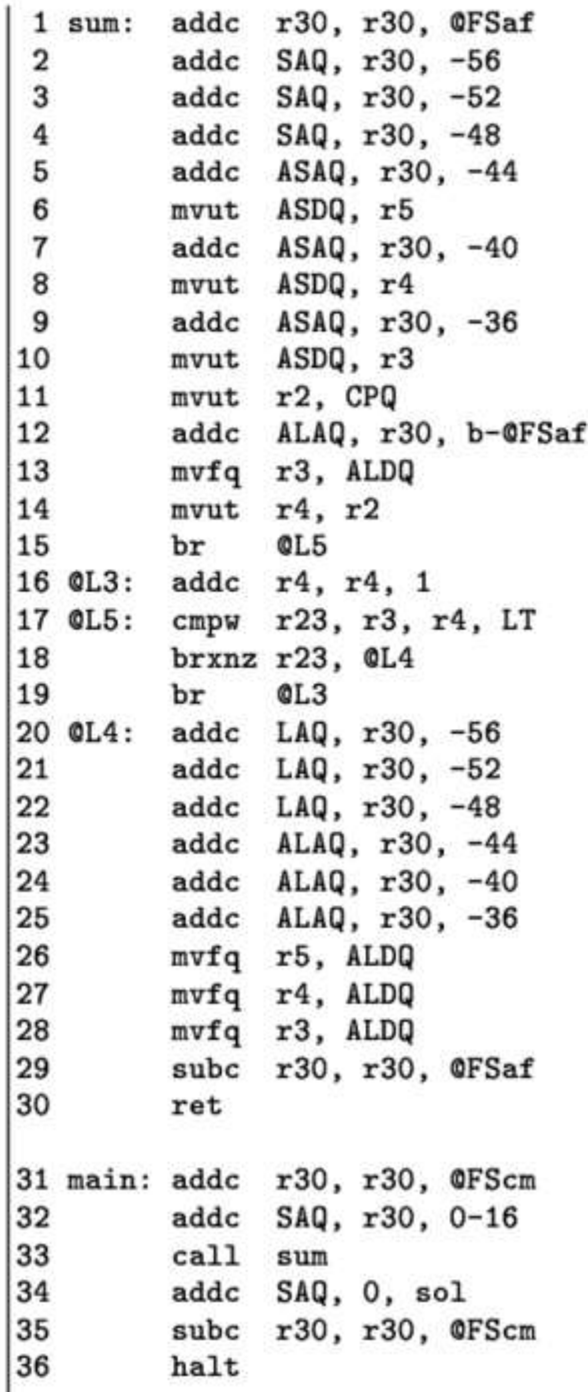
JavaScript runtime engine

- 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.



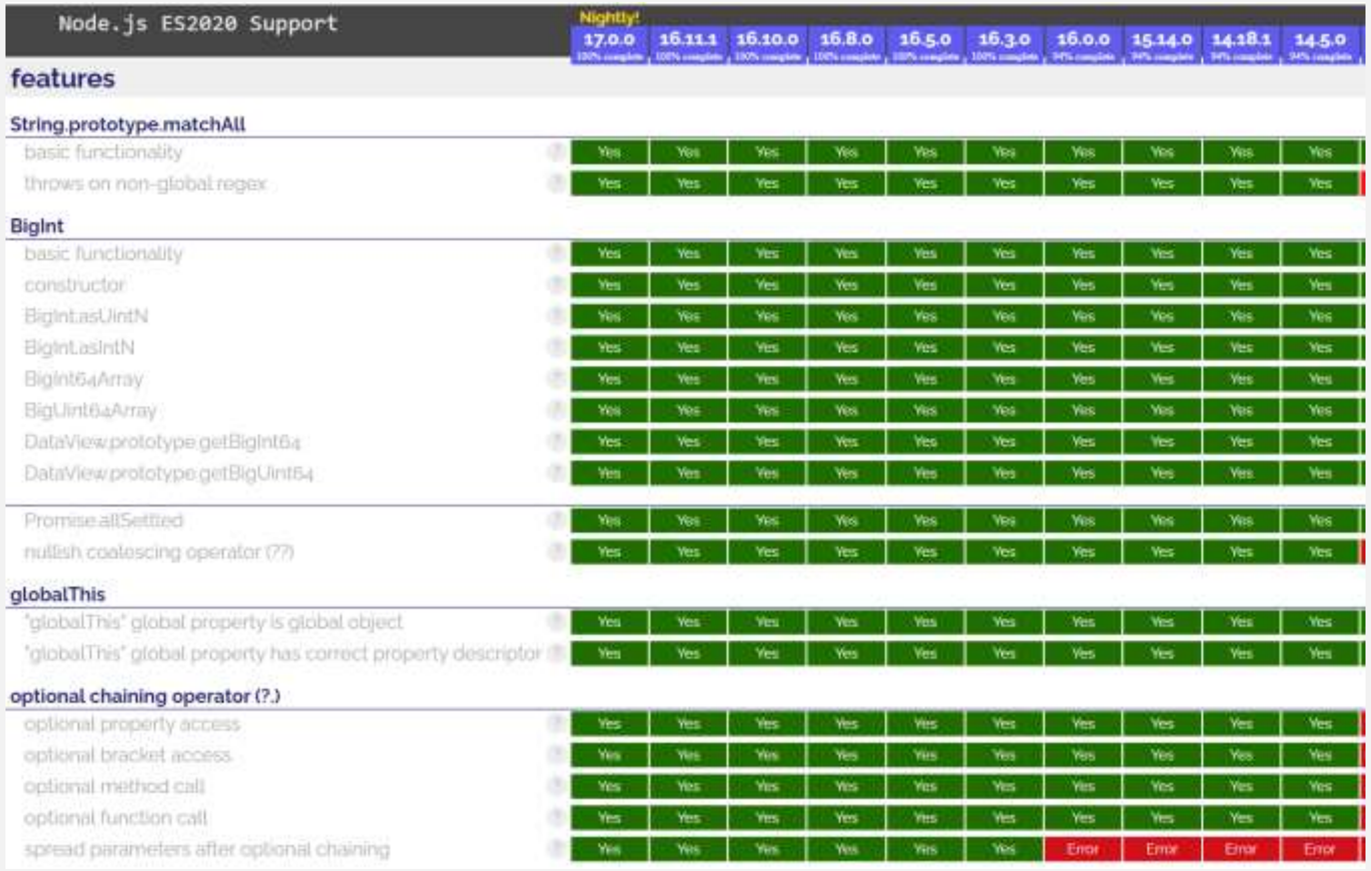
```
sum:    addc   r30, r30, @FSaf
        addc   SAQ, r30, -56
        addc   SAQ, r30, -52
        addc   SAQ, r30, -48
        addc   ASAQ, r30, -44
        mvut   ASDQ, r5
        addc   ASAQ, r30, -40
        mvut   ASDQ, r4
        addc   ASAQ, r30, -36
        mvut   ASDQ, r3
        mvut   r2, CPQ
        addc   ALAQ, r30, b-@FSaf
        mvfq   r3, ALDQ
        mvut   r4, r2
        br     @L5
@L3:    addc   r4, r4, 1
@L5:    cmpw   r23, r3, r4, LT
        brxnz  r23, @L4
        br     @L3
@L4:    addc   LAQ, r30, -56
        addc   LAQ, r30, -52
        addc   LAQ, r30, -48
        addc   ALAQ, r30, -44
        addc   ALAQ, r30, -40
        addc   ALAQ, r30, -36
        mvfq   r5, ALDQ
        mvfq   r4, ALDQ
        mvfq   r3, ALDQ
        subc   r30, r30, @FSaf
        ret

main:   addc   r30, r30, @FScm
        addc   SAQ, r30, 0-16
        call   sum
        addc   SAQ, 0, sol
        subc   r30, r30, @FScm
        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 ES2020 Support

features	Nightly! 17.0.0 100% complete	16.11.1 100% complete	16.10.0 100% complete	16.8.0 100% complete	16.5.0 100% complete	16.3.0 100% complete	16.0.0 94% complete	15.14.0 94% complete	14.18.1 94% complete	14.5.0 94% complete
String.prototype.matchAll										
basic functionality	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
throws on non-global regex	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BigInt										
basic functionality	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
constructor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BigInt.asUintN	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BigInt.asIntN	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BigInt64Array	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
BigUint64Array	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DataView.prototype.getBigInt64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
DataView.prototype.getBigUint64	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Promise.allSettled	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
nullish coalescing operator (??)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
globalThis										
"globalThis" global property is global object	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
"globalThis" global property has correct property descriptor	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
optional chaining operator (?.)										
optional property access	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
optional bracket access	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
optional method call	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
optional function call	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
spread parameters after optional chaining	Yes	Yes	Yes	Yes	Yes	Yes	Error	Error	Error	Error

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

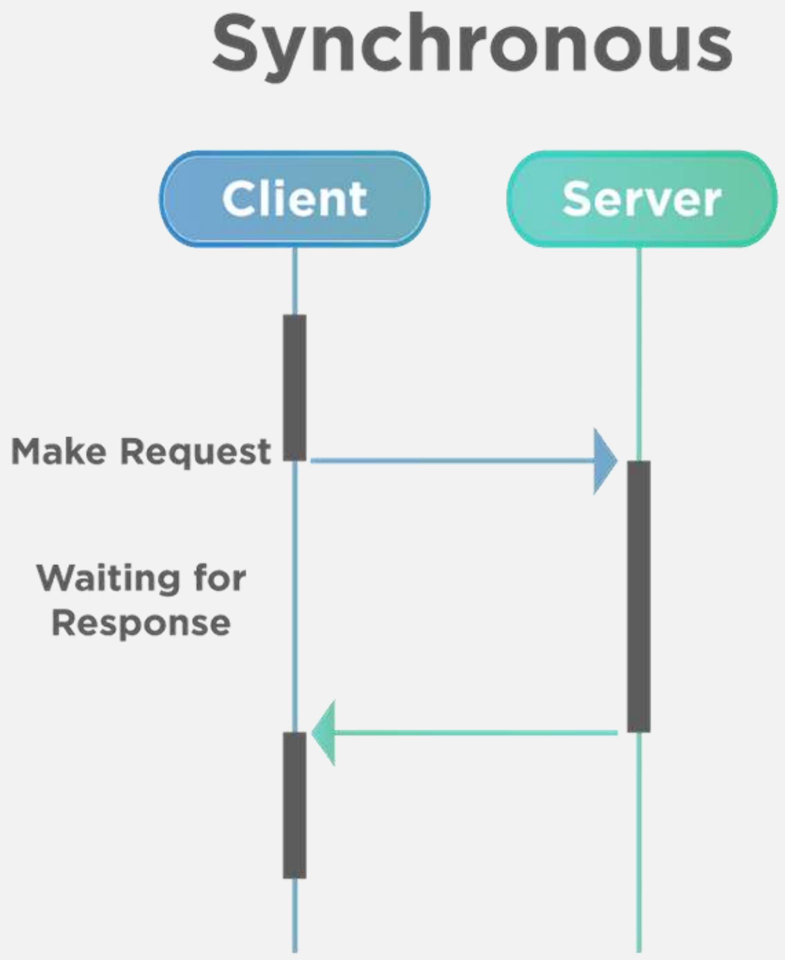
Input/Output

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

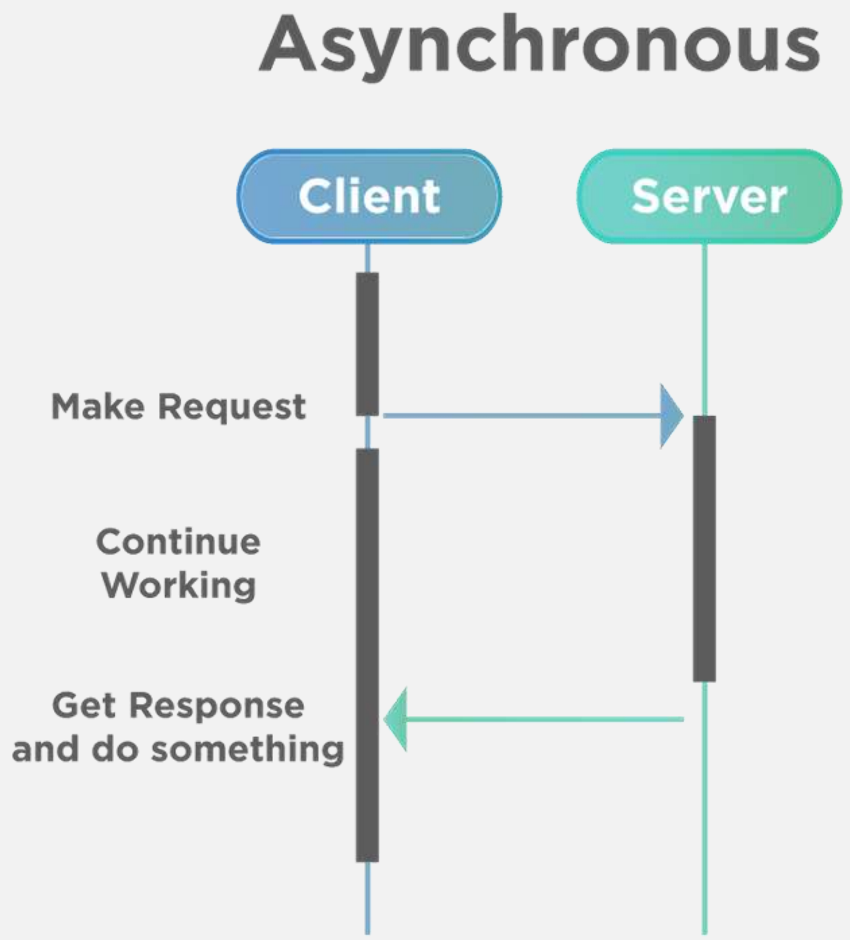
For example: database operations, sending and receiving HTTP requests, file system operations.

I/O operations take time!

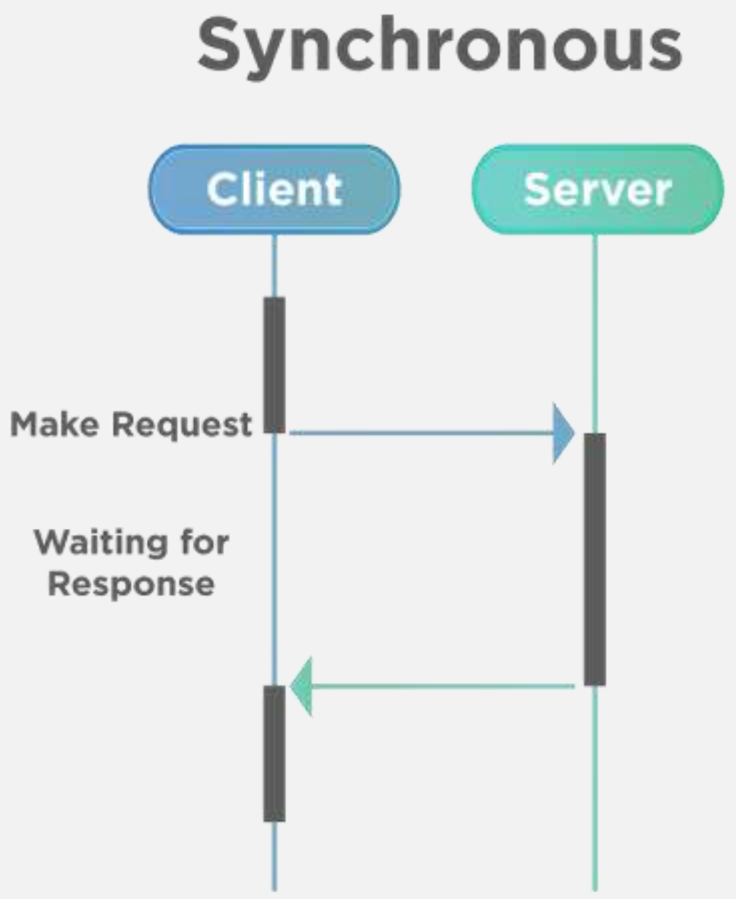
Synchronous operations

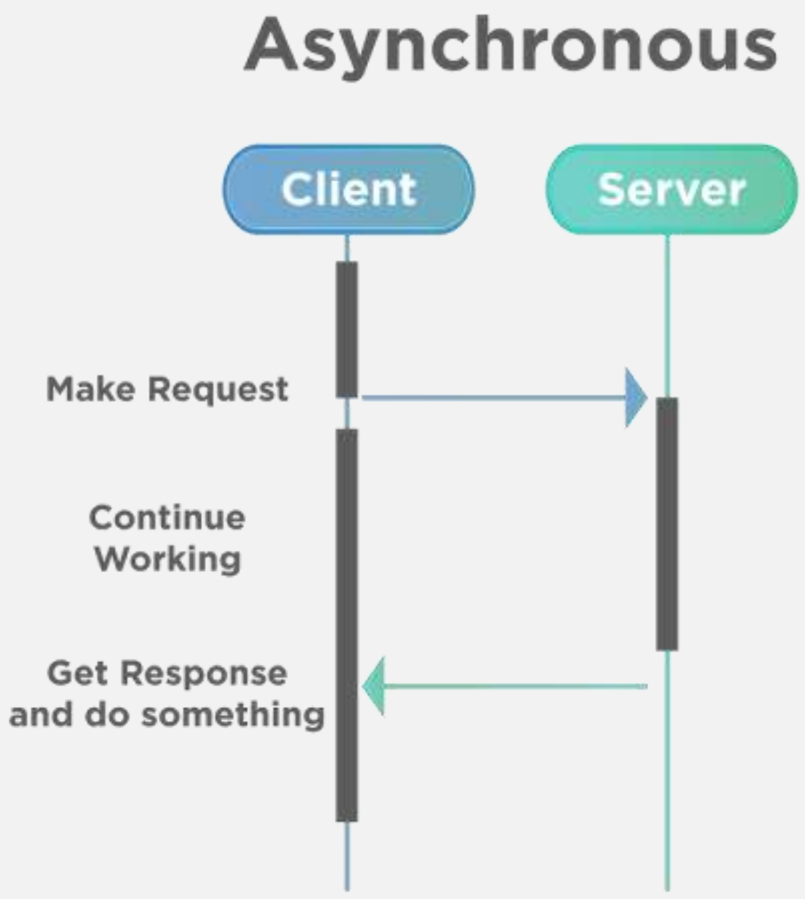


Asynchronous operations



Sync and Async operations



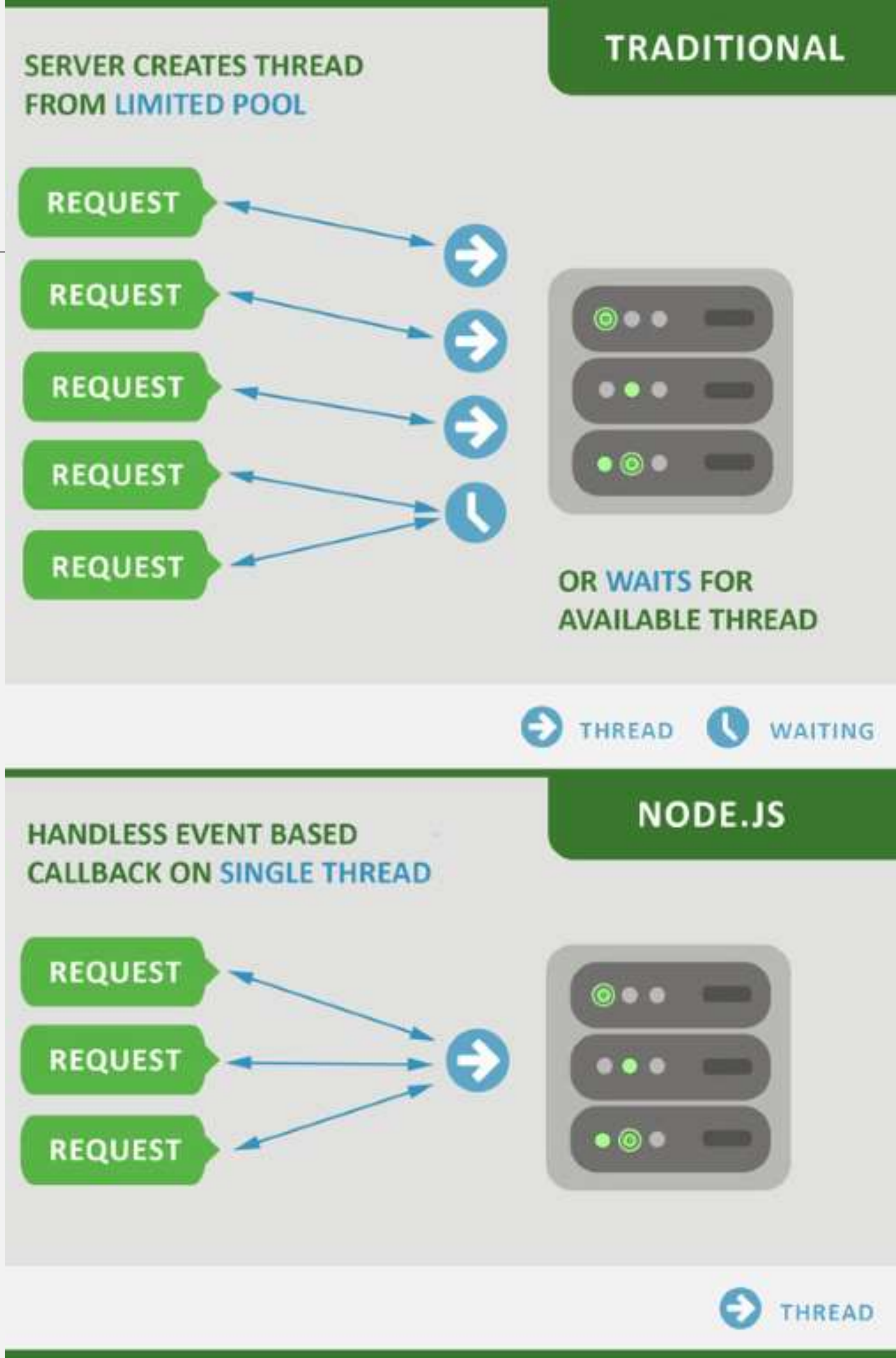


Single/Multi threading

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.

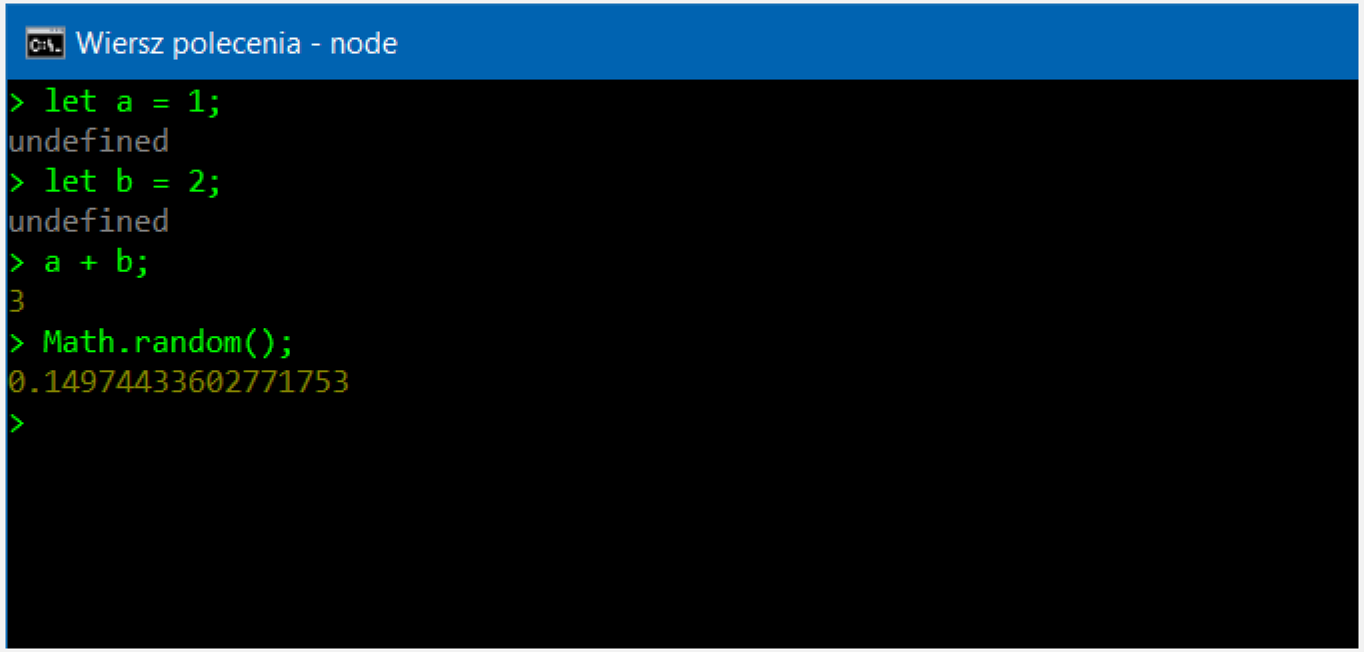
Node.js is singlethreaded.


TRADITIONAL
SERVER CREATES THREAD FROM LIMITED POOL
REQUEST
REQUEST
REQUEST
REQUEST
REQUEST
OR WAITS FOR AVAILABLE THREAD
THREAD
WAITING
NODE.JS
HANDLESS EVENT BASED CALLBACK ON SINGLE THREAD
REQUEST
REQUEST
REQUEST
THREAD

CLI: REPL

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

REPL: Read → Eval → Print → Loop



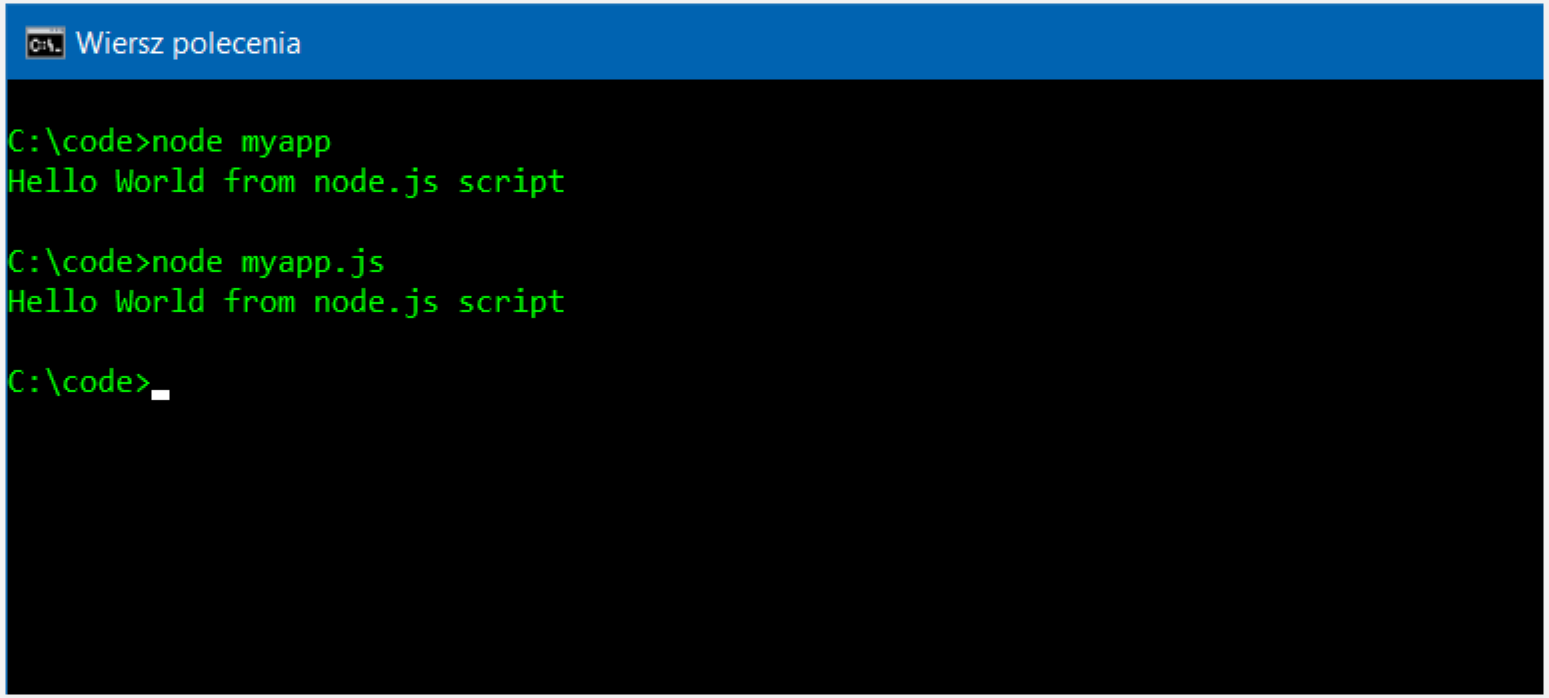
```
> let a = 1;
undefined
> let b = 2;
undefined
> a + b;
3
> Math.random();
0.1497443360277175З
>
```

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-v14.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	
Git	git-scm.com	
text editor	Visual Studio Code	code.visualstudio.com
console		
file manager	Double Commander	doublecmd.sourceforge.io