

Hands-on NodeJS

+ JavaScript

+ Debug (Chrome / IntelliJ)

+ TypeScript

arnaud.nauwynck@gmail.com

Esilv 2024

This document:

<https://github.com/Arnaud-Nauwynck/presentations/tree/main/web/tp1-nodejs.pdf>

Objectives

- Discover NodeJs using practical « Hello World » developper code
- Using Simple baby steps, step by step
- Setup your web environment
- Become familiar with the Web FrontEnd ecosystem

Outline

- Download + Install NodeJs
- Run NodeJS from the command line
- Discover and run basic javascript program
- Discover NPM Node Package Manager
- Install package.json dependencies
- Install WebStorm IDE
(students may choose any IDE, VisualStudio Code)
- Debug Javascript program
- TypeScript to JavaScript from the IDE
- Tsc compiler
- Debug TypeScript

Google NodeJs

← → C google.com/search?q=nodejs&rlz=1C1CHBF_frFR928FR928&oq=nodejs&aqs=chrome..69i57j69i59j0i20i263i512j0i512j0i10i512j69i60l3.1583j0j9&sourceid=chrome&ie=UTF-8

nodejs

X | 🔍 | 📸 | 🔎

Videos Images Tutorial Course Documentation Download W3schools Framework Jobs All filters ▾ Tools SafeSearch ▾

About 267,000,000 results (0.29 seconds)

 Node.js
<https://nodejs.org> › ...

Node.js

Node.js® is an open-source, cross-platform JavaScript runtime environment. Download for Linux (x64). 18.17.1 LTS Recommended For Most ...

You've visited this page many times. Last visit: 7/9/2023

Download

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript ...

Docs

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript ...

Téléchargements

Dernière version LTS: 18.17.1 (includes npm 9.6.7 ...)

About

As an asynchronous event-driven JavaScript runtime, Node.js is ...

 Node.js

node JS Hours Full Course

More images

Node.js

Platform

Node.js is a cross-platform, open-source server environment that can run on Windows, Linux, Unix,

Download NodeJs

nodejs.org/en/download/current

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | CERTIFICATION | NEWS

Downloads

Latest Current Version: 20.5.1 (includes npm 9.8.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS	Current
Recommended For Most Users	Latest Features
 Windows Installer node-v20.5.1-x64.msi	 macOS Installer node-v20.5.1.pkg
	 Source Code node-v20.5.1.tar.gz

Windows Installer (.msi)

Windows Binary (.zip)

macOS Installer (.pkg)

macOS Binary (.tar.gz)

Linux Binaries (x64)

Linux Binaries (ARM)

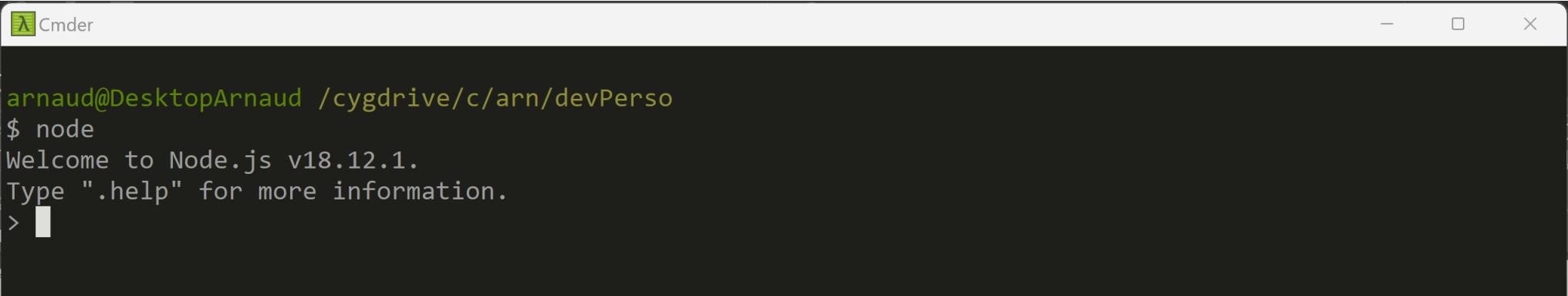
Source Code

32-bit	64-bit	ARM64
32-bit	64-bit	ARM64
64-bit / ARM64		
64-bit		ARM64
64-bit		
ARMv7		ARMv8
node-v20.5.1.tar.gz		

Additional Platforms

Start NodeJs interactive prompt

C:> node

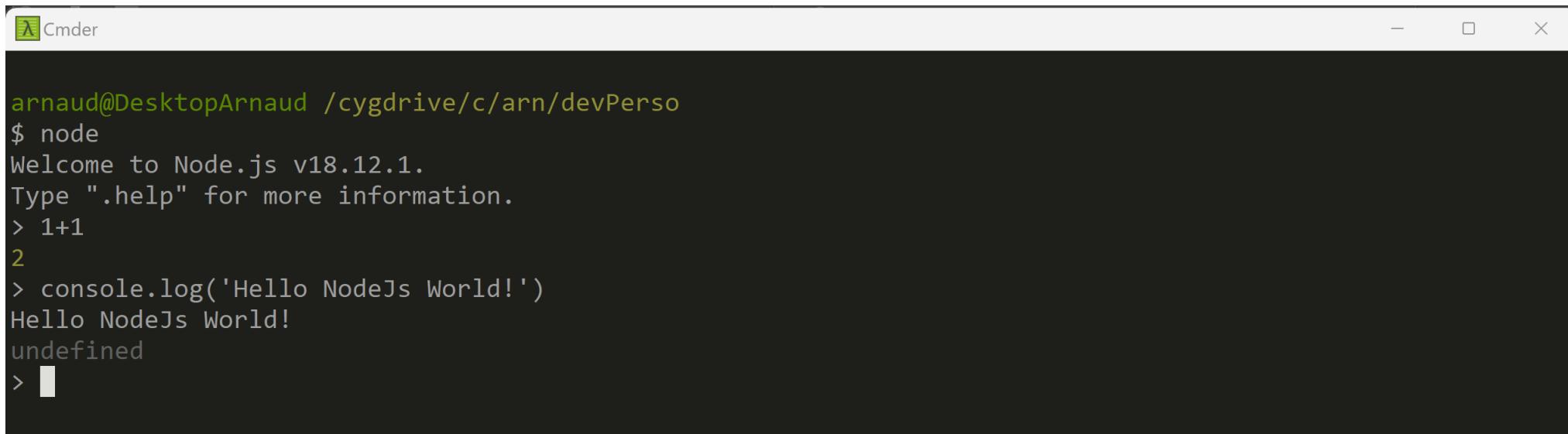


The screenshot shows a terminal window titled "Cmder". The command line shows the user's path: "arnaud@DesktopArnaud /cygdrive/c/arn/devPerso". The user has run the command "\$ node", which has resulted in the output: "Welcome to Node.js v18.12.1. Type ".help" for more information." A cursor is visible at the beginning of a new line starting with ">".

```
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
> 
```

Play with interactive Javascript eval

1+1
console.log('Hello')

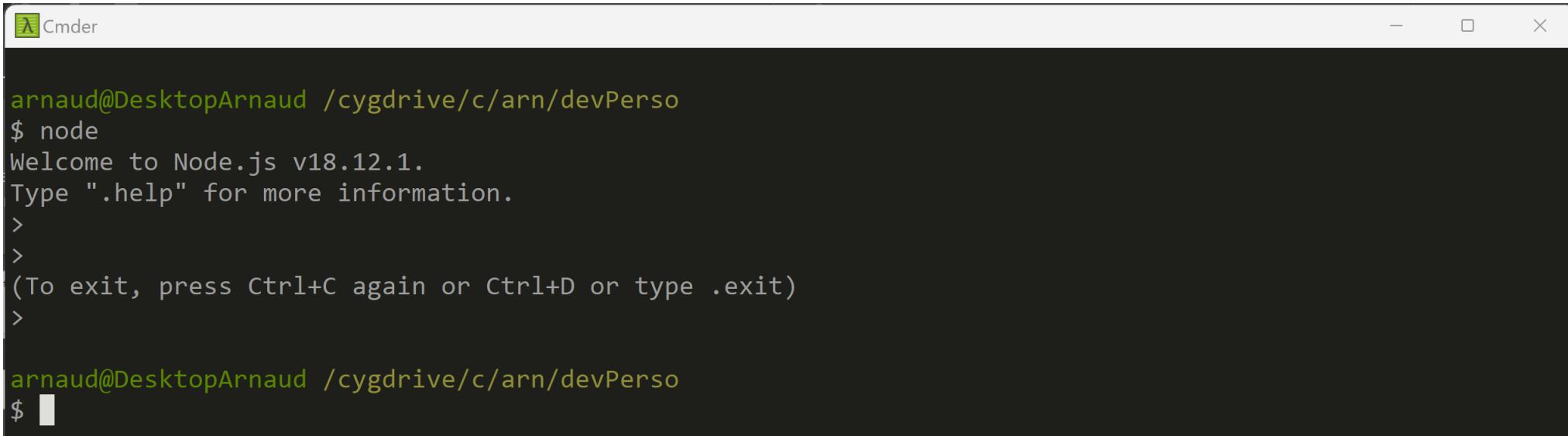


A screenshot of a terminal window titled "Cmder". The window shows a command-line interface for Node.js version 18.12.1. The user has run the command "\$ node" and is interacting with the Node.js REPL. They have entered the expression "1+1" which evaluates to "2". They then ran the command "console.log('Hello NodeJS World!')", which output "Hello NodeJS World!". Finally, they entered "undefined". The terminal has a dark background with light-colored text.

```
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
> 1+1
2
> console.log('Hello NodeJS World!')
Hello NodeJS World!
undefined
> 
```

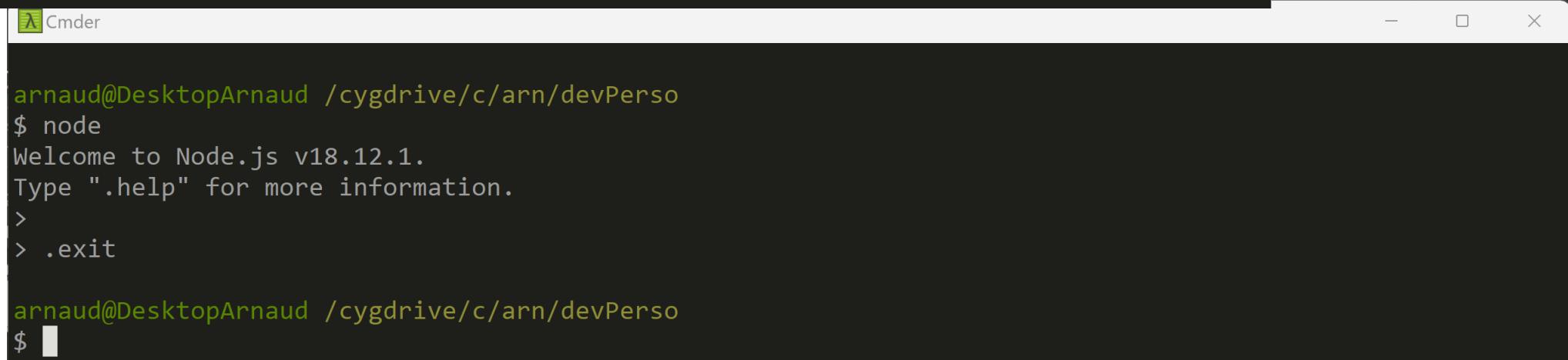
Stop Prompt ...

Ctrl+C Ctrl+C or .exit



```
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
>
>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
>

arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ █
```

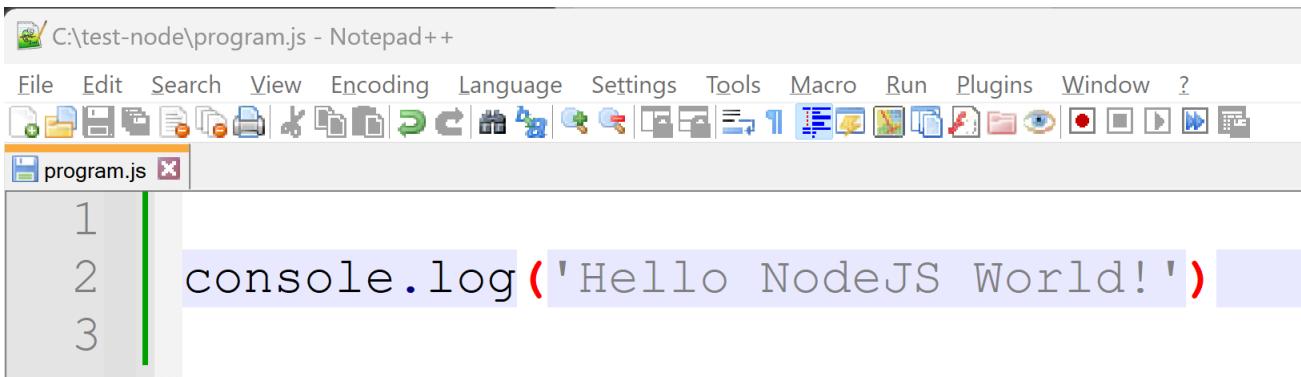


```
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ node
Welcome to Node.js v18.12.1.
Type ".help" for more information.
>
> .exit

arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ █
```

Start NodeJs on a « program.js » file

C:> node program.js



A screenshot of a terminal window titled "Cmder". The window shows the following session:

```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ node program.js
Hello NodeJS World!

arnaud@DesktopArnaud /cygdrive/c/test-node
$
```

using Export then Import [1/5]

Write a program using several *.js files:

export a function from file "my-util.js",

import in file "app.js", and call it

using Export then Import [2/5] : util.js ... the "export" part

Notice there are several possible syntaxes to do the export !

1/ using legacy NodeJs specific syntax

```
function myExportedFunc() { ... }  
module.exports = { myExportedFunc };
```

2/ OR newer standard

```
export myExportedFunc () { ... }
```

There can be several named exported symbols, or a default one.

The symbol can be a function, an object, a value (a constant / global), etc.

using Export then Import [3/5] : app.js ... the "import" part

Notice there are several possible syntaxes to do the import

1/ using legacy NodeJs specific syntax

```
const moduleAlias = require("filename");
moduleAlias.myExportedFunc();
```

2/ OR newer standard when

```
import {myExportedFunc } from 'filename';
myExportedFunc();
```

using Export then Import [4/5]

... being compatible with W3C Spec

If you choose the correct way : compatible with W3C Spec

i.e. using standard ECMAScript Module,

"import { .. } from .. "

instead of

".. = require(..)"

=> Then you can re-use the same code in backend(NodeJS) & Front-end Web Browser (Chrome)

The code is simpler / better

using Export then Import [5/5]

Find doc by yourself

ask ChatGPT / Google / NodeJS doc / Standard W3C Specification

what are "NodeJS module" compared to "ECMAScript module" ?
What is module System (CommonJS, ..) ?

What is "node_modules" path ?

Simplest Answer ... Scenario 1

The screenshot shows a development environment with the following details:

- Project:** td2 (C:\web\td2)
- Files:**
 - app.js:** Contains code to import and use a function from util.js.
 - util.js:** Contains the definition of the function imported in app.js.
- Code Completion:** A tooltip is visible over the word "const" in the app.js code.
- Syntax Errors:** A warning icon is shown above the util.js code.
- Debugger Console:** Shows the execution flow and log output:
 - "Starting app.js ..."
 - "call myExportedFunc > Object {x: 1, y: 2}"
 - ".. Finished app.js"
- Status Bar:** Shows file path (td2 > app.js), character count (1:4 (44 chars)), encoding (CRLF), and other settings.

Advanced

Scenario 2 (export multiple named symbols)

The screenshot shows the IntelliJ IDEA interface with two code files:

- app.js:** Contains code that requires 'util.js' and uses its 'myUtil' object. It logs "Starting app.js ..." and calls `myUtil.myExportedFunc(x: 1, y: 2);`. It also logs "... Finished app.js".
- util.js:** Contains a function `myExportedFunc(x, y)` that logs 'call myExportedFunc' with arguments {x: 1, y: 2}, returns `x+y`, and exports `myExportedFunc`.

The **Debugger** tool window at the bottom shows the execution flow:

- Starting app.js ...
- call myExportedFunc > Object {x: 1, y: 2}
- ... Finished app.js

Project navigation shows the files are located in a project named `td2` under `C:\web\td2`.

Scenario 3 .. export-import ERROR

Advanced

The screenshot shows a code editor with two files open:

- app.js**:

```
1 import { myExportedFunc } from './util.js'; ✓
2
3 console.log("Starting app.js ...");
4
5 myExportedFunc( x: 1, y: 2);
6
7 console.log("... Finished app.js");
8
```
- util.js**:

```
1
2 usages
2 export function myExportedFunc(x, y) {
3   console.log('call myExportedFunc', {x,y});
4   return x+y;
5
6
```

The IDE interface includes:

- Project** sidebar showing the project structure: `td2` (C:\web\td2) containing `.idea`, `app.js`, and `util.js`.
- Debug** tab selected in the bottom toolbar.
- Debugger** tab active in the bottom toolbar.
- Output** panel showing errors and warnings:
 - Warning:** (node:14972) Warning: To load an ES module, set "type": "module" in the package.json or use the .mjs extension. (Use `node --trace-warnings ...` to show where the warning was created)
 - Error:** Uncaught C:\web\td2\app.js:1 Loader:1320
- Status Bar** at the bottom showing: td2 > js app.js, 1:1, CRLF, UTF-8, 4 spaces.

Advanced

Scenario 4, type:"module"

The screenshot shows a development environment with the following components:

- Project View:** Shows a file structure for a project named "td2" located at "C:\web\td2". It includes files ".idea", "app.js", "package.json", and "util.js", along with "External Libraries" and "Scratches and Consoles".
- Editors:** Three code editors are open:
 - app.js:** Contains code that imports a function from "util.js" and logs a message.
 - util.js:** Contains a function that logs a message and returns the sum of its arguments.
 - package.json:** A JSON file specifying the project metadata, including "name": "td2", "version": "1.0.0", "description": "", "main": "app.js", "scripts": {"test": "echo \\"Er"}, and "type": "module".
- Debug Bar:** Shows a "Debug" button and a dropdown menu for "app.js". Below it are tabs for "Debugger" and "Debugger Console".
- Output Panel:** Displays the execution log:

```
Starting app.js ...
call myExportedFunc > Object {x: 1, y: 2}
.. Finished app.js
```
- Status Bar:** Shows the file path "td2 > package.json", status indicators for line endings (11:3 (16 chars)), character encoding (LF), and file format (UTF-8, 2 spaces, JSON: package).

Scenario 5, import file extension ".mjs" ERROR

Advanced

The screenshot shows a code editor interface with two files open:

- app.js**:

```
1 "use strict";
2 // const myUtil = require("./util.mjs");
3 const myUtil :{} = require("./util.mjs");
4 console.log("Starting app.js ...");
5
6 myUtil.myExportedFunc(x: 1, y: 2);
7
8 console.log(.. Finished app.ts");
```
- util.mjs**:

```
1 "use strict";
2 1 usage
3 function myExportedFunc(x, y) {
4   console.log('call myExportedFunc', { x, y });
5   return x + y;
6
7 exports.myExportedFunc = myExportedFunc;
```

The bottom status bar displays the error message: **Uncaught Error [ERR_REQUIRE_ESM]: require() of ES Module C:\web\td2\util.mjs not supported.** at loader:1204.

Scenario 6, file ".mjs" + type:module

Advanced

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Structure:** The left sidebar shows a project named "td2" containing "app.js", "util.mjs", and "package.json".
- Editors:** Three code editors are open:
 - app.js:** Contains code that logs "Starting app.js ...", imports `myExportedFunc` from `util.mjs`, calls it with arguments `x: 1, y: 2`, and logs "... Finished app.js".
 - util.mjs:** Contains code that logs "Starting util.mjs ...", defines an export function `myExportedFunc` that adds `x` and `y`, and logs the result.
 - package.json:** Contains a JSON object defining the project metadata, including "name": "td2", "version": "1.0.0", "description": "", "main": "app.js", "scripts": {"test": "echo \\"Hello World\\!"}, "author": "", "license": "ISC", and **"type": "module"**.
- Toolbars:** The top toolbar includes icons for file operations, search, and settings.
- Bottom Status Bar:** Shows the path "td2 > app.js", the current time "5:18", line separator "LF", encoding "UTF-8", and code style "4 spaces".
- Debugger:** The bottom panel features a "Debug" tab and a "Debugger Console" tab. The "Debugger Console" tab displays the execution log:
 - "Starting app.js ..."
 - "call myExportedFunc > Object {x: 1, y: 2}"
 - "... Finished app.js"

Advanced

Best Answer - Scenario 7 (TypeScript .. see end of TD)

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure with files `app.ts`, `util.ts`, and `tsconfig.json`.
- Code Editors:** Two tabs are open:
 - `app.ts`: Contains code to import `myExportedFunc` from `util.ts`, log "Starting app.ts ...", call `myExportedFunc(1, 2)`, and log "... Finished app.ts".
 - `util.ts`: Contains code to export `myExportedFunc` which logs "call myExportedFunc" and returns the sum of `x` and `y`.
- Toolbars:** Includes tabs for `TS`, `Debug`, and `app`. The `app` tab is selected.
- Debugger Tool Window:** Shows the execution flow:
 - Step 1: `Starting app.ts ...` (app.ts:5)
 - Step 2: `call myExportedFunc > Object {x: 1, y: 2}` (util.ts:5)
 - Step 3: `.. Finished app.ts` (app.ts:7)
- Status Bar:** Displays file statistics: 5:1 CRLF, UTF-8, TypeScript 5.1.3, 4 spaces.

Scenario 7 ... Generated *.js code for *.ts

Advanced

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The left sidebar shows a project named "td2" containing files "app.ts", "app.js", "tsconfig.json", "util.ts", and "util.js".
- Editors:** Two editors are open:
 - app.ts:** Contains the following code:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
// const myExportedFunc = require('./util.ts');
const util_1 = require("./util");
console.log("Starting app.ts ...");
(util_1.myExportedFunc)(1, 2);
console.log("... Finished app.ts");
```
 - util.js:** Contains the following code:

```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.myExportedFunc = void 0;
1 usage
function myExportedFunc(x, y) {
    console.log('call myExportedFunc', { x, y });
    return x + y;
}
exports.myExportedFunc = myExportedFunc;
```
- Toolbars:** The top toolbar includes icons for file operations, search, and settings.
- Bottom Status Bar:** Shows the path "td2 > util.js", file encoding "1:1 LF", character encoding "UTF-8", version control "TypeScript 5.1.3", and code style "4 spaces".

Node ... need Package !!!

Npm = Node Package Manager

google.com/search?q=node+packages&rlz=1C1CHBF_frFR928FR928&oq=node+packages&aqs=chrome..69i57j0i512l5j0i390i650l2.5132j0j15&sourceid=chrome&ie=UTF-8

About 539,000,000 results (0.60 seconds)

npm
The free npm Registry has become the center of JavaScript code sharing, and with more than two million packages, the largest software registry in the world. Our ...

Package
Intro. This module provides an easy and simple way to export ...

About packages and modules
About modules. A module is any file or directory in the ...

Packages and modules
Documentation for the npm registry, website, and command-line ...

Using npm packages in your ...
If you are creating a Node.js module, you can use a package ...

npm
Computer program

npm is a package manager for the JavaScript programming language maintained by npm, Inc. npm is the default package manager for the JavaScript runtime environment Node.js. It consists of a command line client, also called npm, and an online database of public and paid-for private packages,

C:> npm

```
Cmder
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm
npm <command>

Usage:

npm install      install all the dependencies in your project
npm install <foo> add the <foo> dependency to your project
npm test         run this project's tests
npm run <foo>   run the script named <foo>
npm <command> -h quick help on <command>
npm -l          display usage info for all commands
npm help <term> search for help on <term> (in a browser)
npm help npm    more involved overview (in a browser)

All commands:

access, adduser, audit, bugs, cache, ci, completion,
config, dedupe, deprecate, diff, dist-tag, docs, doctor,
edit, exec, explain, explore, find-dupes, fund, get, help,
help-search, hook, init, install, install-ci-test,
install-test, link, ll, login, logout, ls, org, outdated,
owner, pack, ping, pkg, prefix, profile, prune, publish,
query, rebuild, repo, restart, root, run-script, search,
set, shrinkwrap, star, stars, start, stop, team, test,
token, uninstall, unpublish, unstar, update, version, view,
whoami

Specify configs in the ini-formatted file:
  C:\apps\cygwin64\home\arnaud\.npmrc
```

\$ npm init

```
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

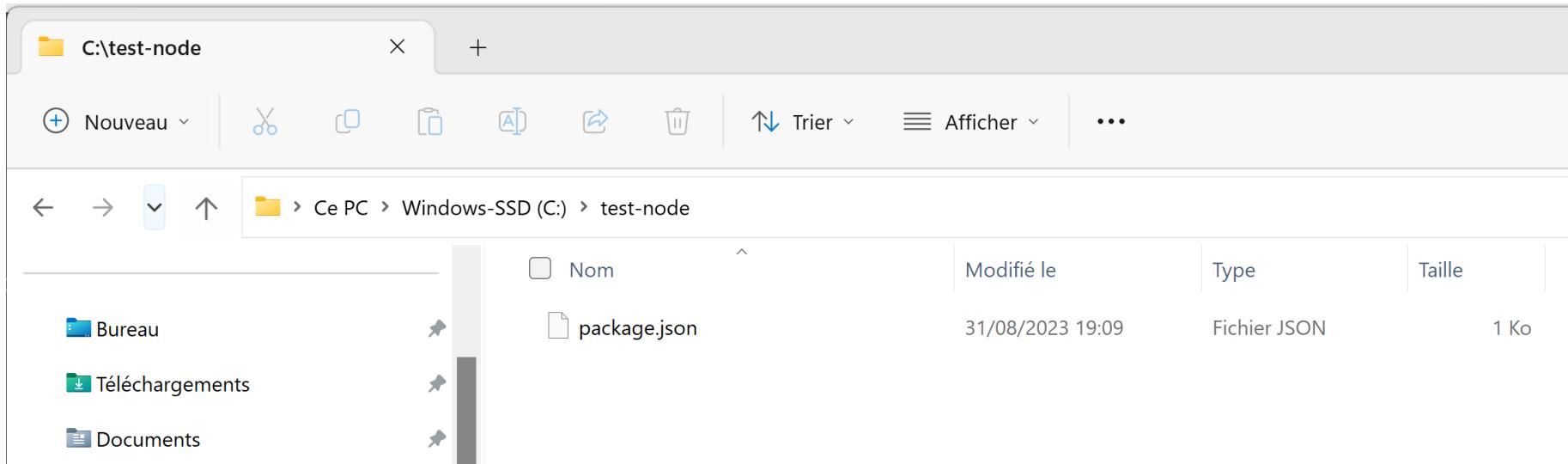
Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (test-node)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\arn\devPerso\Presentations\web\test-node\package.json:

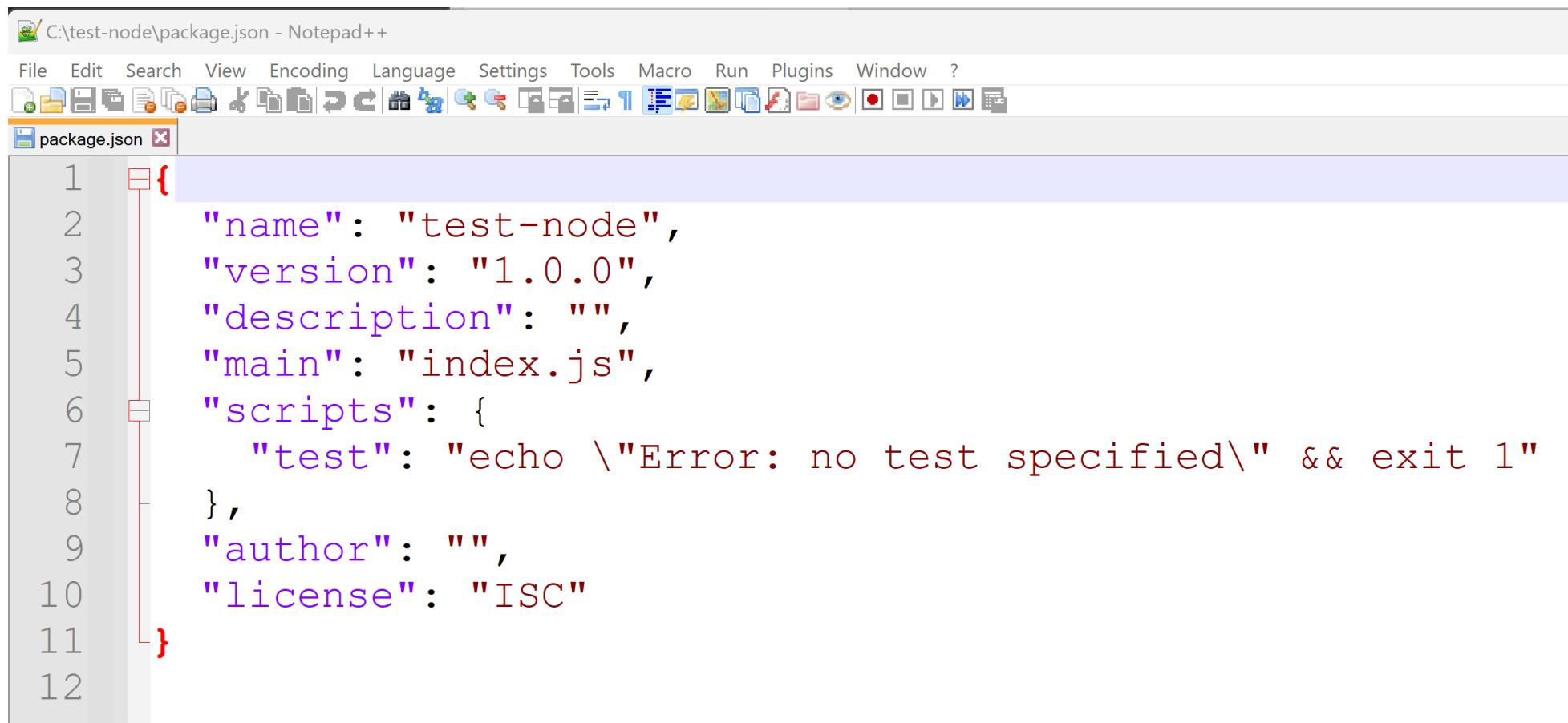
{
  "name": "test-node",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) █
```

npm init => open File explorer
=> 1 File « package.json »



package.json



A screenshot of the Notepad++ text editor showing the contents of a `package.json` file. The file path is `C:\test-node\package.json`. The code is as follows:

```
1 {  
2   "name": "test-node",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }  
12
```

The code is color-coded: curly braces (`{`, `}`) are red, strings are purple, and the file number column (1-12) is light blue.

Search Package dependencies ...

Npm search ... Registry (=Repository)

A screenshot of a Google search results page. The search bar at the top contains the query "npm search". Below the search bar, there are several navigation links: Images, Videos, News, Shopping, Books, Maps, Flights, and Finance. The main search results section starts with a message indicating approximately 192 million results found in 0.39 seconds. The first result is a link to "npm Docs" from <https://docs.npmjs.com>, specifically the "cli > commands > npm-search" page. The second result is a link to "npm-search" with a brief description: "Search the registry for packages matching the search terms. **npm search** performs a linear, incremental, lexically-ordered search through package metadata for ...". Below this, there are links to "Synopsis · Description · Configuration". The third result is a link to "npm.js" from <https://www.npmjs.com>. The fourth result is a link to "npm" with a brief description: "The free **npm** Registry has become the center of JavaScript code sharing, and with more than two million packages, the largest software registry in the world. Our ...". Below this, there are links to "Express · React · About npm · Npm Docs".

← → C google.com/search?q/npm+search&sca_esv=561694184&rlz=1C1CHBF_frFR928FR928&sxsrf=AB5stBiQAUFBtC8abXy4KljzfKDHH2f3cg%3A16

Google

npm search

X |

Images Videos News Shopping Books Maps Flights Finance

About 192,000,000 results (0.39 seconds)

[npm Docs](#)
<https://docs.npmjs.com> › cli › commands › npm-search

[:::](#)

[npm-search](#)

Search the registry for packages matching the search terms. **npm search** performs a linear, incremental, lexically-ordered search through package metadata for ...

[Synopsis](#) · [Description](#) · [Configuration](#)

[npm.js](#)
<https://www.npmjs.com>

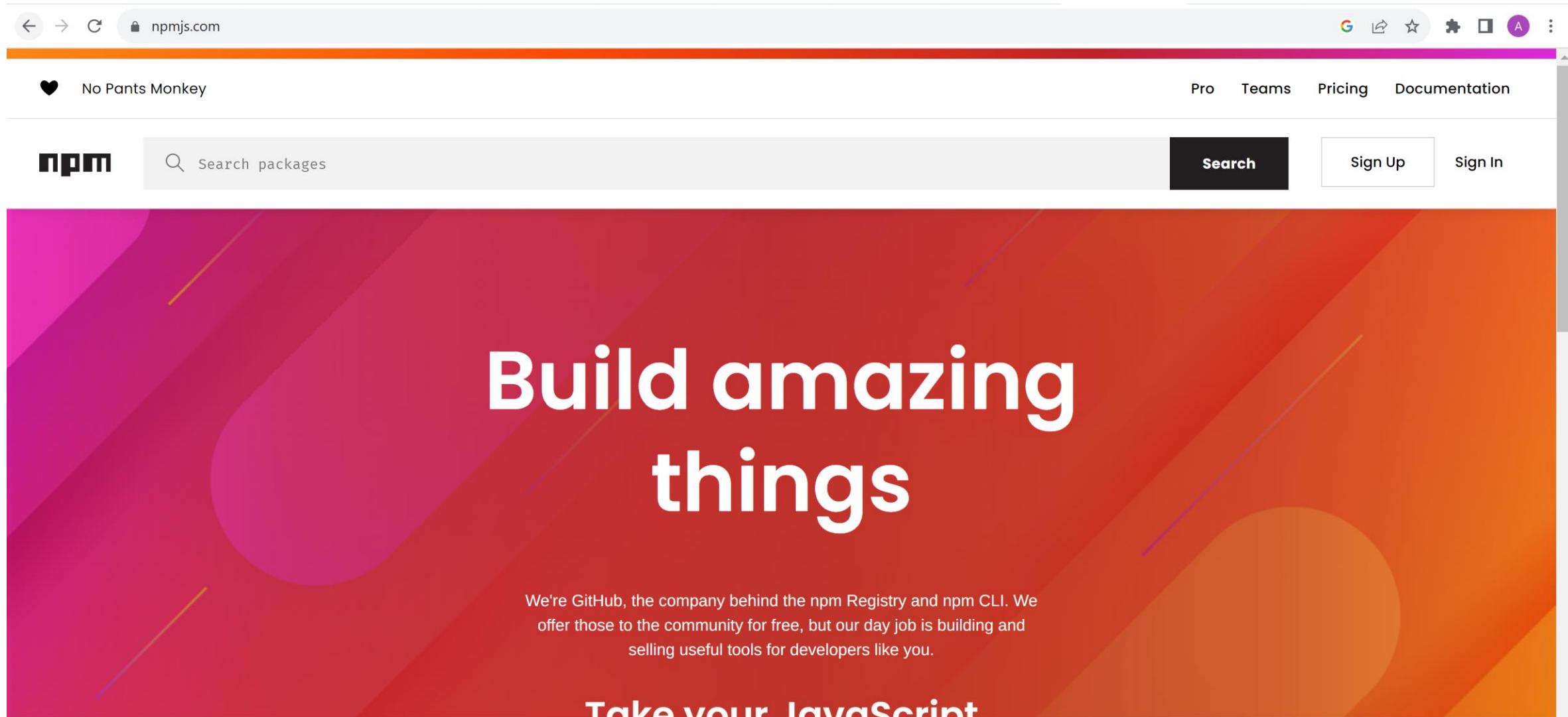
[:::](#)

[npm](#)

The free **npm** Registry has become the center of JavaScript code sharing, and with more than two million packages, the largest software registry in the world. Our ...

[Express](#) · [React](#) · [About npm](#) · [Npm Docs](#)

<https://npmjs.com>



A screenshot of the npmjs.com homepage. The page features a large orange header bar with the URL "npmjs.com" in the address bar. Below the header is a navigation bar with links for "Pro", "Teams", "Pricing", and "Documentation". On the left, there's a user profile icon for "No Pants Monkey". A search bar with the placeholder "Search packages" and a "Search" button are also present. The main content area has a vibrant orange and red abstract background with white text. The central text reads "Build amazing things" in a large, bold, sans-serif font. Below this, a smaller paragraph states: "We're GitHub, the company behind the npm Registry and npm CLI. We offer those to the community for free, but our day job is building and selling useful tools for developers like you." At the bottom, the text "Take your JavaScript" is partially visible.

← → C 🔒 npmjs.com G ↗ ☆ ⚙ □ A :

No Pants Monkey

Pro Teams Pricing Documentation

npm Search packages Search Sign Up Sign In

Build amazing things

We're GitHub, the company behind the npm Registry and npm CLI. We offer those to the community for free, but our day job is building and selling useful tools for developers like you.

Take your JavaScript

Search « express » package

The screenshot shows the npmjs.com search interface with the query "express" entered into the search bar. The results list various packages related to Express.js, each with its name, description, and latest version. The background features a large, abstract orange and red graphic.

Package	Description	Latest Version
<code>express</code>	Fast, unopinionated, minimalist web framework	4.18.2
<code>express-validator</code>	Express middleware for the validator module.	7.0.1
<code>express-handlebars</code>	A Handlebars view engine for Express which doesn't suck.	7.1.2
<code>express-session</code>	Simple session middleware for Express	1.17.3
<code>express-brute</code>	A brute-force protection middleware for express routes that rate limits incoming requests	1.0.1
<code>express-favicon</code>	Favicon middleware for express-like applications	2.0.4
<code>express-rate-limit</code>	Basic IP rate-limiting middleware for Express. Use to limit repeated requests to public APIs and/or endpoints such as password reset.	6.10.0
<code>express-promise-router</code>	A lightweight wrapper for Express 4's Router that allows middleware to return promises	4.1.1
<code>express-fileupload</code>	Simple express file upload middleware that wraps around Busboy	1.4.0
<code>express-openapi-validator</code>	Automatically validate API requests and responses with OpenAPI 3 and Express.	5.0.6

Package express ... version / doc / install

npmjs.com/package/express

Pro Teams Pricing Documentation

Nestable Processes Mutate

Search packages

Search Sign Up Sign In

express DT

4.18.2 • Public • Published a year ago

Readme Code Beta 31 Dependencies 72,994 Dependents 270 Versions

express

Fast, unopinionated, minimalist web framework for Node.js.

npm v4.18.2 install size 1.89 MB downloads 123.2M/month

```
const express = require('express')
const app = express()

app.get('/', function (req, res) {
  res.send('Hello World')
})
```

Install

```
npm i express
```

Repository

github.com/expressjs/express

Homepage

expressjs.com/

Weekly Downloads

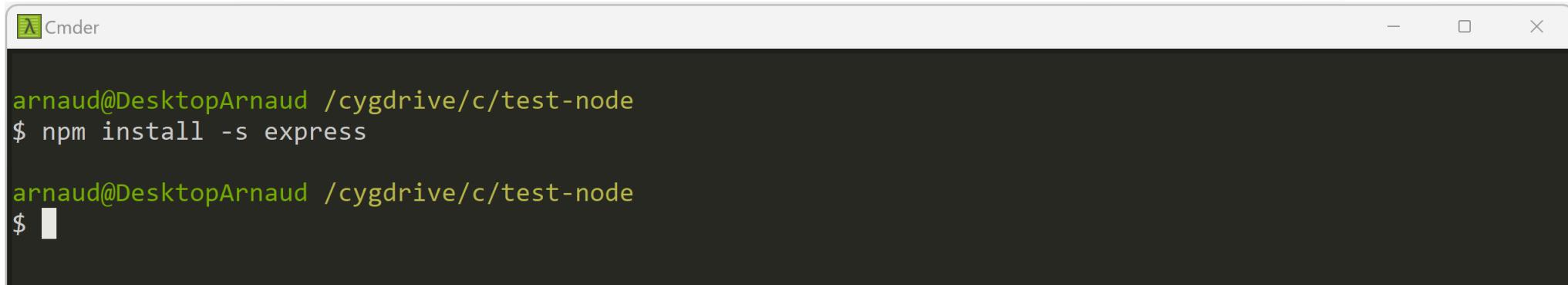
29,144,294

Version License

This screenshot shows the npmjs.com package page for 'express'. At the top, there's a navigation bar with links for Pro, Teams, Pricing, and Documentation. Below that is a header with a search bar, a 'Search' button, and links for Sign Up and Sign In. The main content area starts with the package name 'express' and its version '4.18.2'. It indicates the package is 'Public' and was published a year ago. Below this are tabs for Readme, Code (Beta), 31 Dependencies, 72,994 Dependents, and 270 Versions. A large 'express' logo is prominently displayed. Below the logo, a description states it's a 'Fast, unopinionated, minimalist web framework for Node.js.'. There are also stats for npm version (v4.18.2), install size (1.89 MB), and monthly downloads (123.2M/month). A code snippet shows a simple 'Hello World' application. To the right, there are sections for 'Install' (with a command line example), 'Repository' (link to GitHub), 'Homepage' (link to expressjs.com), and 'Weekly Downloads' (a chart showing 29,144,294). At the bottom, there are links for 'Version' and 'License'.

Add npm dependency « express »

C:> npm install –s express
(equivalent: npm install --save express)



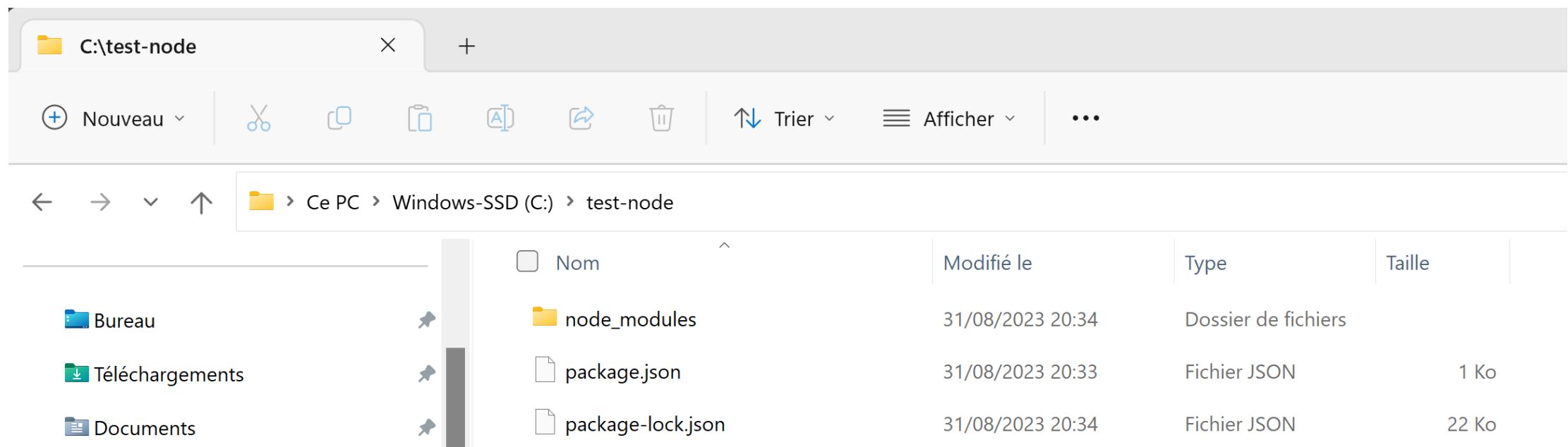
```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm install -s express

arnaud@DesktopArnaud /cygdrive/c/test-node
$ █
```

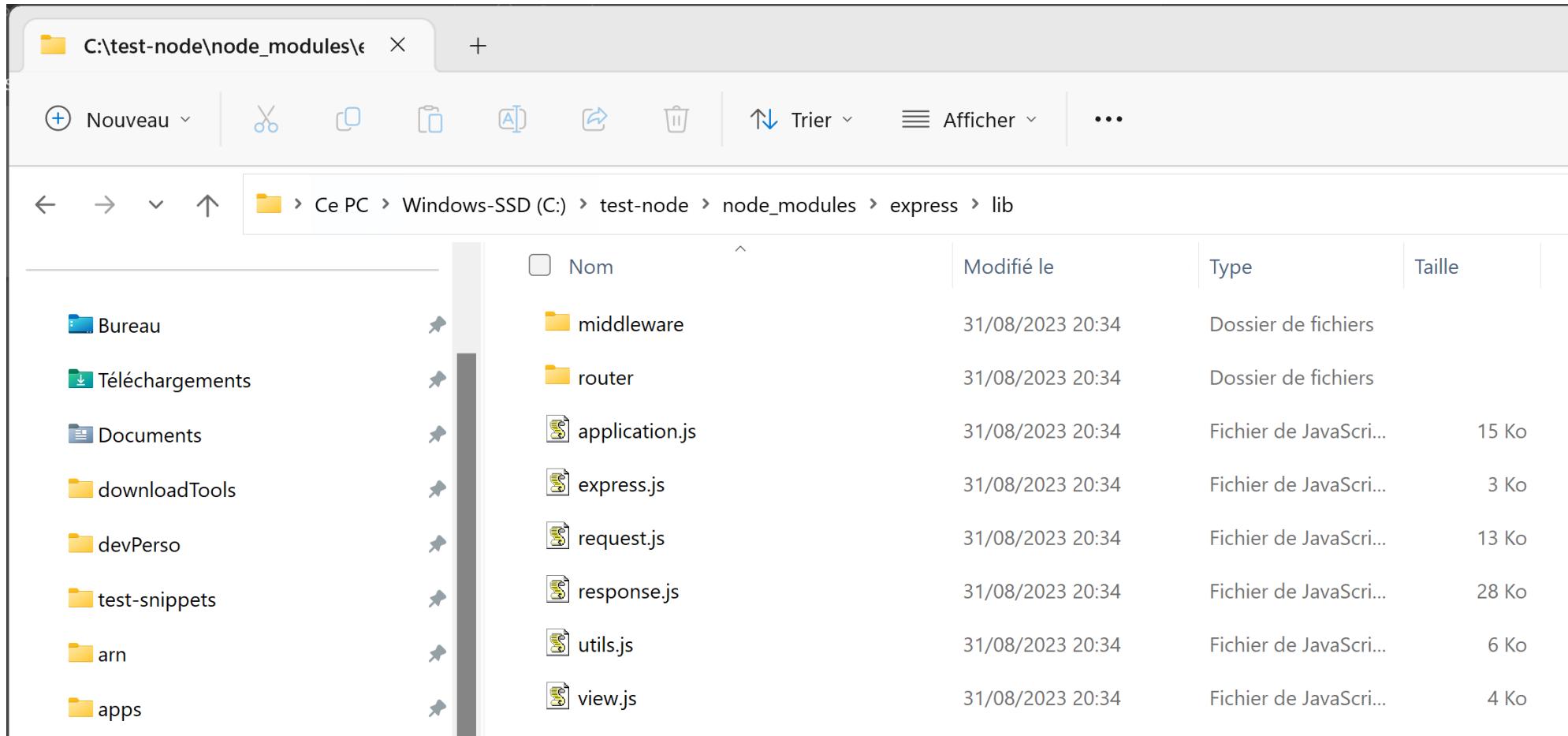
A screenshot of a terminal window titled "Cmder". The window has a dark background and light-colored text. It shows a command-line session where the user is in a directory called "test-node". The user types the command "npm install -s express" and presses enter. The terminal then displays the command again followed by a prompt symbol (\$) and a small black square cursor.

npm --save => save to package.json file
+ package-lock.json

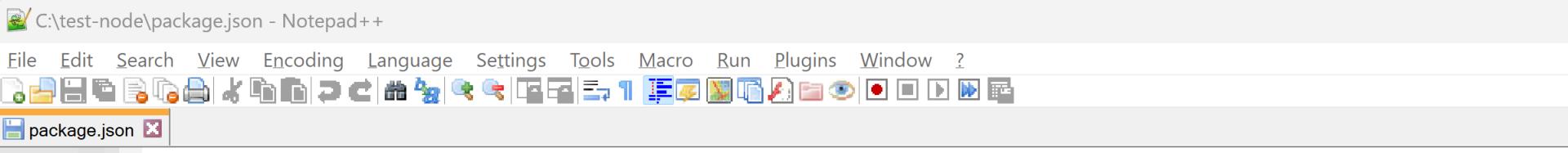
+ download **/** to node_modules/



node_modules/express/lib/**/*.js



... 1 Line added in package.json dependencies: { «pck-name» : « version », ... }



C:\test-node\package.json - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

package.json

```
1  {
2      "name": "test-node",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      "scripts": {
7          "test": "echo \\\"Error: no test specified\\\" && exit 1"
8      },
9      "author": "",
10     "license": "ISC",
11     "dependencies": {
12         "express": "^4.18.2"
13     }
14 }
15 }
```

<https://docs.npmjs.com/about-semantic-versioning>

Using semantic versioning to specify update types your package can accept

You can specify which update types your package can accept from dependencies in your package's `package.json` file.

For example, to specify acceptable version ranges up to 1.0.4, use the following syntax:

- Patch releases: `1.0` or `1.0.x` or `~1.0.4`
- Minor releases: `1` or `1.x` or `^1.0.4`
- Major releases: `*` or `x`

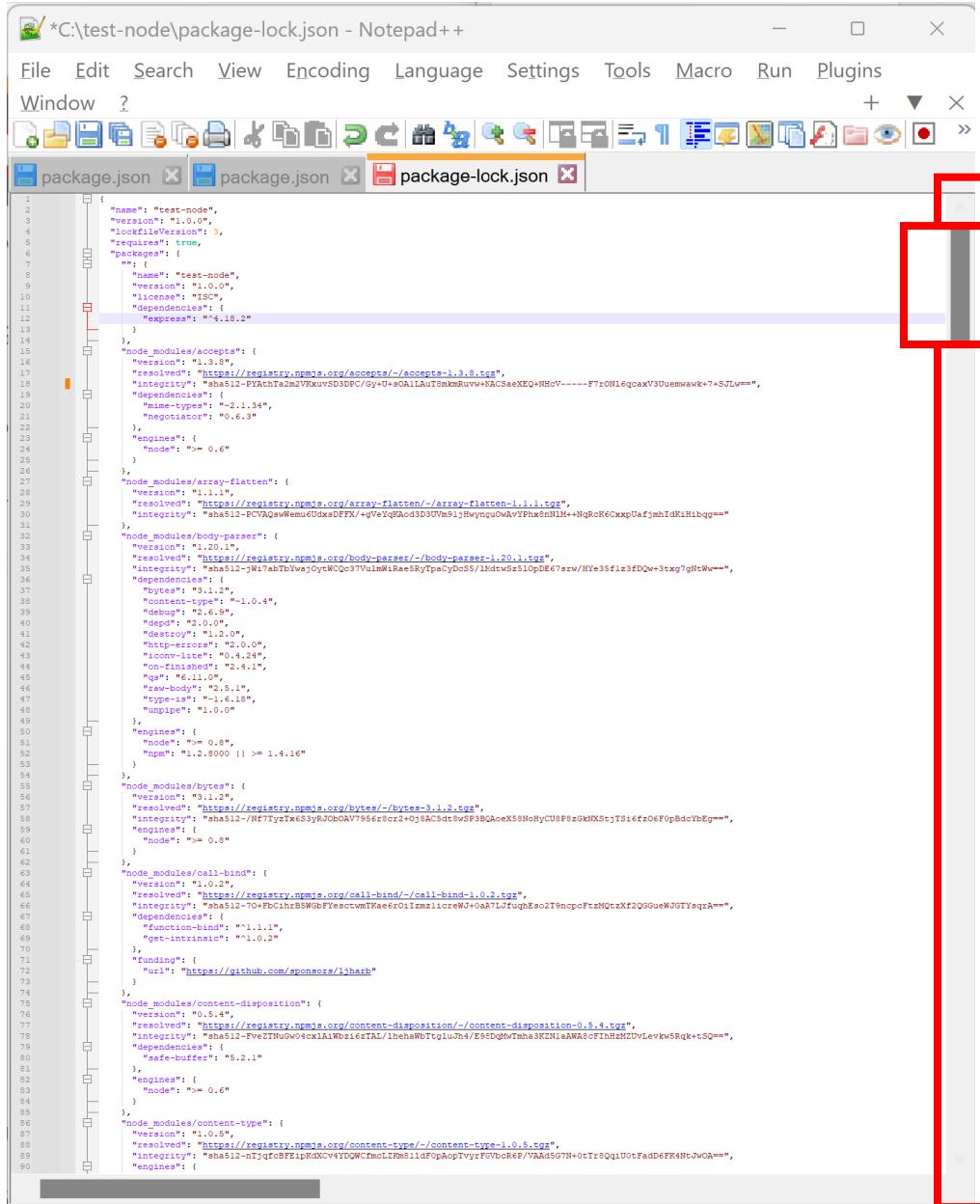
For more information on semantic versioning syntax, see the [npm semver calculator](#).

Example

```
"dependencies": {  
  "my_dep": "^1.0.0",  
  "another_dep": "~2.2.0"  
},
```

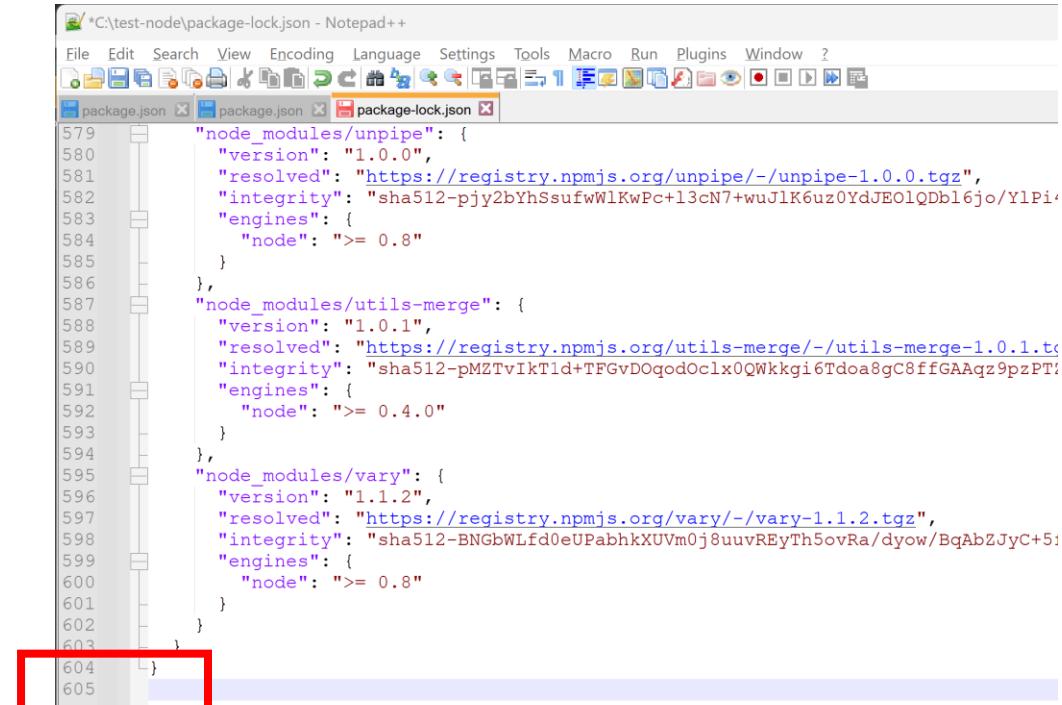


Locking versions number ... package-lock.json



```
*C:\test-node\package-lock.json - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins  
Window ?  
package.json package.json package-lock.json  
  
1 {  
2   "name": "test-node",  
3   "version": "1.0.0",  
4   "lockfileVersion": 3,  
5   "requires": true,  
6   "packages": [  
7     {  
8       "name": "test-node",  
9       "version": "1.0.0",  
10      "license": "ISC",  
11      "dependencies": {  
12        "express": "4.16.2"  
13      },  
14    },  
15    "node_modules/accepts": {  
16      "version": "1.3.0",  
17      "resolved": "https://registry.npmjs.org/accepts/-/accepts-1.3.0.tgz",  
18      "integrity": "sha512-PtAChIa2m7TkuvS3DfPC/G+U+eAIaLkut0mkmuWv+9NCSeXEQ+NHCv-----F7zON16qcaXV3Uemwawk+7SJLw==",  
19      "dependencies": {  
20        "mime-types": "2.1.34",  
21        "negotiator": "0.6.3"  
22      },  
23      "engines": {  
24        "node": ">= 0.6"  
25      }  
26    },  
27    "node_modules/array-flatten": {  
28      "version": "1.1.1",  
29      "resolved": "https://registry.npmjs.org/array-flatten/-/array-flatten-1.1.1.tgz",  
30      "integrity": "sha512-PCVAQswWemudUdxsDFX/+yVeYQKad3DUvM91jHwlynquOwAyPHx8nNlM++NgRcK6CxpxUafjmhdKiHibqq==",  
31    },  
32    "node_modules/body-parser": {  
33      "version": "1.20.1",  
34      "resolved": "https://registry.npmjs.org/body-parser/-/body-parser-1.20.1.tgz",  
35      "integrity": "sha512-jh7abTbVwOyTCQc37VulWRae5KyPacCyS5/Mdtw5z51OpDE7srw/HYe35fiz3fDw+3txg7gNtWw==",  
36      "dependencies": {  
37        "bytes": "3.1.2",  
38        "content-type": "1.1.0-4",  
39        "debug": "2.6.9",  
40        "debugs": "2.0.0",  
41        "destory": "1.2.0",  
42        "http-errors": "2.0.0",  
43        "iconv-lite": "0.4.24",  
44        "on-finished": "2.4.1",  
45        "qs": "6.11.0",  
46        "raw-body": "2.5.1",  
47        "type-is": "1.1.16",  
48        "unpipe": "1.0.0"  
49      },  
50      "engines": {  
51        "node": "0.8",  
52        "npm": "1.2.8000 || >= 1.4.16"  
53      }  
54    },  
55    "node_modules/bytes": {  
56      "version": "3.1.2",  
57      "resolved": "https://registry.npmjs.org/bytes/-/bytes-3.1.2.tgz",  
58      "integrity": "sha512-Hd7yTkh463y80bOHV734csseD+0j8Ac5dts8gFBQoeX58NoMyCUBP8zGkNKS7TSi6fz06f0pBdcYbEg==",  
59      "engines": {  
60        "node": "0.8"  
61      }  
62    },  
63    "node_modules/call-bind": {  
64      "version": "1.0.2",  
65      "resolved": "https://registry.npmjs.org/call-bind/-/call-bind-1.0.2.tgz",  
66      "integrity": "sha512-70+8C1hrB5GbFYescwTKtaee011mzmlcreW+0a7JtuqEco2T9ncpoFtzNQtzXf2QGGueWJGTysqrA==",  
67      "functions": {  
68        "function-bind": "1.1.1",  
69        "get-intrinsic": "1.0.2"  
70      },  
71      "funding": {  
72        "url": "https://github.com/sponsors/ljharb"  
73      }  
74    },  
75    "node_modules/content-disposition": {  
76      "version": "0.5.4",  
77      "resolved": "https://registry.npmjs.org/content-disposition/-/content-disposition-0.5.4.tgz",  
78      "integrity": "sha512-FveINu04eAIaM0z1e2TAU/IehaHb7rgLuh7/E5IDMw7mha3KZniIAWA6cF1H2zHCUvLevkWSRqk+tSQ==",  
79      "dependencies": {  
80        "safe-buffer": "5.2.1"  
81      },  
82      "engines": {  
83        "node": ">= 0.6"  
84      }  
85    },  
86    "node_modules/content-type": {  
87      "version": "1.0.5",  
88      "resolved": "https://registry.npmjs.org/content-type/-/content-type-1.0.5.tgz",  
89      "integrity": "sha512-nTjqcfBFipKdxCv4YDQWCFmcl2Km$1ldOpkOpTvyrFGVbcr6P/VAAdSG7N+0tTr8Q1U0tFadDF4NtJwOA==",  
90      "engines": {  
91        "node": "0.6"  
92      }  
93  }
```

File is **HUGE** ...
Like expanded package/package/package !!!
ONLY « express » → 605 lines



```
*C:\test-node\package-lock.json - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
package.json package.json package-lock.json  
  
579 "node_modules/unpipe": {  
580   "version": "1.0.0",  
581   "resolved": "https://registry.npmjs.org/unpipe/-/unpipe-1.0.0.tgz",  
582   "integrity": "sha512-pjy2bYhSsufwWlKwPc+l3cN7+wuJlK6uz0YdJE01QDb16jo/YlPi",  
583   "engines": {  
584     "node": ">= 0.8"  
585   }  
586 },  
587 "node_modules/utils-merge": {  
588   "version": "1.0.1",  
589   "resolved": "https://registry.npmjs.org/utils-merge/-/utils-merge-1.0.1.tgz",  
590   "integrity": "sha512-pM2TvIkTid+TFGvDOqodOclx0QWkkgi6Tdoa8gC8ffGAQz9pzPT",  
591   "engines": {  
592     "node": ">= 0.4.0"  
593   }  
594 },  
595 "node_modules/vary": {  
596   "version": "1.1.2",  
597   "resolved": "https://registry.npmjs.org/vary/-/vary-1.1.2.tgz",  
598   "integrity": "sha512-BNGbWLfd0eUPabhkXUVm0j8uuvREyTh5ovRa/dyow/BqAbZJyC+5:",  
599   "engines": {  
600     "node": ">= 0.8"  
601   }  
602 },  
603 },  
604 }  
605 }
```

`node_modules/*/*/*/*/*.js`

Directory `node_modules`

1/ may be BIG

.... ~300 Mega is frequent with few dependencies

`$ du -sh node_modules`

2/ can be deleted + re-downloaded safely

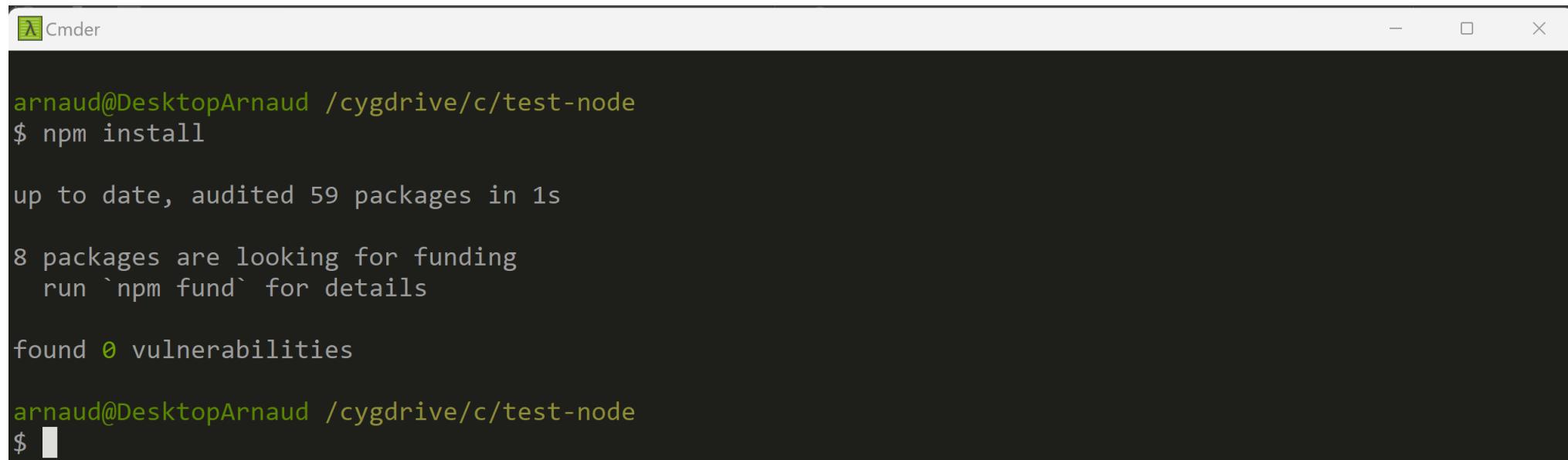
`$ rm -rf node_modules; npm install`

3/ should NEVER be committed to any git repository

`$ echo node_modules/ >> .gitignore`

After getting some web project source code...
=> downloads to re-fill node_modules/**
as defined in package.json

\$ npm install



A screenshot of a terminal window titled "Cmder". The terminal shows the following command and its execution:

```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm install

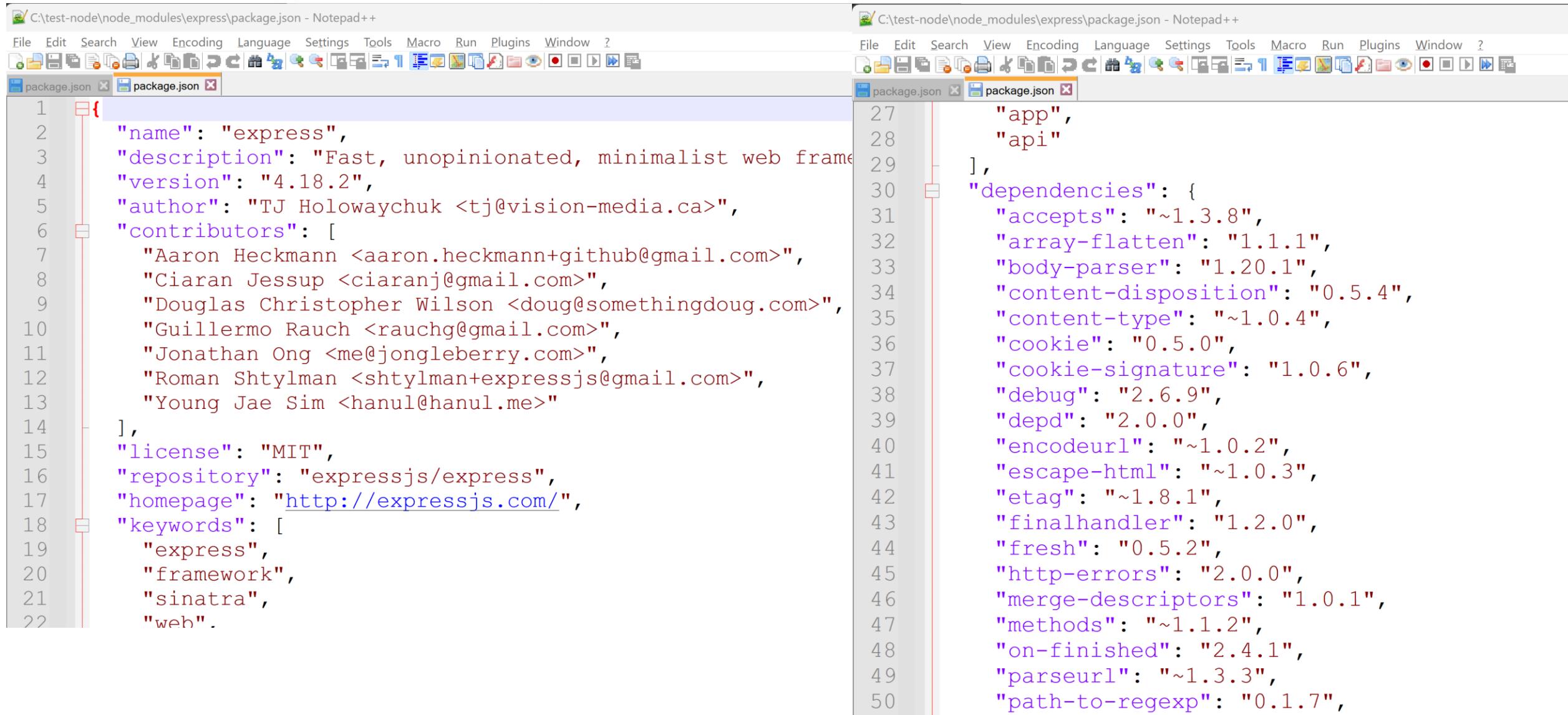
up to date, audited 59 packages in 1s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

arnaud@DesktopArnaud /cygdrive/c/test-node
$ █
```

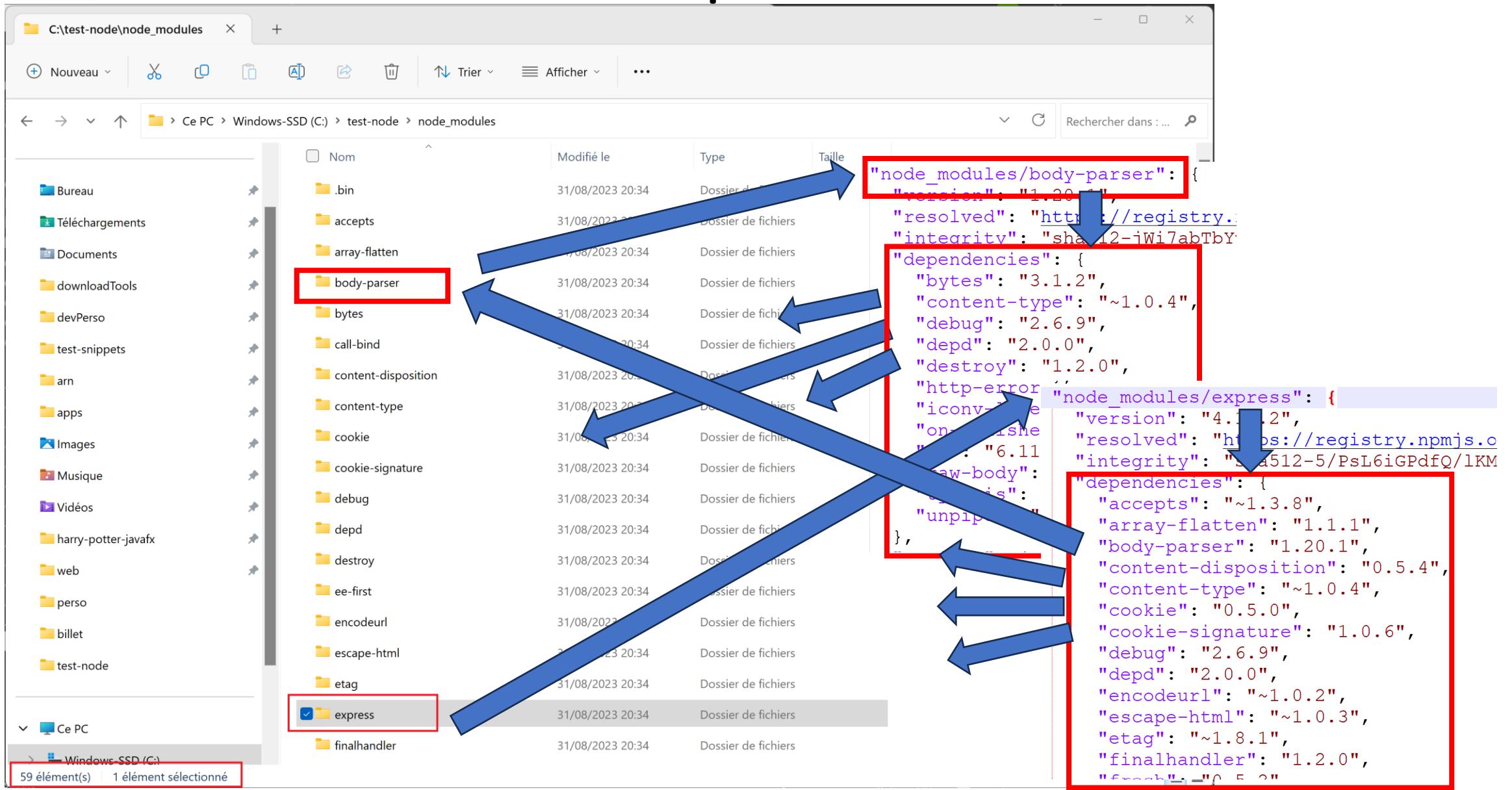
node_modules/express/package.json



The image shows two side-by-side Notepad++ windows displaying the same JSON file, `package.json`, for the `express` module. The left window shows the first half of the file, and the right window shows the second half.

```
1 {  
2   "name": "express",  
3   "description": "Fast, unopinionated, minimalist web framework",  
4   "version": "4.18.2",  
5   "author": "TJ Holowaychuk <tj@vision-media.ca>",  
6   "contributors": [  
7     "Aaron Heckmann <aaron.heckmann+github@gmail.com>",  
8     "Ciaran Jessup <ciaranj@gmail.com>",  
9     "Douglas Christopher Wilson <doug@somethingdoug.com>",  
10    "Guillermo Rauch <rauchg@gmail.com>",  
11    "Jonathan Ong <me@jongleberry.com>",  
12    "Roman Shtylman <shtylman+expressjs@gmail.com>",  
13    "Young Jae Sim <hanul@hanul.me>"  
14  ],  
15  "license": "MIT",  
16  "repository": "expressjs/express",  
17  "homepage": "http://expressjs.com/",  
18  "keywords": [  
19    "express",  
20    "framework",  
21    "sinatra",  
22    "web"  
23  ],  
24  "dependencies": {  
25    "accepts": "~1.3.8",  
26    "array-flatten": "1.1.1",  
27    "body-parser": "1.20.1",  
28    "content-disposition": "0.5.4",  
29    "content-type": "~1.0.4",  
30    "cookie": "0.5.0",  
31    "cookie-signature": "1.0.6",  
32    "debug": "2.6.9",  
33    "depd": "2.0.0",  
34    "encodeurl": "~1.0.2",  
35    "escape-html": "~1.0.3",  
36    "etag": "~1.8.1",  
37    "finalhandler": "1.2.0",  
38    "fresh": "0.5.2",  
39    "http-errors": "2.0.0",  
40    "merge-descriptors": "1.0.1",  
41    "methods": "~1.1.2",  
42    "on-finished": "2.4.1",  
43    "parseurl": "~1.3.3",  
44    "path-to-regexp": "0.1.7",  
45  },  
46  "devDependencies": {  
47    "body-parser": "1.19.0",  
48    "content-disposition": "0.5.4",  
49    "content-type": "1.2.7",  
50    "cookie": "0.5.0",  
51    "cookie-signature": "1.0.6",  
52    "debug": "2.6.9",  
53    "depd": "2.0.0",  
54    "encodeurl": "~1.0.2",  
55    "escape-html": "~1.0.3",  
56    "etag": "~1.8.1",  
57    "finalhandler": "1.2.0",  
58    "fresh": "0.5.2",  
59    "http-errors": "2.0.0",  
60    "merge-descriptors": "1.0.1",  
61    "methods": "~1.1.2",  
62    "on-finished": "2.4.1",  
63    "parseurl": "~1.3.3",  
64    "path-to-regexp": "0.1.7",  
65    "qs": "6.5.1",  
66    "setimmediate": "1.0.0",  
67    "statuses": "1.5.0",  
68    "type-is": "1.6.17",  
69    "util-deprecate": "1.0.2",  
70    "vary": "1.0.3",  
71  },  
72  "optionalDependencies": {}  
73}
```

package.json → package.json → ... package.json
transitive dependencies



npm ls --all

```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm ls --all
test-node@1.0.0 C:\test-node
`-- express@4.18.2
  +-+ accepts@1.3.8
  | +-+ mime-types@2.1.35
  | | `-- mime-db@1.52.0
  | | `-- negotiator@0.6.3
  | +-+ array-flatten@1.1.1
  | +-+ body-parser@1.20.1
  | | +-+ bytes@3.1.2
  | | +-+ content-type@1.0.5 deduped
  | | +-+ debug@2.6.9 deduped
  | | +-+ depd@2.0.0 deduped
  | | +-+ destroy@1.2.0
  | | +-+ http-errors@2.0.0 deduped
  | | +-+ iconv-lite@0.4.24
  | | | `-- safer-buffer@2.1.2
  | | +-+ on-finished@2.4.1 deduped
  | | +-+ qs@6.11.0 deduped
  | | +-+ raw-body@2.5.1
  | | | +-+ bytes@3.1.2 deduped
  | | | +-+ http-errors@2.0.0 deduped
  | | | +-+ iconv-lite@0.4.24 deduped
  | | | | `-- unpipe@1.0.0 deduped
  | | | +-+ type-is@1.6.18 deduped
  | | | | `-- unpipe@1.0.0
  | | +-+ content-disposition@0.5.4
  | | | `-- safe-buffer@5.2.1 deduped
  | | +-+ content-type@1.0.5
  | | +-+ cookie-signature@1.0.6
  | | +-+ cookie@0.5.0
  | | +-+ debug@2.6.9
  | | | `-- ms@2.0.0
  | | +-+ depd@2.0.0
```

Coffee break (?)

Choose a Text Editor

Notepad++

SublimeText

Vi Notepad

Wordpad

IntelliJ Ultimate

WebStorm

VisualStudio Code

Eclipse

NetBeans

Choose (~~NOT~~ a Text Editor) a modern IDE

IntelliJ Ultimate



**Free for students
(register with esilv email)
else €€ or 30 days eval**

WebStorm



VisualStudio Code

Eclipse + plugins

Eclipse (plugins not pre-installed)

IntelliJ (No web js Support)

<https://www.jetbrains.com/idea/download>

The screenshot shows the official IntelliJ IDEA download page on the JetBrains website. At the top, there's a navigation bar with links for Developer Tools, Team Tools, Education, Solutions, Support, Store, and a search icon. Below the navigation is a main menu with IntelliJ IDEA, JetBrains IDEs, Coming in 2024.3, What's New, Features, Resources, Pricing, and a prominent blue Download button.

Below the main menu, there are links for Windows, macOS, and Linux. The Windows link is underlined, indicating it's the active platform. The page features a large "IntelliJ IDEA Ultimate" heading with a logo, followed by the subtext "The Leading Java and Kotlin IDE".

At the bottom left, there are two download buttons: a blue "Download" button and a ".exe (Windows)" dropdown menu showing "Windows". Below these buttons is a link for "Free 30-day trial".

On the right side of the page, there's a screenshot of the IntelliJ IDEA interface showing a project structure and some Java code in the editor. The code is related to a "PetClinicApplication" and includes annotations like @InitBinder, @ModelAttribute, and @GetMapping.

At the very bottom, there are links for "Version: 2024.2.1", "Build: 242.21829.142", and the date "29 August 2024". There are also links for "System requirements", "Other versions", "Installation instructions", and "Third-party software".

Pricing > Special Categories > Students



Developer Tools Team Tools Education Solutions Support Store

IntelliJ IDEA

JetBrains IDEs

Coming in 2024.3 What's New Features ▾ Resources

Pricing

Download

Subscription Options and Pricing



For Organizations



For Individual Use



Special Categories

For students and teachers

Students and academic staff members are eligible to use all JetBrains tools free, upon verification of their university/college domain email or ISIC card.

[Learn more](#)

For classroom assistance

Universities, colleges, schools, and non-commercial educational organizations are eligible for free licensing to install all JetBrains tools in classrooms and computer labs for educational purposes.

[Learn more](#)

For Open Source projects

Non-commercial open source projects can qualify for free licenses to all JetBrains tools if they meet the support program requirements.

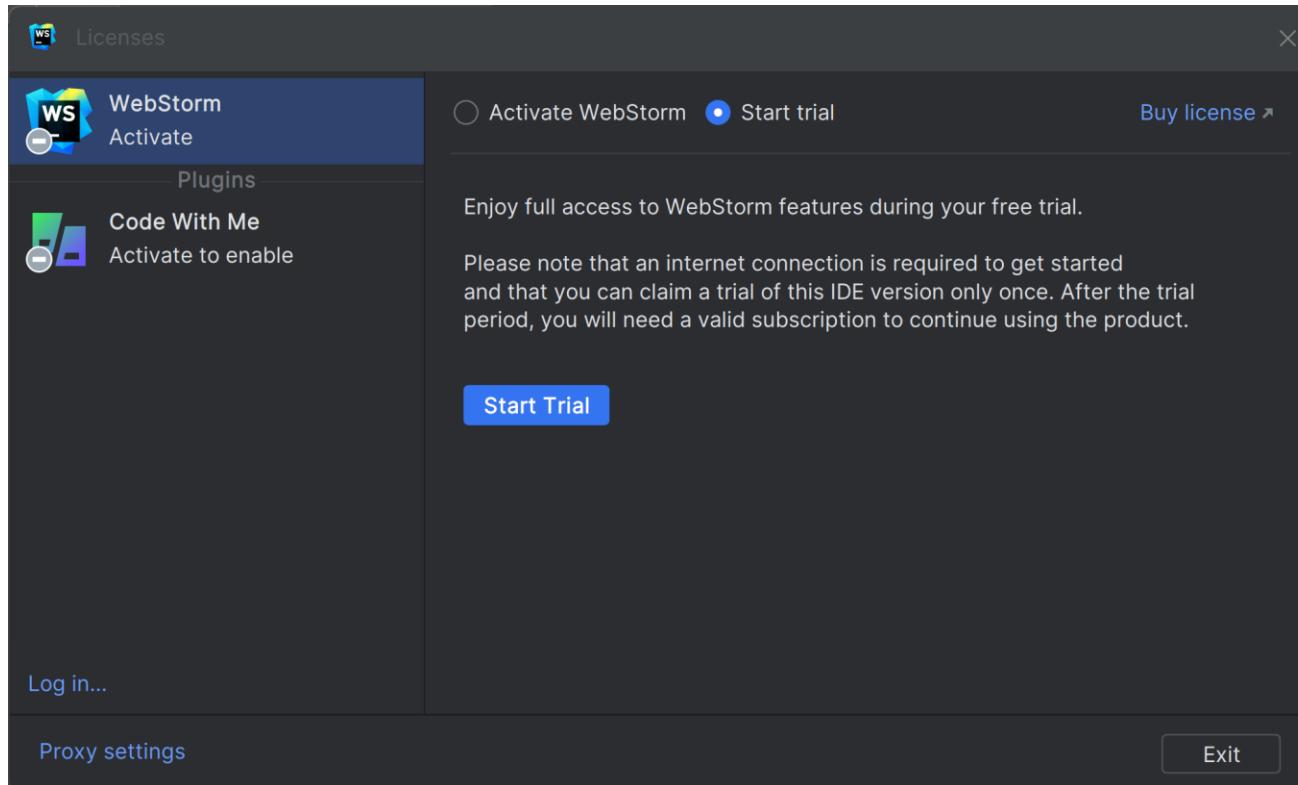
[Learn more](#)

Notice Licenses:

IntelliJ Ultimate = WebStorm + IntelliJ +

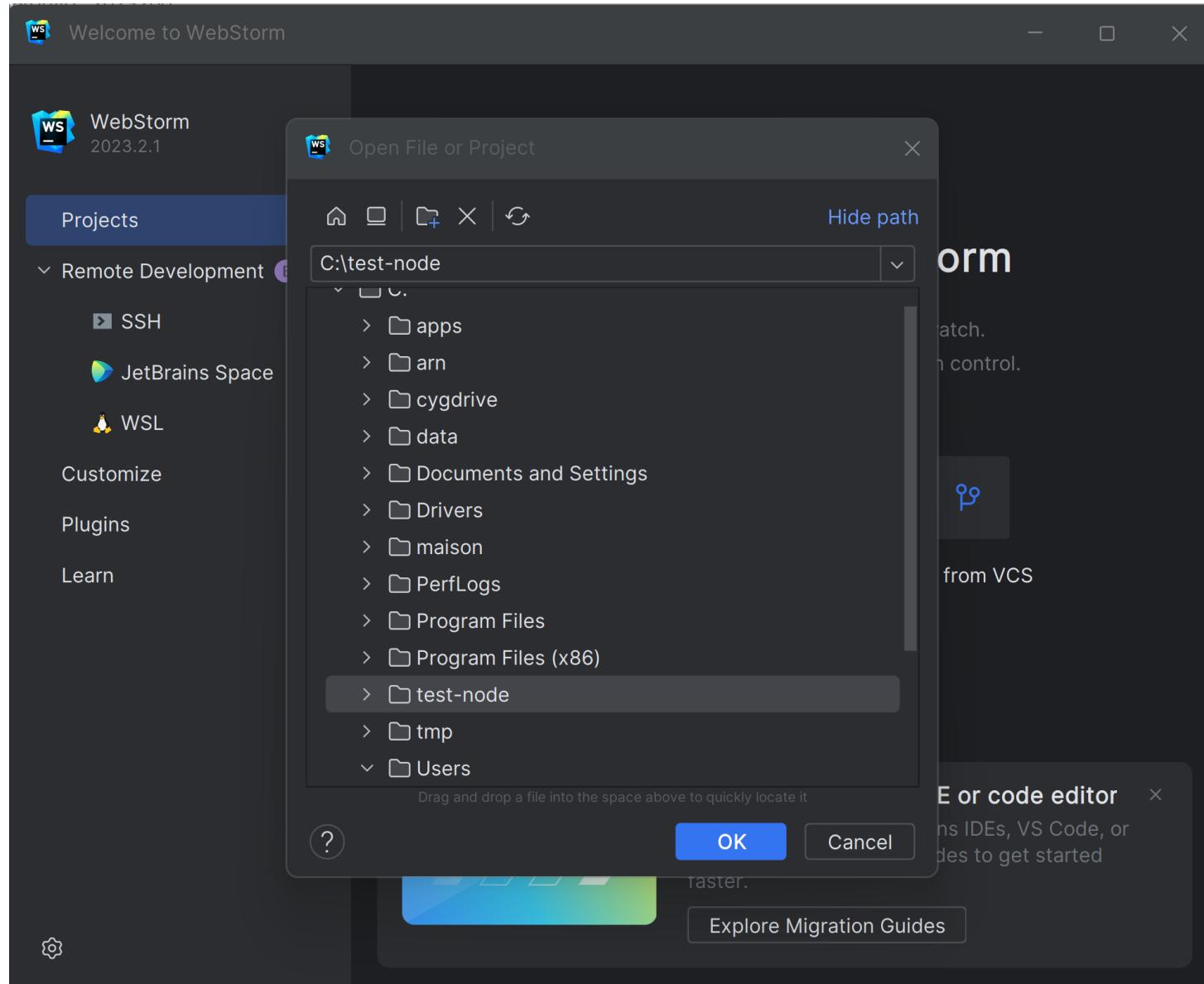
IntelliJ Community = Java only .. NO web !

WebStorm = Web Only .. !

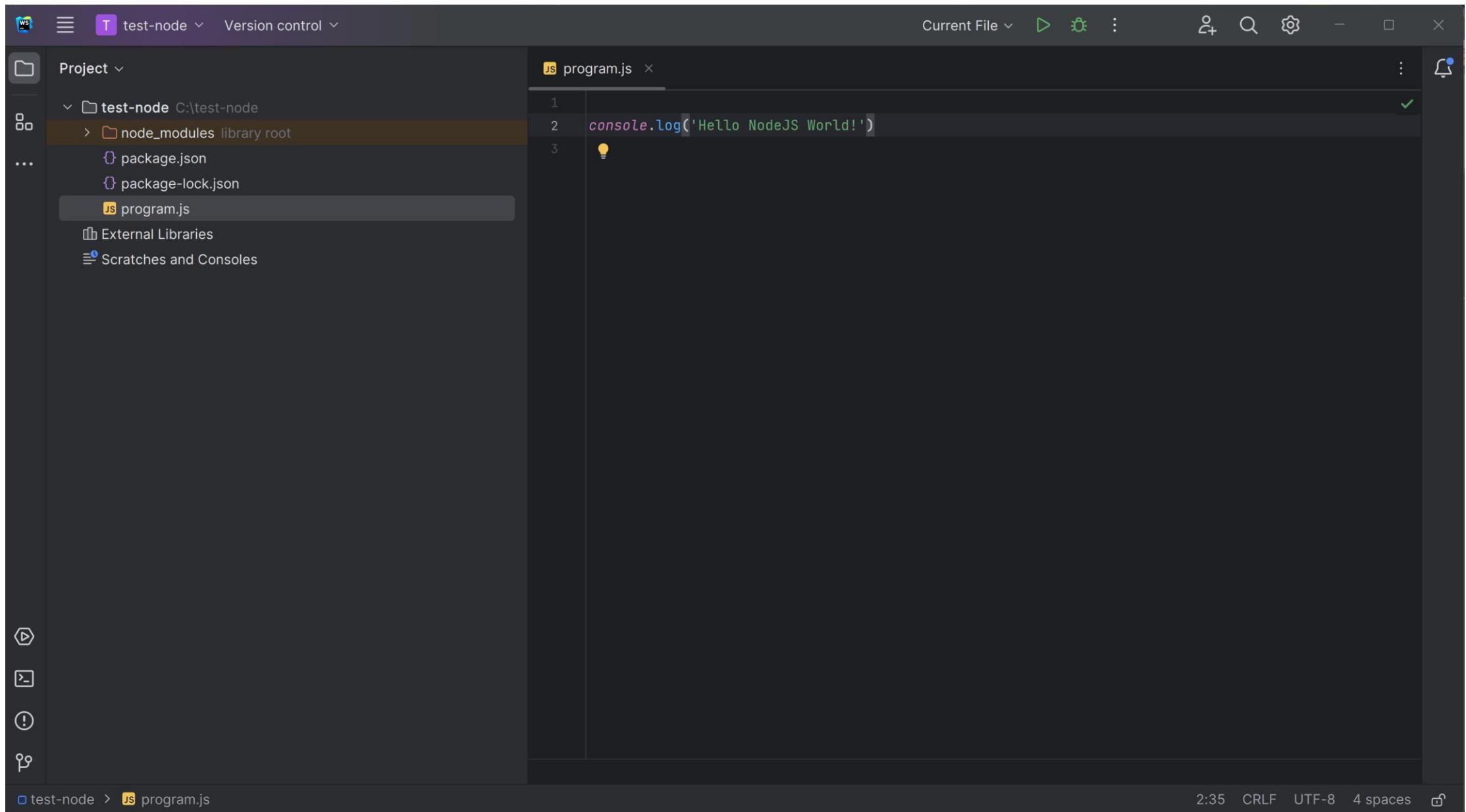


click on "Buy License" then fill esilv email
else click on "Start Trial" (30 days)

Open Project



Edit program.js File

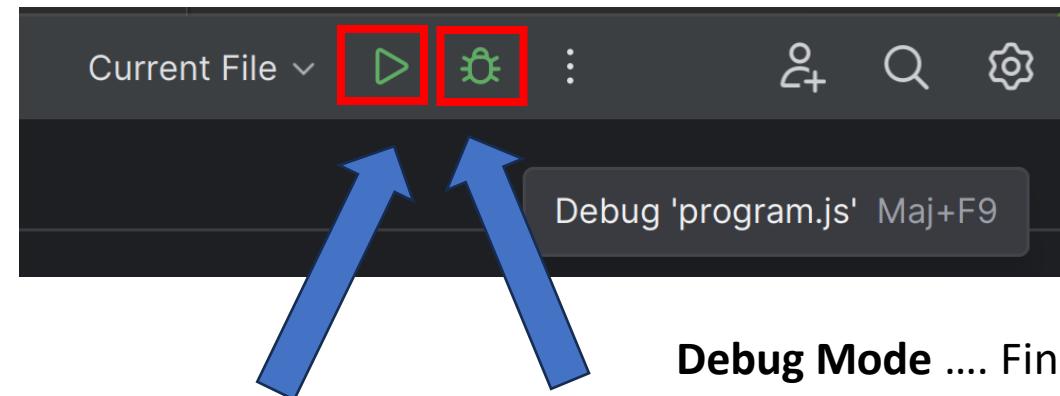


A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "test-node" and "Version control". The left sidebar displays a project structure for "test-node" at "C:\test-node", including "node_modules" (library root), "package.json", "package-lock.json", and "program.js" (which is currently selected). The main area shows the "program.js" file with the following content:

```
1
2 console.log('Hello NodeJS World!')
3
```

The code editor has several status icons: a green checkmark in the top right corner, a yellow lightbulb icon above line 3, and a small error icon next to the closing parenthesis of the log call. The bottom status bar shows "test-node > program.js", "2:35 CRLF UTF-8 4 spaces", and a file save icon.

Click on « Run » or « Debug » ?



Run Mode

Start and forget

Can NOT activate any breakpoint

... can only see logs, and Kill

Why do you use an IDE Debugger

Use directly « C:> node » ??

Debug Mode Find the « bug »

Can put breakpoints !

Google

bug

< All Images Videos News Books : More



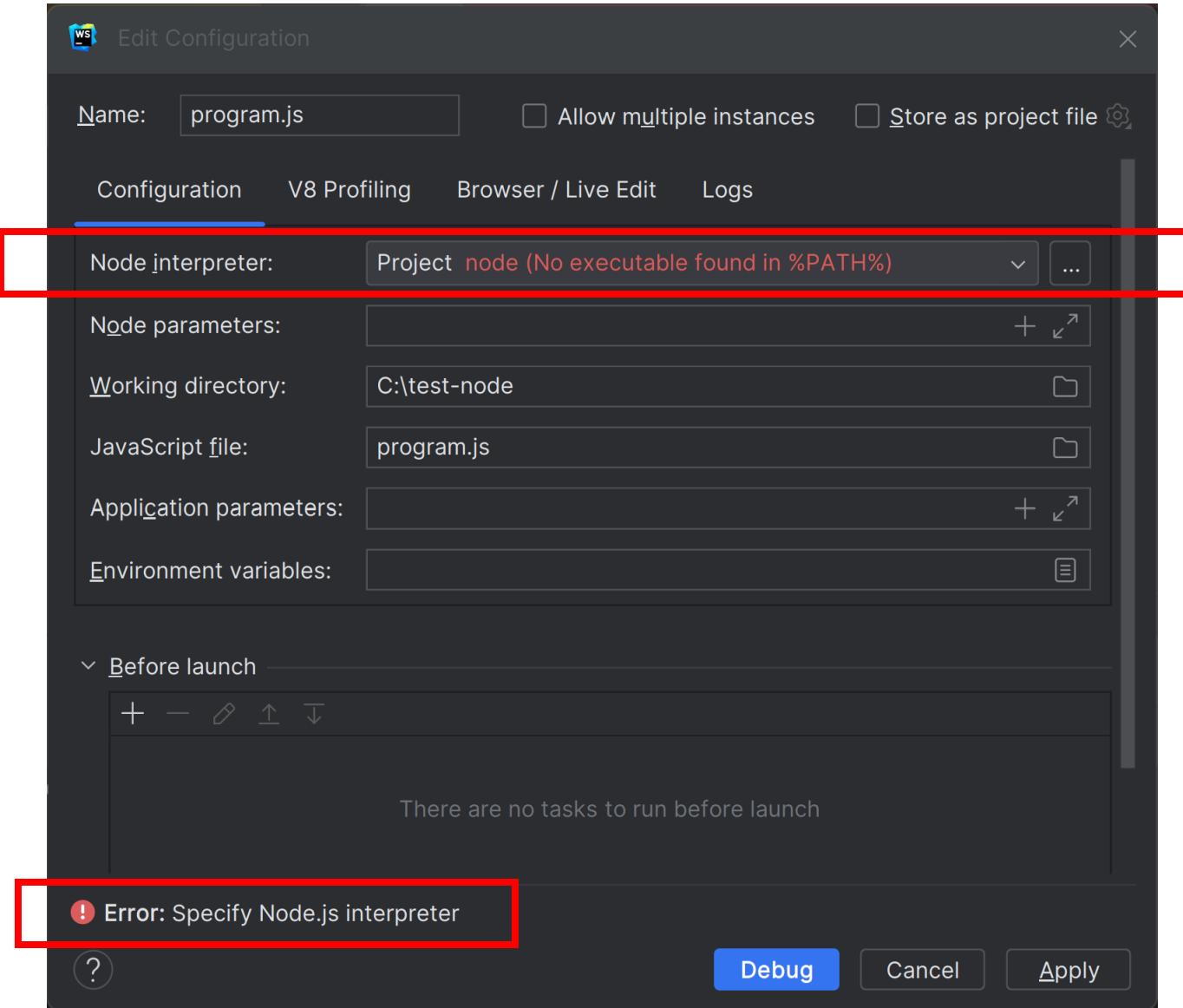
W Wikipedia
Coccinellidae - Wikipedia

W Wikipedia
Bed bug - Wikipedia

W Wikipedia
Scutelleridae - Wikipedia

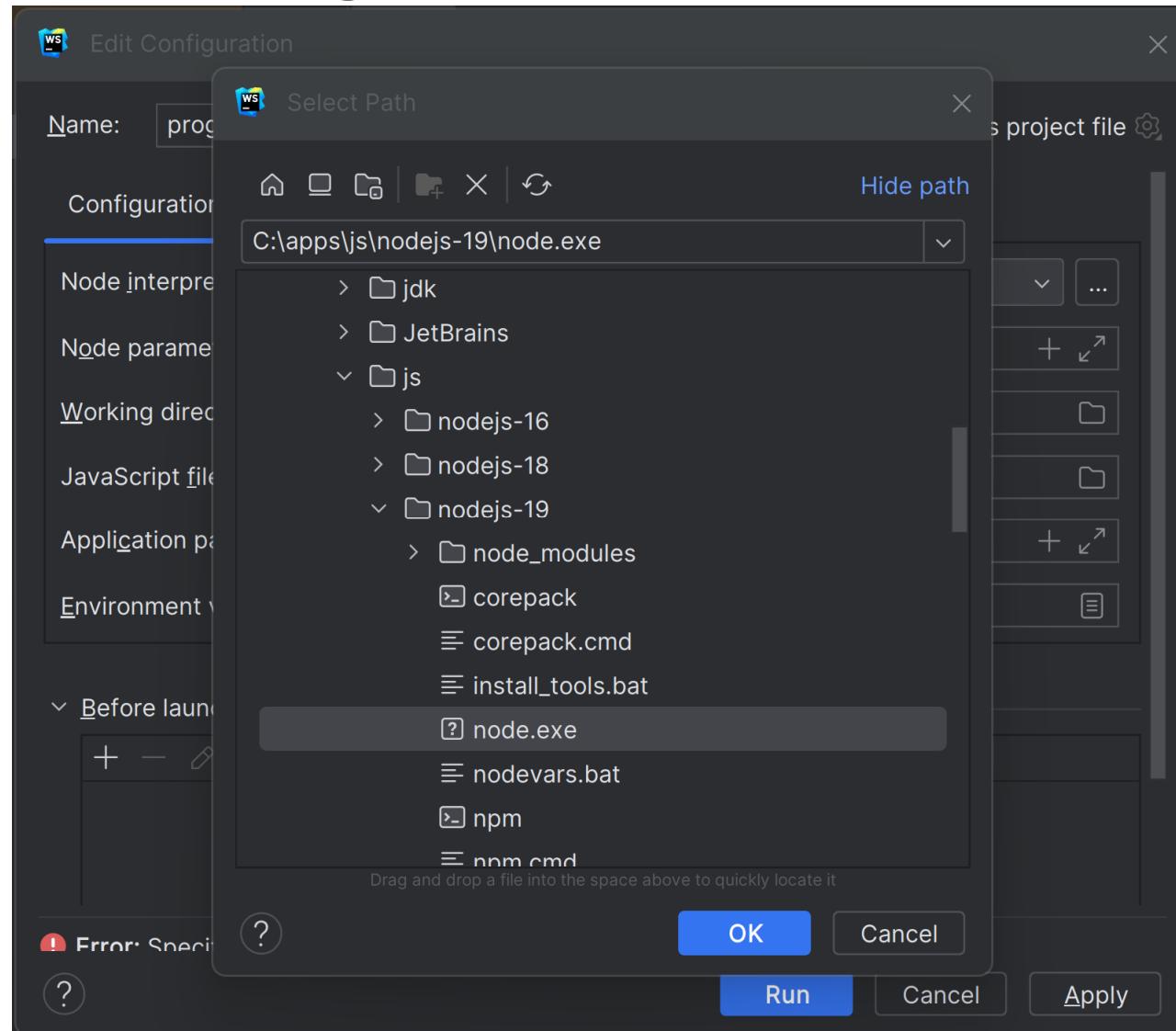
System Diagnostic

... Case undetected « node » not in %PATH%



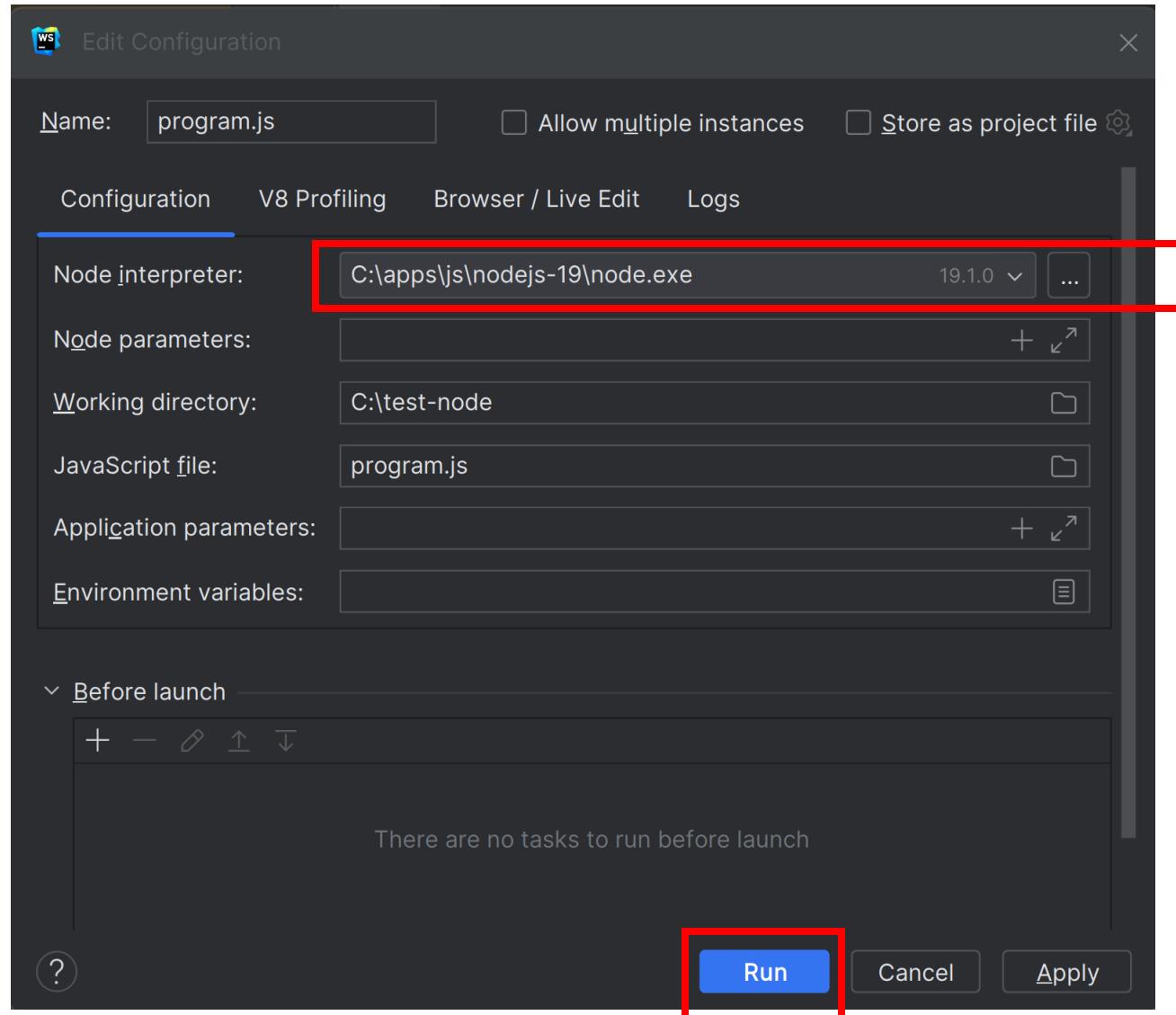
System Diagnostic

Configure ... Add node



System Diagnostic

Edit Run/Debug configuration...



« Run » ... Program Finished + console print

The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar displays a project structure for 'test-node' located at 'C:\test-node'. The 'node_modules' folder is highlighted as the 'library root'. Inside, there are files for 'package.json' and 'package-lock.json', and a script file named 'program.js'. The main editor area shows the contents of 'program.js':

```
1
2 console.log('Hello NodeJS World!')
3
```

Below the editor is the 'Run' view, which lists the currently selected task as 'program.js'. The terminal at the bottom shows the execution of the script and its output:

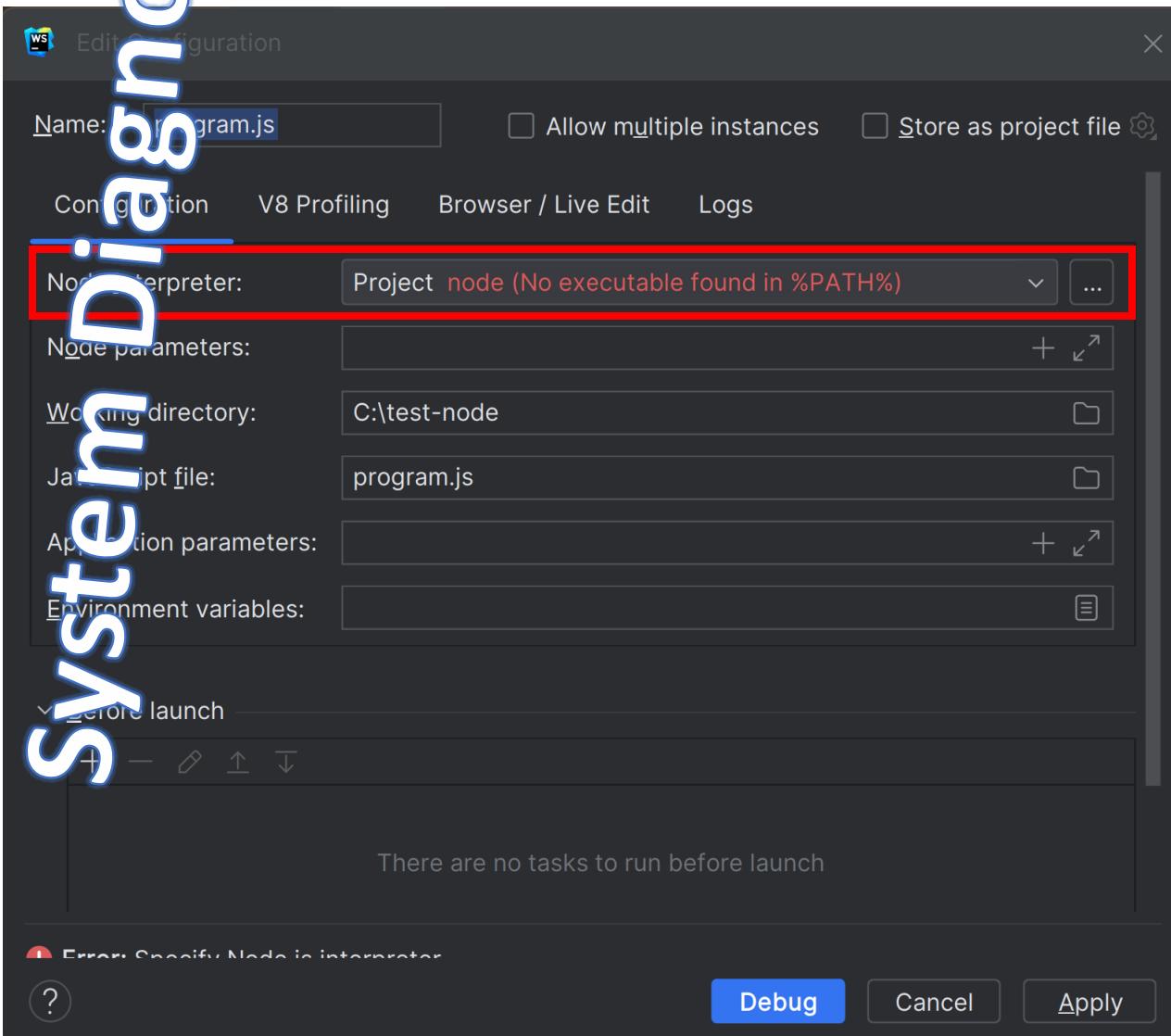
```
C:\apps\js\nodejs-19\node.exe C:\test-node\program.js
Hello NodeJS World!
Process finished with exit code 0
```

A red box highlights the terminal output, specifically the line 'Hello NodeJS World!', which corresponds to the console log statement in the code.

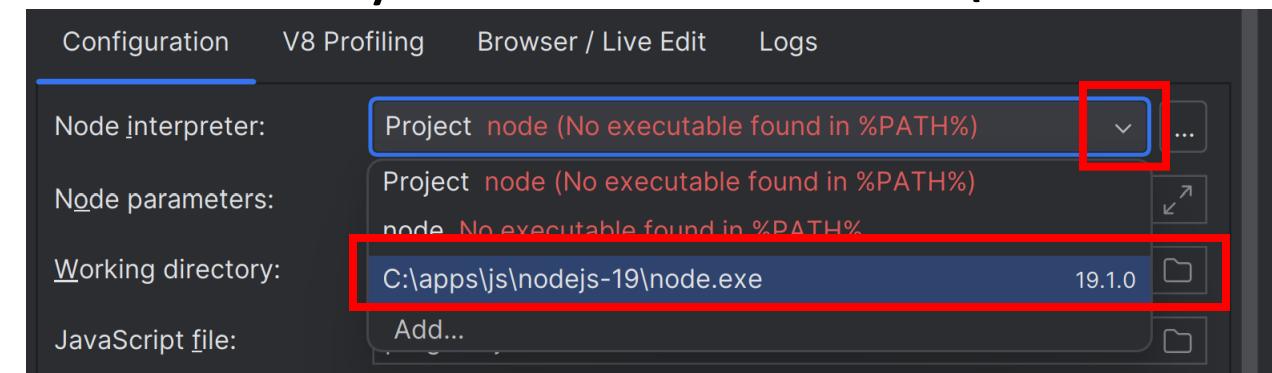
System Diagnostic

Redo « Run » or « Debug »

Need to re-edit node in %PATH% ???

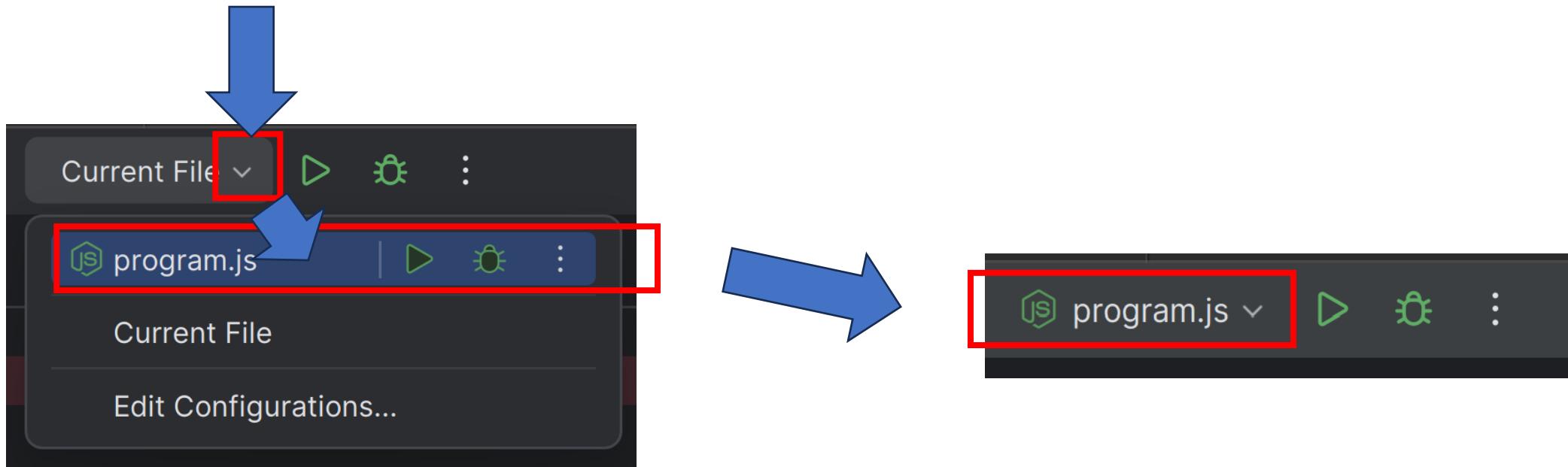


At least, can re-select
already added choice « C:\... » !

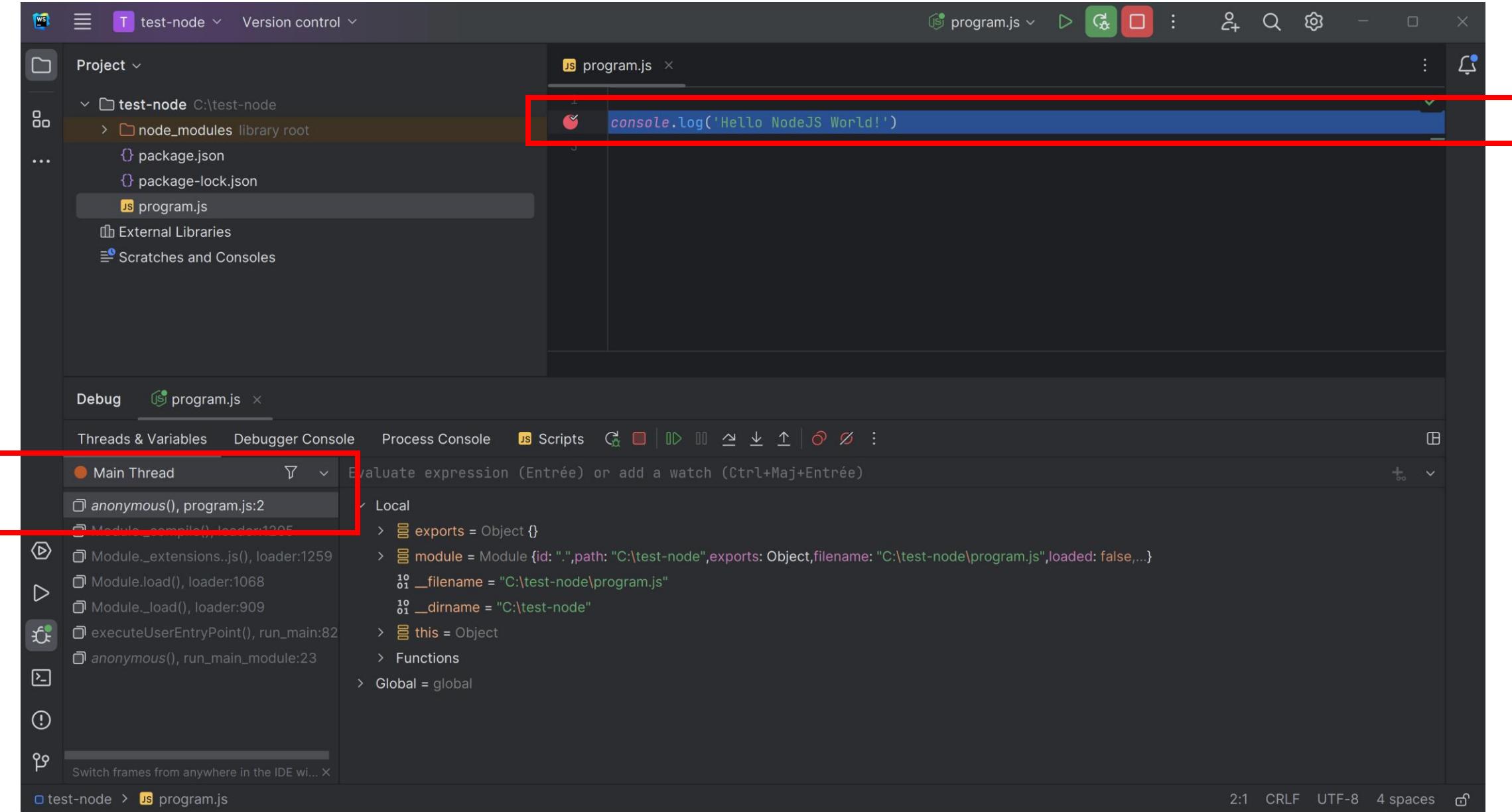


System Diagnostic

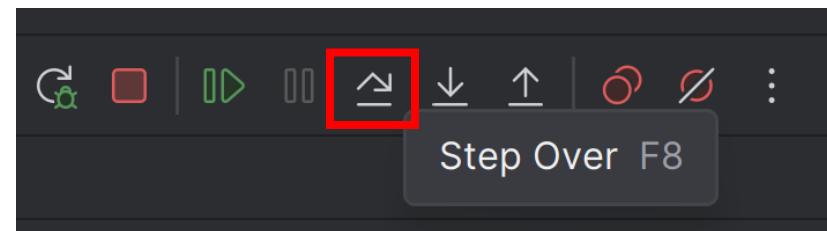
Change configuration « your-config »
instead of « Current File »



Debug « Mode » + Breakpoint => Program Paused

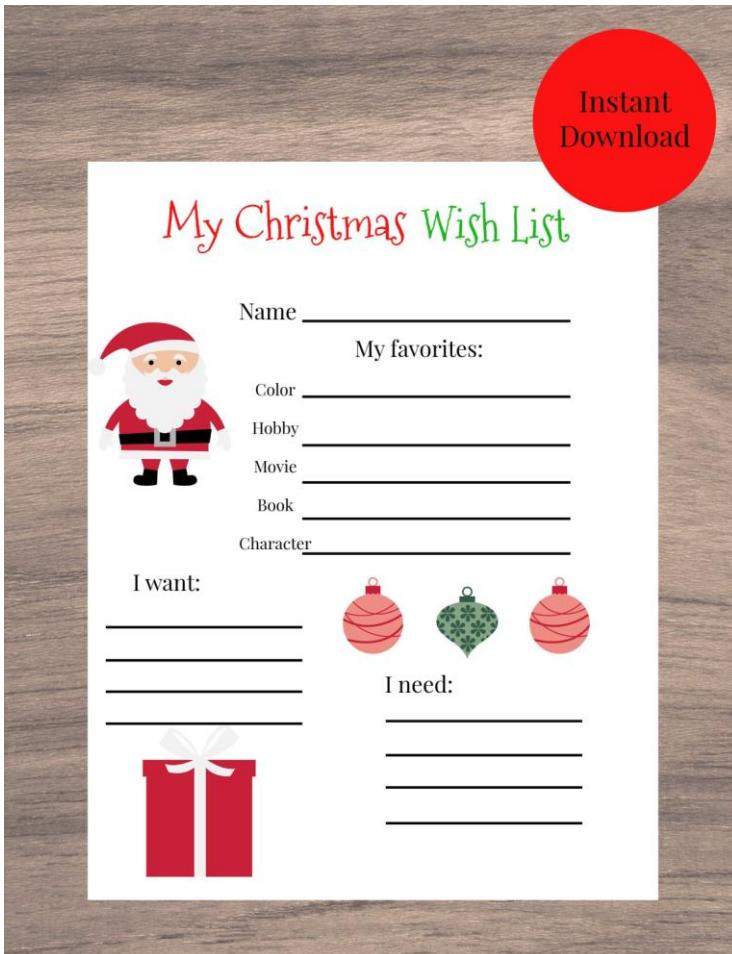


Step Into(F7)
< Step Over(F8)
< Step Out (Maj F8)
< Resume (F9)



Coffee Break

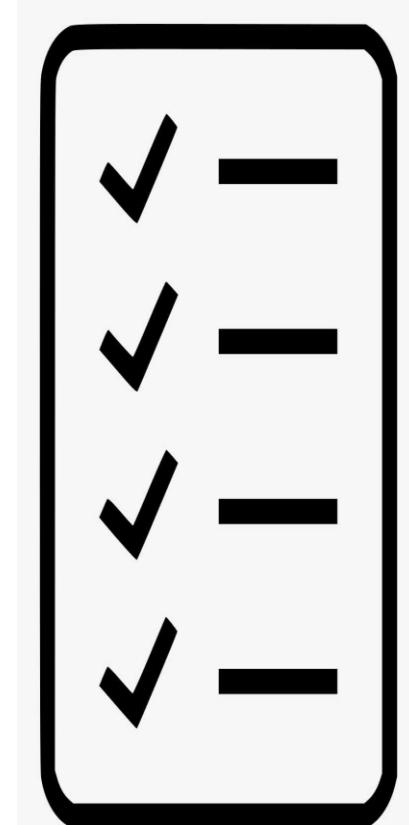
Complains on JavaScript ?



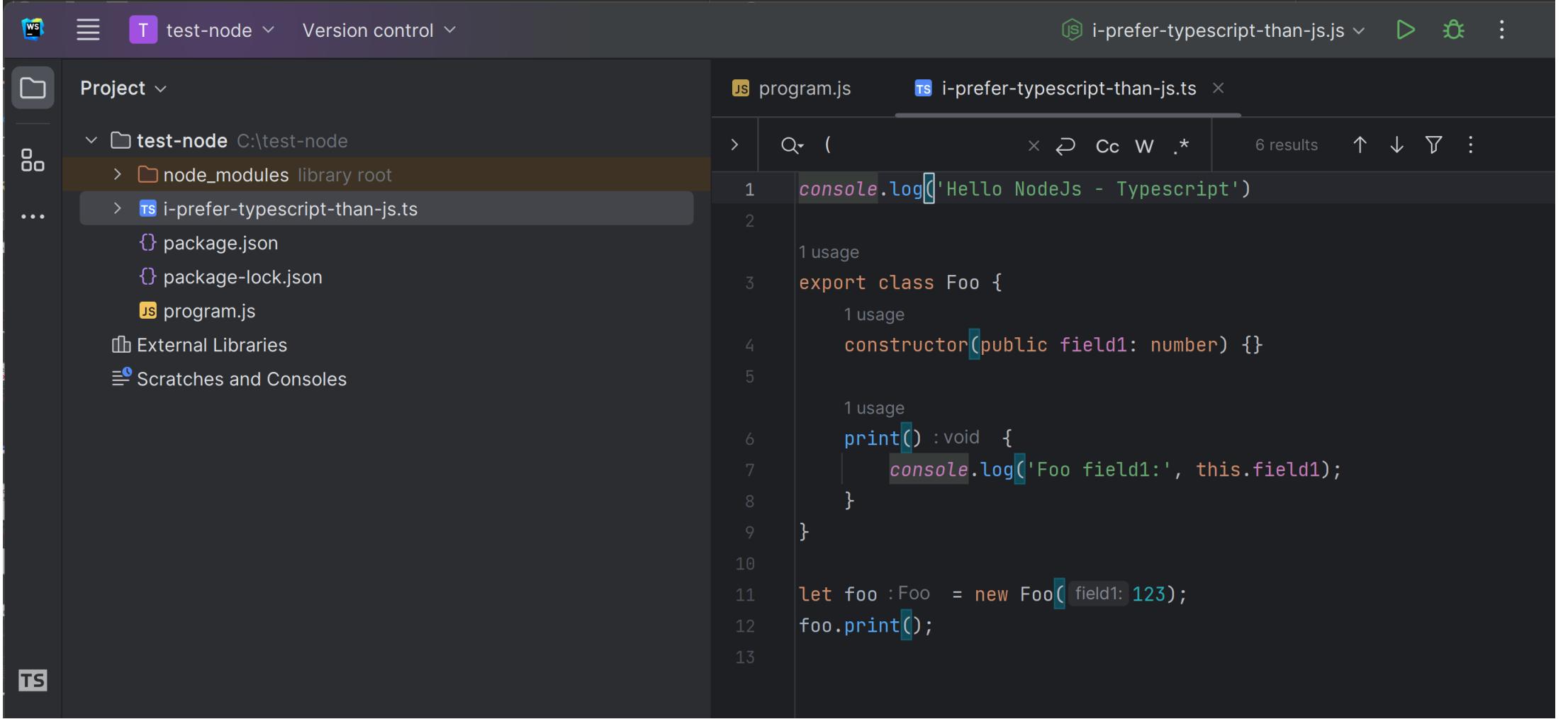
for Christmas Wish List, ask

- Types declarations
(standard types, structural types, union types)
- **Compiled-time Type checking**
- With Implicit types inference
- **Object-Oriented Classes**
- Templates
- Spread operators
- Syntactic sugars
- ..

TypeScript



Edit a TypeScript *.ts File

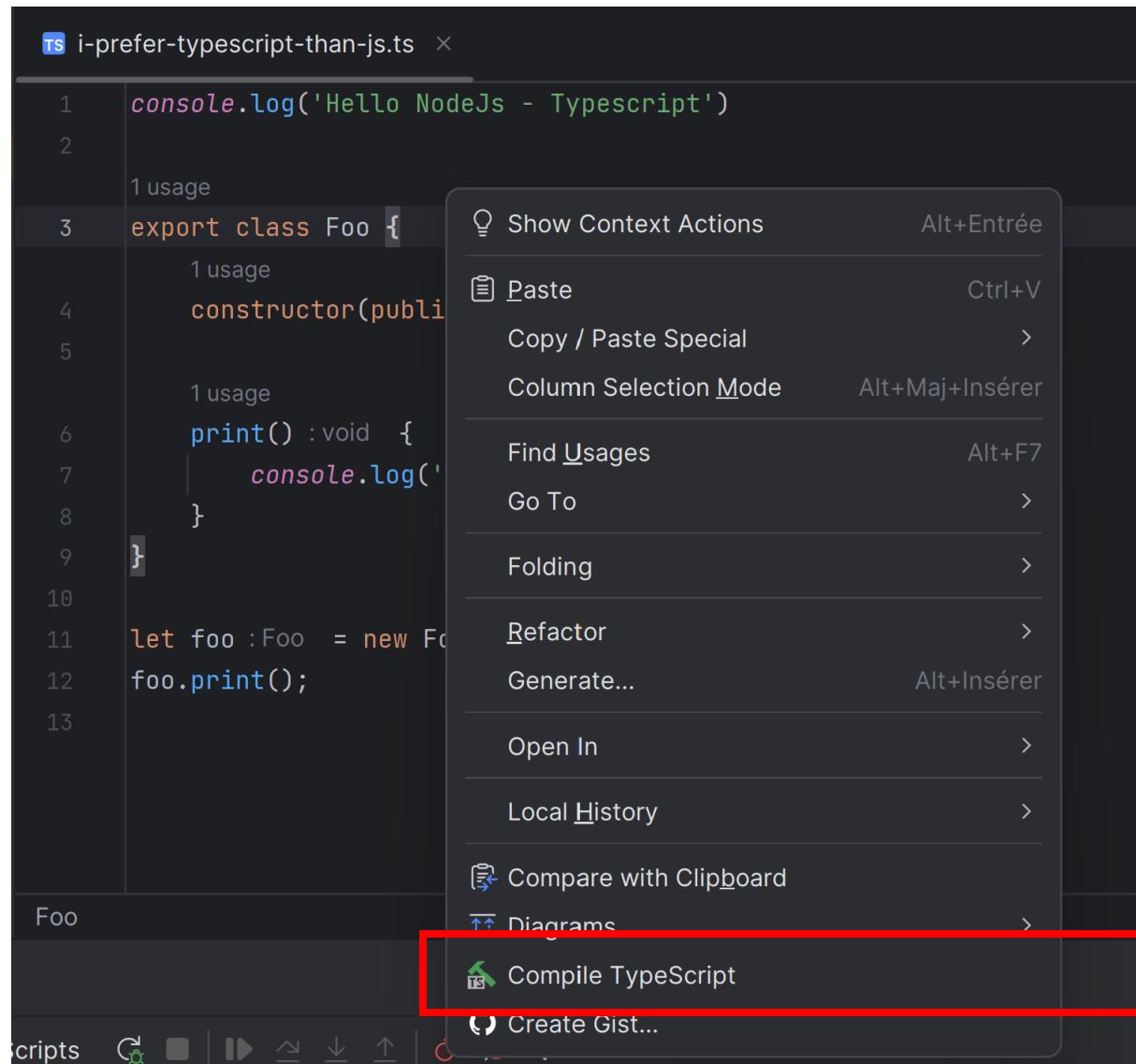


The screenshot shows the Visual Studio Code interface with the following details:

- Project Explorer:** Shows a project named "test-node" located at "C:\test-node". Inside the project, there is a "node_modules" folder (marked as "library root") and a file named "i-prefer-typescript-than-js.ts". Other files listed are "package.json", "package-lock.json", and "program.js".
- Search Results:** A search bar at the top right shows the query "console.log". Below it, a results panel displays 6 results. The results show code from the "i-prefer-typescript-than-js.ts" file:

```
1 console.log('Hello NodeJs - Typescript')
2
3 export class Foo {
4     constructor(public field1: number) {}
5
6     print(): void {
7         console.log('Foo field1:', this.field1);
8     }
9 }
10
11 let foo : Foo = new Foo( field1: 123);
12 foo.print();
```
- Status Bar:** At the bottom left, there is a small "TS" icon.

Right Click « Compile TypeScript »



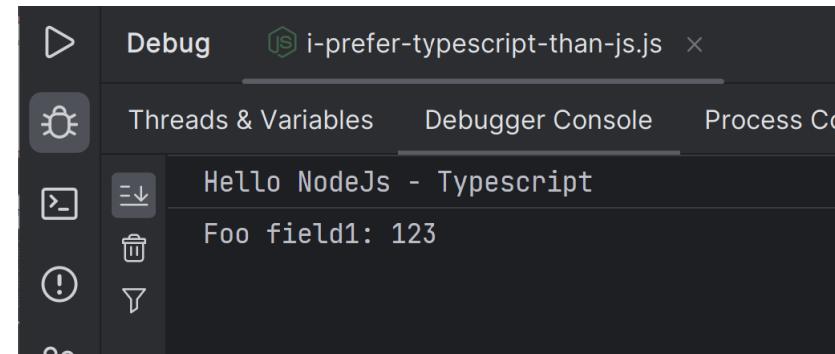
Open Item « .ts » > Sub Item « .js » see generated Js !!

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays a project structure under 'test-node'. A red box highlights the 'i-prefer-typescript-than-js.ts' file and its corresponding 'i-prefer-typescript-than-js.js' file, which is also highlighted in the list. On the right, the editor shows two tabs: 'i-prefer-typescript-than-js.ts' and 'i-prefer-typescript-than-js.js'. The 'i-prefer-typescript-than-js.ts' tab contains TypeScript code, and the 'i-prefer-typescript-than-js.js' tab contains the generated JavaScript code. A large red box highlights the generated JavaScript code, specifically the class definition and its methods.

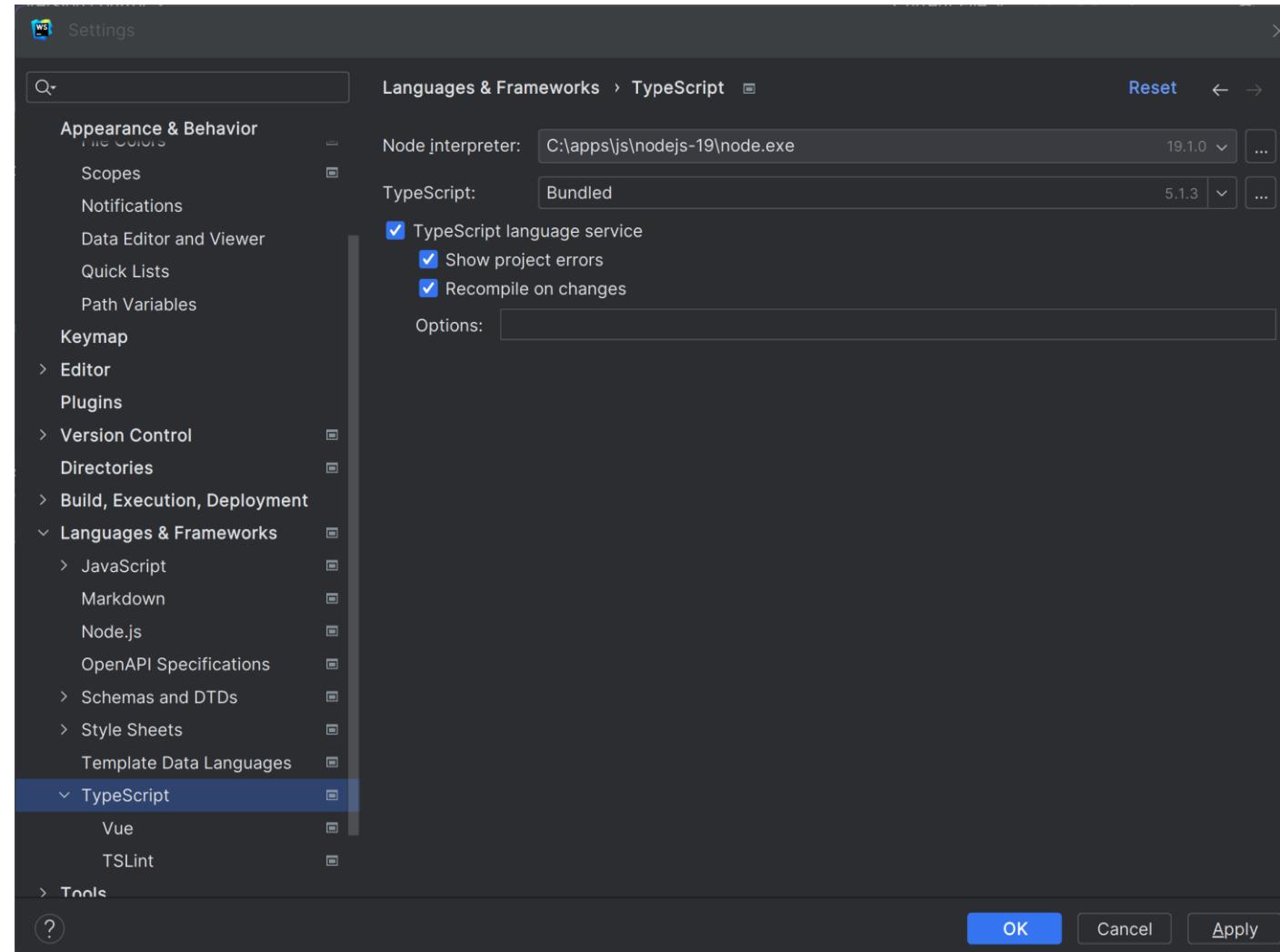
```
"use strict";
Object.defineProperty(exports, "__esModule", { value: true });
exports.Foo = void 0;
console.log('Hello NodeJs - Typescript');
var Foo = /** @class */ (function () {
    2 usages
    function Foo(field1) {
        this.field1 = field1;
    }
    Foo.prototype.print = function () {
        console.log('Foo field1:', this.field1);
    };
    return Foo;
}());
exports.Foo = Foo;
var foo = new Foo(123);
foo.print();
```

Debug Ts program ... (No Run/Debug/breakpoint in *.ts, but ok in *.js)

need « .map » file for mapping ts line X <-> js line Y



Edit Settings > TypeScript > Recompile on changes



Edit on Left *.ts ... see real time change in right *.js

The image shows a split-screen code editor with two panes. The left pane is titled "i-prefer-typescript-than-js.ts" and contains TypeScript code. The right pane is titled "i-prefer-typescript-than-js.js" and contains the corresponding generated JavaScript code. Both panes have dark themes with syntax highlighting.

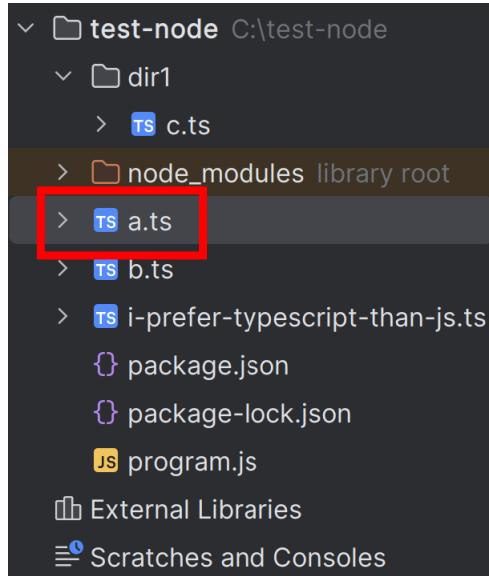
Left Pane (TypeScript):

```
ts i-prefer-typescript-than-js.ts
1 console.log('Hello NodeJs - Typescript')
2   1 usage
3 export class Foo {
4     field2: number = 1;
5
6     1 usage
7     constructor(public field1: number) {}
8
9     1 usage
10    print(): void {
11      console.log('Foo field1:', this.field1);
12      console.log('Foo field2:', this.field2);
13      console.log('Foo fieldxxxx:', this.fieldxxxx);
14    }
15
16    let foo : Foo = new Foo({ field1: 123 });
17    foo.print();
```

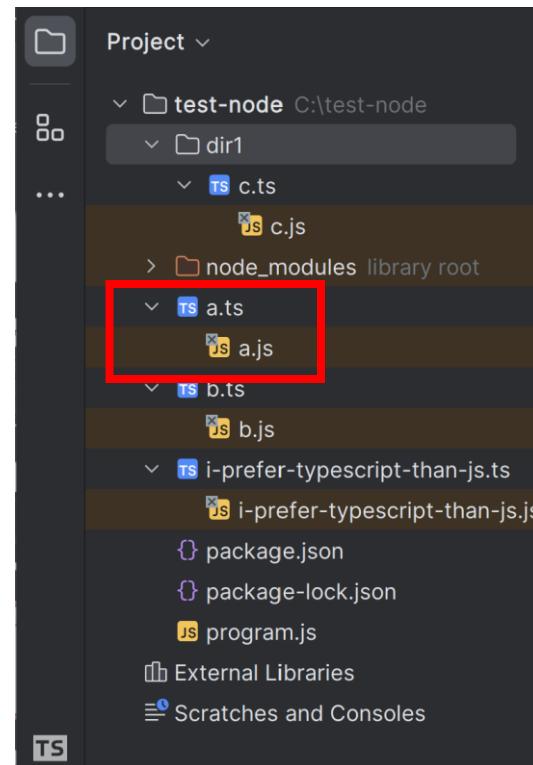
Right Pane (JavaScript):

```
js i-prefer-typescript-than-js.js
1 "use strict";
2 Object.defineProperty(exports, "__esModule", { value: true });
3 exports.Foo = void 0;
4 console.log('Hello NodeJs - Typescript');
5 var Foo = /** @class */ (function () {
6   2 usages
7   function Foo(field1) {
8     this.field1 = field1;
9     this.field2 = 1;
10   }
11   Foo.prototype.print = function () {
12     console.log('Foo field1:', this.field1);
13     console.log('Foo field2:', this.field2);
14     console.log('Foo fieldxxxx:', this.fieldxxxx);
15   };
16   return Foo;
17 }());
18 exports.Foo = Foo;
19 var foo = new Foo(123);
20 foo.print();
```

Edit many **/*.ts ... see corresponding **/*.js



Expand
views
→



Check
on FileSystem
→

Nom	Modifié le	Type	Taille
.idea	31/08/2023 23:29	Dossier de fichiers	
dir1	31/08/2023 23:30	Dossier de fichiers	
node_modules	31/08/2023 20:34	Dossier de fichiers	
a.js	31/08/2023 23:30	Fichier de JavaScript	0 Ko
a.ts	31/08/2023 23:30	Fichier TS	0 Ko
b.js	31/08/2023 23:30	Fichier de JavaScript	0 Ko
b.ts	31/08/2023 23:30	Fichier TS	0 Ko
i-prefer-typescript-than-js.js	31/08/2023 23:29	Fichier de JavaScript	1 Ko
i-prefer-typescript-than-js.ts	31/08/2023 23:28	Fichier TS	1 Ko
package.json	31/08/2023 20:33	Fichier JSON	1 Ko
package-lock.json	31/08/2023 20:51	Fichier JSON	22 Ko
program.js	31/08/2023 20:12	Fichier de JavaScript	1 Ko

tsc = TypeScript Compiler

... same as in IDE, but in batch mode, for CI-CD

tsc is internally used by WebStorm

... / IntelliJ / VisualStudio Code / Eclipse !!

tsc even generalised it as « language server »

=> You have exact same compilation in all IDEs !!

<https://www.typescriptlang.org/docs/handbook/typescript-tooling-in-5-minutes.html>

The screenshot shows the TypeScript handbook page. The header includes the TypeScript logo and navigation links for Download, Docs, Handbook, Community, Playground, and Tools. A sidebar on the left under 'Get Started' contains links for TS for the New Programmer, TypeScript for JS Programmers, TS for Java/C# Programmers, TS for Functional Programmers, and 'TypeScript Tooling in 5 minutes', which is highlighted with a blue border. The main content area features a large heading 'TypeScript Tooling in 5 minutes'. Below it, a paragraph reads 'Let's get started by building a simple web application with TypeScript.' A section titled 'Installing TypeScript' follows, with a note about two installation methods: via npm or Visual Studio plugins. It also mentions that Visual Studio 2017 and 2015 Update 3 support TypeScript by default and provides a command-line example for npm users.

TypeScript Tooling in 5 minutes

Let's get started by building a simple web application with TypeScript.

Installing TypeScript

There are two main ways to add TypeScript to your project:

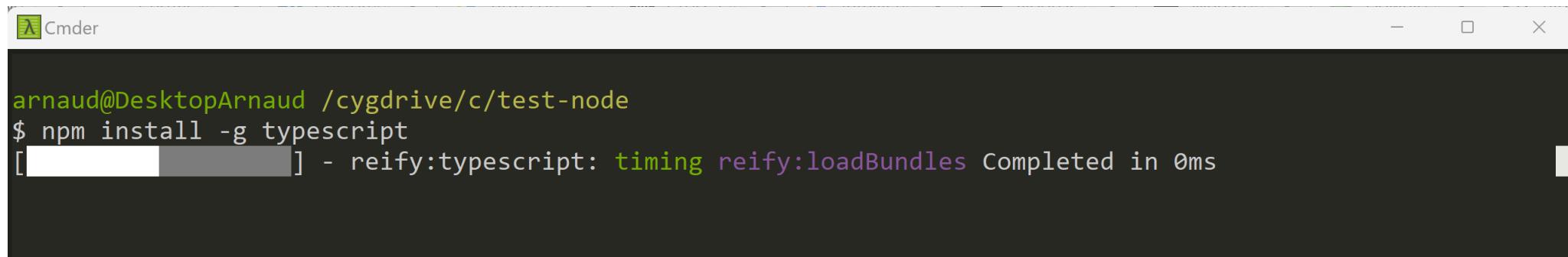
- Via npm (the Node.js package manager)
- By installing TypeScript's Visual Studio plugins

Visual Studio 2017 and Visual Studio 2015 Update 3 include TypeScript language support by default but does not include the TypeScript compiler, `tsc`. If you didn't install TypeScript with Visual Studio, you can still [download it](#).

For npm users:

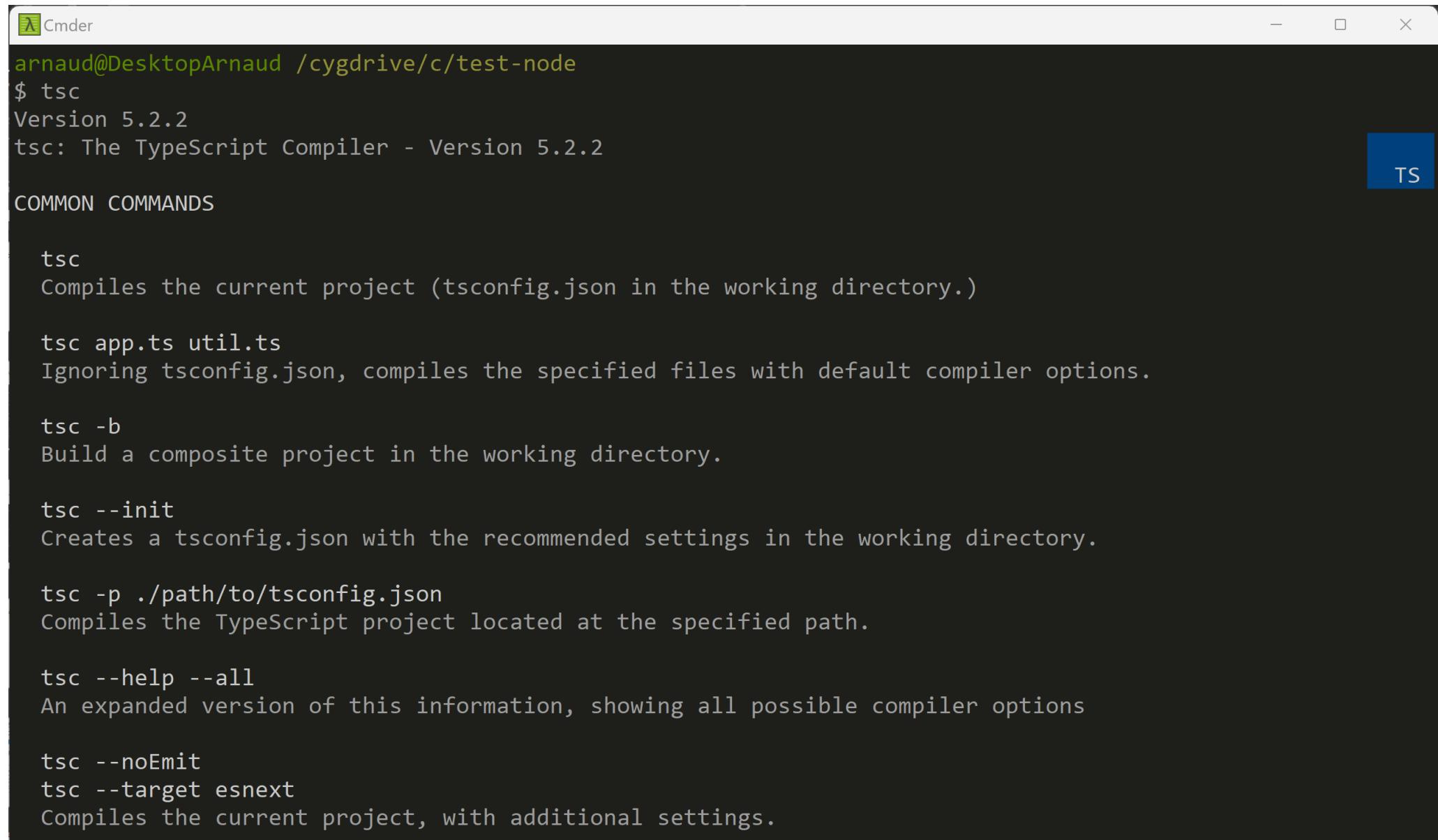
```
> npm install -g typescript
```

\$ npm install -g typescript
(« -g » for « --global » ... add tsc in node %PATH%)



```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm install -g typescript
[██████████] - reify:typescript: timing reify:loadBundles Completed in 0ms
```

\$ tsc



A screenshot of a terminal window titled "Cmder". The window shows the command \$ tsc followed by the TypeScript compiler's help output. The output includes common commands like tsc, tsc -b, and tsc --init, along with descriptions of what each command does. A blue "TS" logo is visible in the top right corner of the terminal window.

```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ tsc
Version 5.2.2
tsc: The TypeScript Compiler - Version 5.2.2

COMMON COMMANDS

tsc
Compiles the current project (tsconfig.json in the working directory.)

tsc app.ts util.ts
Ignoring tsconfig.json, compiles the specified files with default compiler options.

tsc -b
Build a composite project in the working directory.

tsc --init
Creates a tsconfig.json with the recommended settings in the working directory.

tsc -p ./path/to/tsconfig.json
Compiles the TypeScript project located at the specified path.

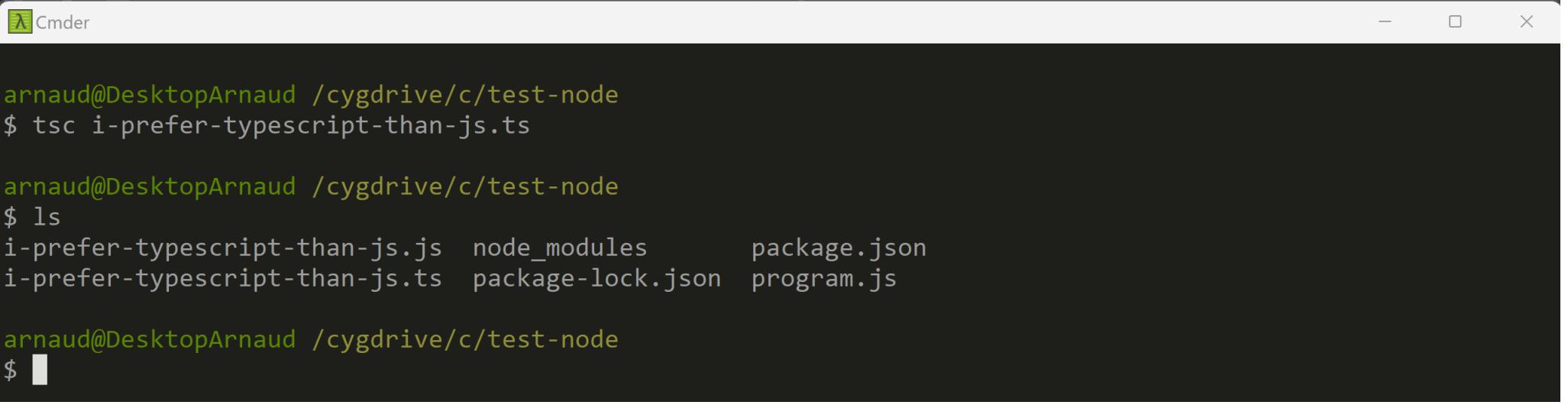
tsc --help --all
An expanded version of this information, showing all possible compiler options

tsc --noEmit
tsc --target esnext
Compiles the current project, with additional settings.
```

3 uses cases ...

- 1/ compile 1 single .ts file
- 2/ compile **/*.ts in project
- 3/ « -w » real-time watch to recompile on change

\$ tsc single-file.ts



The screenshot shows a terminal window titled "Cmder". The command \$ tsc single-file.ts is run, followed by an ls command to show the resulting files. The terminal window has a dark background with light-colored text.

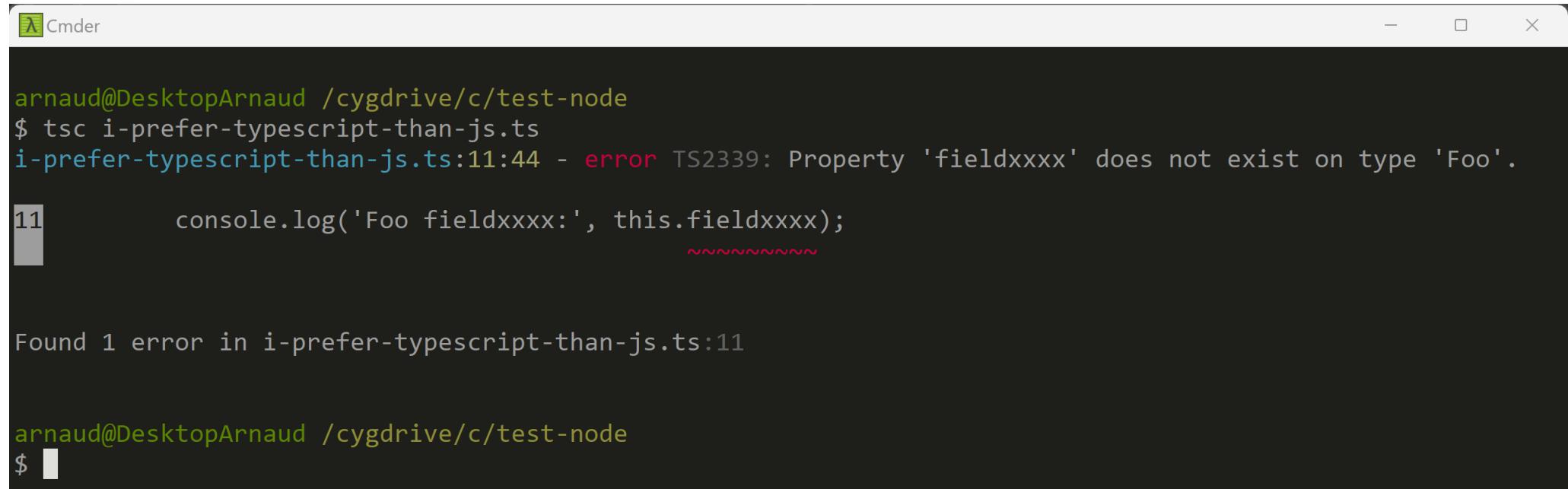
```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ tsc i-prefer-typescript-than-js.ts

arnaud@DesktopArnaud /cygdrive/c/test-node
$ ls
i-prefer-typescript-than-js.js  node_modules      package.json
i-prefer-typescript-than-js.ts   package-lock.json  program.js

arnaud@DesktopArnaud /cygdrive/c/test-node
$ █
```

Tsc Compiles ... find ERRORS messages at compile-time

Compare... remember typo found at runtime ?
<https://my-broken-web-site-deployed-in-PROD.html>



The screenshot shows a terminal window titled 'Cmder' with a black background and white text. It displays the following command and its output:

```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ tsc i-prefer-typescript-than-js.ts
i-prefer-typescript-than-js.ts:11:44 - error TS2339: Property 'fieldxxxx' does not exist on type 'Foo'.
11     console.log('Foo fieldxxxx:', this.fieldxxxx);
                         ~~~~~~
Found 1 error in i-prefer-typescript-than-js.ts:11

arnaud@DesktopArnaud /cygdrive/c/test-node
$
```

The terminal shows a TypeScript compilation error where the property 'fieldxxxx' is missing from the type 'Foo'. The error message is highlighted in red. The file being compiled is 'i-prefer-typescript-than-js.ts' at line 11, column 44. The terminal also indicates that one error was found in the file.

Setup a Typescript project

\$ tsc -init

```
Cmder
arnaud@DesktopArnaud /cygdrive/c/web/td2
$ tsc -init

Created a new tsconfig.json with:

target: es2016
module: commonjs
strict: true
esModuleInterop: true
skipLibCheck: true
forceConsistentCasingInFileNames: true

You can learn more at https://aka.ms/tsconfig

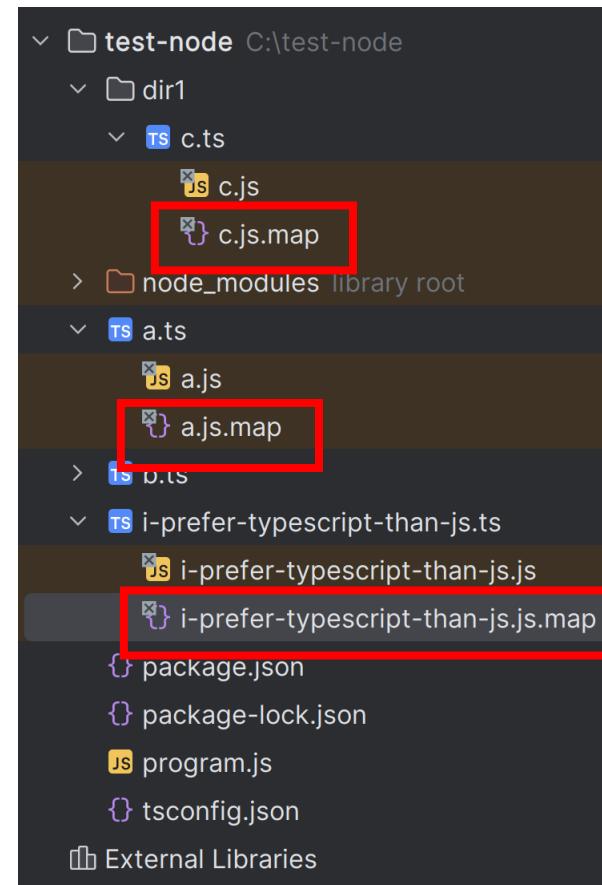
arnaud@DesktopArnaud /cygdrive/c/web/td2
$ ls
tsconfig.json
```

Default tsconfig.json

```
{}
{
  "compilerOptions": {
    "module": "commonjs",
    "target": "es5",
    "sourceMap": true
  },
  "exclude": [
    "node_modules"
  ]
}
```

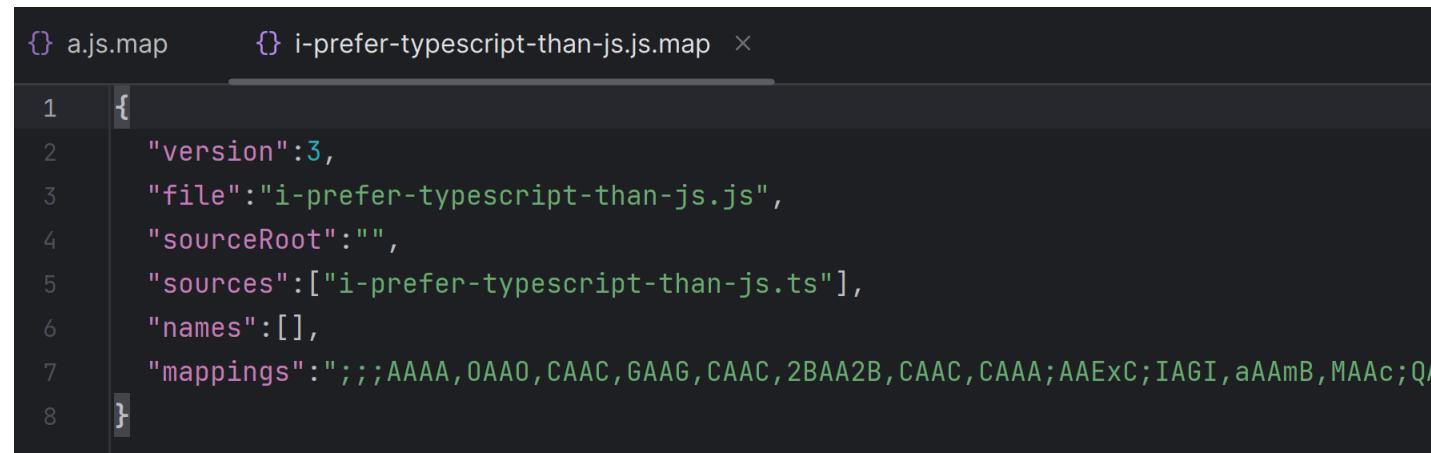
Notice sourceMap:true in tsconfig.json ... and **/*.js.map files generated

```
{} tsconfig.json ×  
1  {  
2      "compilerOptions": {  
3          "module": "commonjs",  
4          "target": "es5",  
5          "sourceMap": true  
6      },  
7      "exclude": [  
8          "node_modules"  
9      ]  
10 }
```



.map file
... = « lineNumber » debugging info
between source « *.ts » and generated « *.js »

Analogy with java:
between source « *.java » and generated « *.class »



The screenshot shows a code editor interface with two tabs at the top: 'a.js.map' and 'i-prefer-typescript-than-js.js.map'. The 'i-prefer-typescript-than-js.js.map' tab is active, displaying the following JSON content:

```
{  
  "version": 3,  
  "file": "i-prefer-typescript-than-js.js",  
  "sourceRoot": "",  
  "sources": ["i-prefer-typescript-than-js.ts"],  
  "names": [],  
  "mappings": ";;;AAAA,OAAO,CAAC,GAAG,CAAC,2BAA2B,CAAC,CAA;AAExC;IAGI,aAAmB,MAAc;Q/  
}
```

Remark and Analogy

JavaScript is the « low-level » assembler language (~x64) on the Web
Supported by all « web browsers » AND « node » (~intel/amd processors)



No developer code anymore in assembly language directly
(they prefer high-level C, C++, Java, C#, Python, ...)



Web developer prefers TypeScript rather than Js directly ..

Remark of The Remark ... WebAssembly

JavaScript is such an **horrible monster langage**
(with **eval** and strange permissive things)

W3c standard « WebAssembly » use only a SUBSET of JS
... that can be really statically typed / analysed / understood
then transformed to native x64 code by « compiler »

« WebAssembly » is trully a fast assembly langage
almost as fast as optimised « normal » code

.map Goodies ... can put breakpoint on *.ts (still debugging *.js for launch)

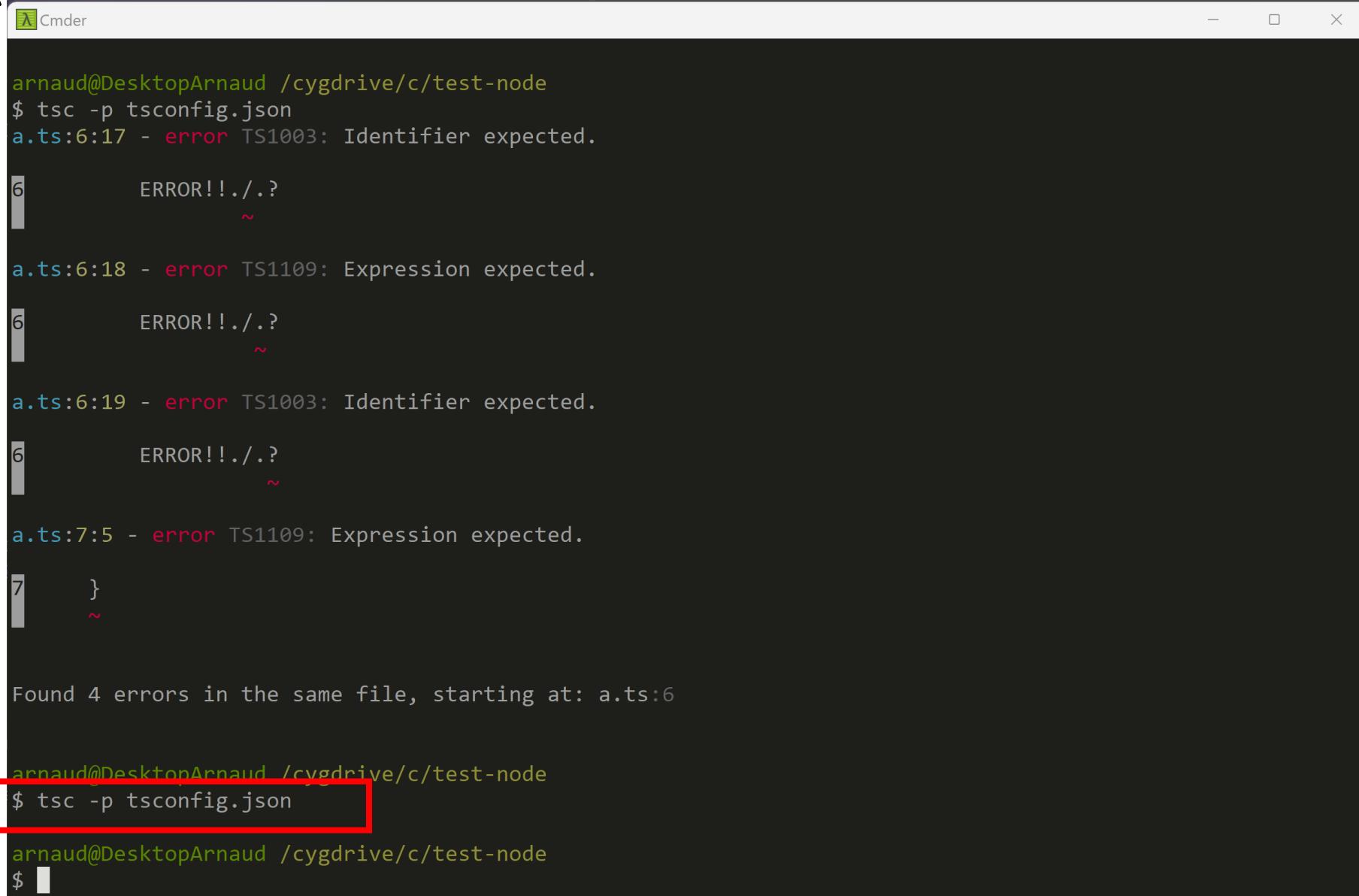
The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Explorer:** Shows a folder structure for "test-node". Inside "test-node", there are "dir1" and "node_modules" (marked as "library root"). Other files include "a.ts", "a.js", "a.js.map", "b.ts", "i-prefer-typescript-than-js.ts", "i-prefer-typescript-than-js.js", "package.json", "package-lock.json", "program.js", "tsconfig.json", and "External Libraries".
- Editor:** The "i-prefer-typescript-than-js.ts" file is open. A red circular breakpoint icon is placed on the line containing the "print()" method definition.
- Terminal:** The terminal shows the command "Foo > print()".
- Debug Bar:** The "Debug" tab is selected, showing the configuration "i-prefer-typescript-than-js.js".
- Threads & Variables:** The "Main Thread" is selected. The "Local" scope shows variables: "this.field1 = 123" and "field2 = 1".
- Status Bar:** The status bar at the bottom right indicates "TypeScript 5.2.2 9:9 CRLF UTF-8 4 spaces".

```
1 console.log('Hello NodeJs - Typescript')
2
3 export class Foo {
4     field2: number = 1;    field2: 1
5
6     constructor(public field1: number) {}  this.field1: 123
7
8     print(): void {
9         console.log('Foo field1:', this.field1);  this.field1: 123
10        console.log('Foo field2:', this.field2);  field2: 1
11    }
12
13
14 let foo : Foo  = new Foo( field1: 123);
15 foo.print();
```

\$ tsc
(or equivalent : \$ tsc -p tsconfig.json)

Case when error...



The screenshot shows a terminal window titled "Cmder" running on a Windows system. The command \$ tsc -p tsconfig.json was run, resulting in four errors in the file a.ts. The errors are:

- a.ts:6:17 - error TS1003: Identifier expected.
- a.ts:6:18 - error TS1109: Expression expected.
- a.ts:6:19 - error TS1003: Identifier expected.
- a.ts:7:5 - error TS1109: Expression expected.

Line 6 contains three consecutive "ERROR! !./.?". Line 7 contains a closing brace "}" followed by a tilde (~). A message at the bottom states: "Found 4 errors in the same file, starting at: a.ts:6".

arnaud@DesktopArnaud /cygdrive/c/test-node
\$ tsc -p tsconfig.json

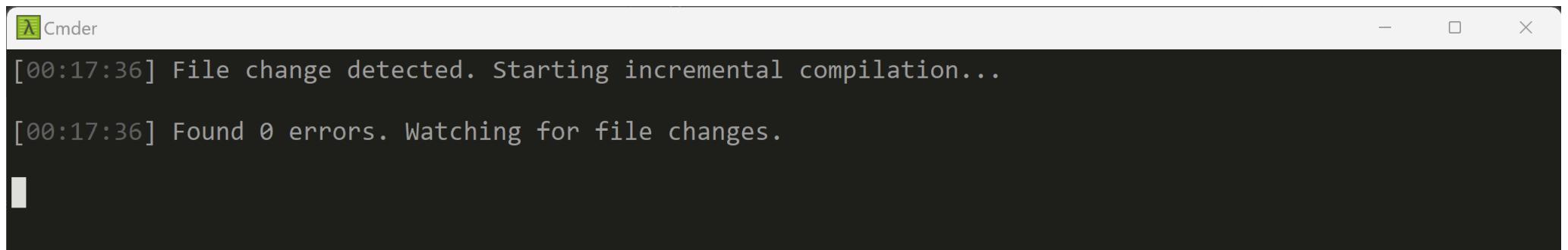
arnaud@DesktopArnaud /cygdrive/c/test-node
\$

Case when OK...

NO log !!

\$ tsc -w

Case when OK...
NO log

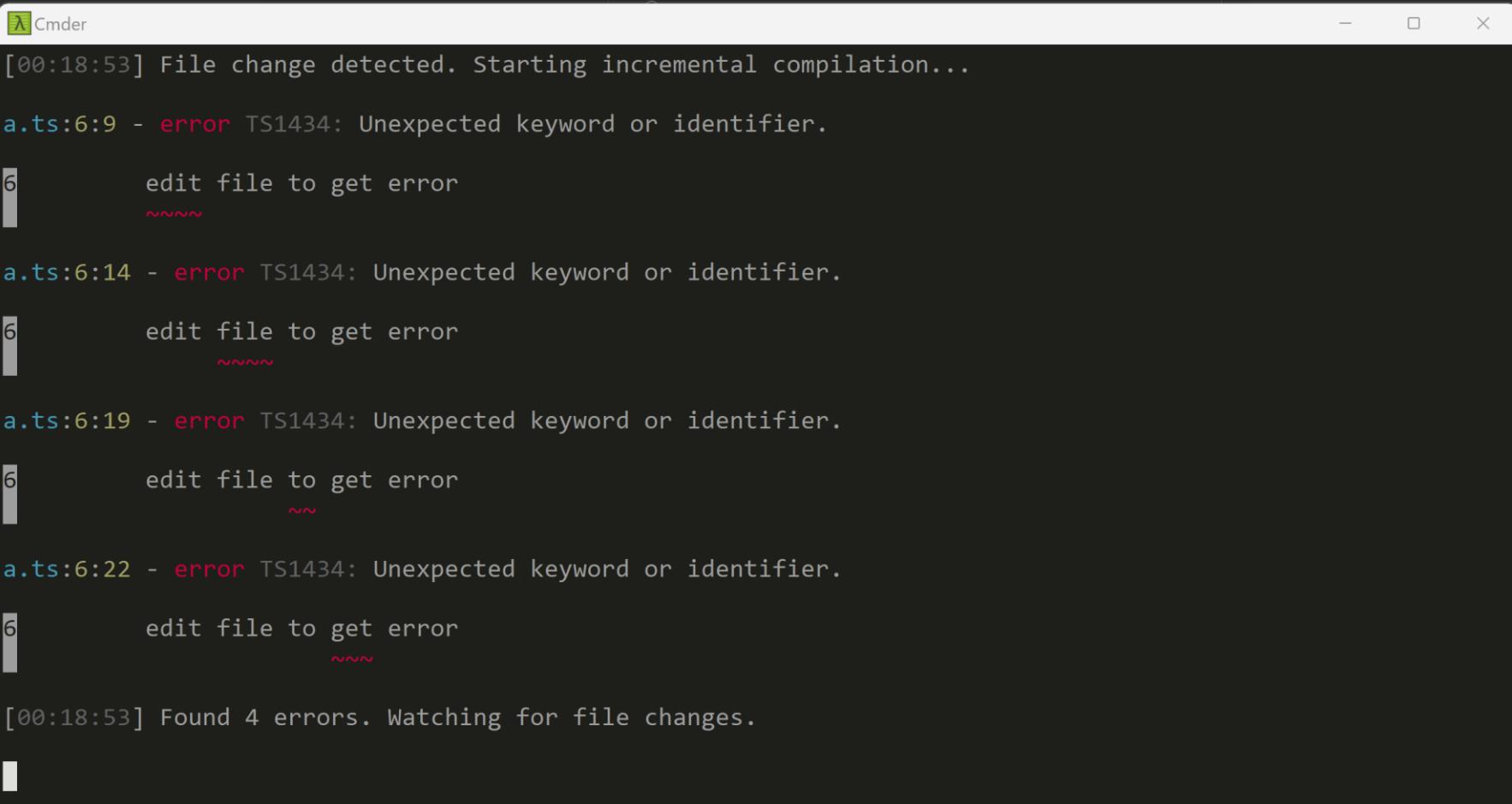


The screenshot shows a terminal window titled "Cmder". The output of the command \$ tsc -w is displayed:

```
[00:17:36] File change detected. Starting incremental compilation...
[00:17:36] Found 0 errors. Watching for file changes.
```

Case when error... cf next slide

Edit file to get Compile Error with -w (watch)



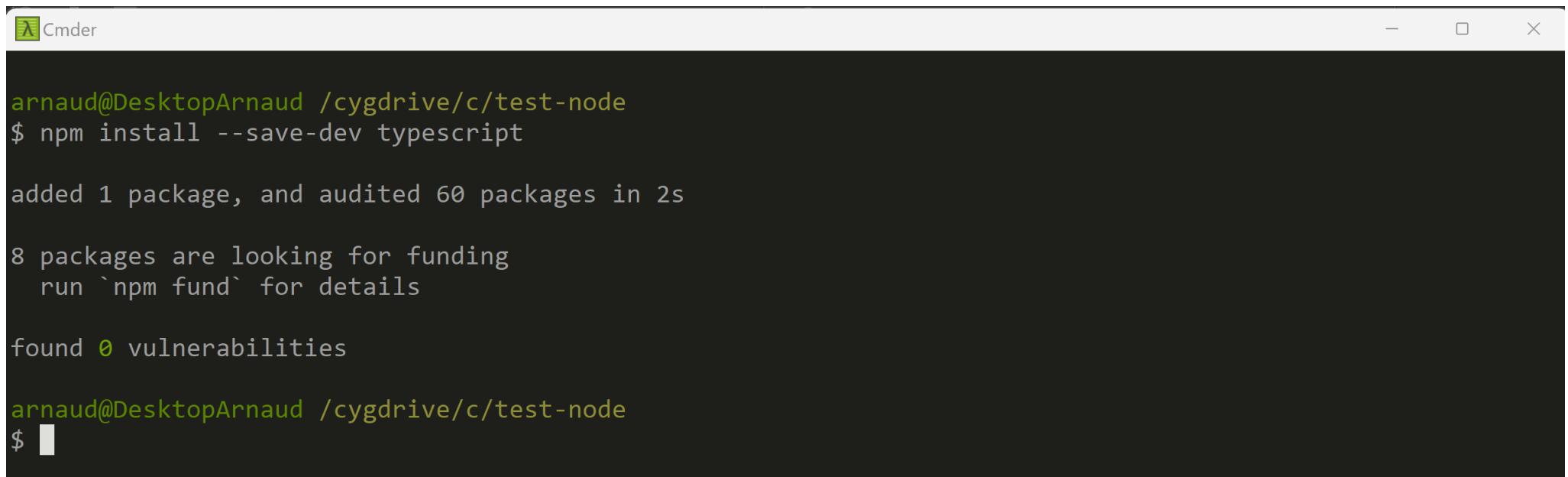
The image shows a code editor and a terminal window. The code editor has two tabs: 'tsconfig.json' and 'a.ts'. The 'a.ts' tab contains the following code:1 no usages
2 export class A {
3 no usages
4 constructor() {
5 console.log('Hey');
6 edit file to get error
7 }
8 }The terminal window, titled 'Cmder', shows the output of a TypeScript compilation process. It starts with a message about file change detection and then lists four errors from the 'a.ts' file, each corresponding to one of the 'edit file to get error' lines in the code:[00:18:53] File change detected. Starting incremental compilation...
a.ts:6:9 - error TS1434: Unexpected keyword or identifier.
6 edit file to get error
~~~~  
  
a.ts:6:14 - error TS1434: Unexpected keyword or identifier.  
6 edit file to get error  
~~~~  

a.ts:6:19 - error TS1434: Unexpected keyword or identifier.
6 edit file to get error
~~

a.ts:6:22 - error TS1434: Unexpected keyword or identifier.
6 edit file to get error
~~~~  
  
[00:18:53] Found 4 errors. Watching for file changes.

Install « typescript » in package.json  
devDependencies  
(devDependencies != dependencies, != global )

\$ npm install –save-dev typescript



```
arnaud@DesktopArnaud /cygdrive/c/test-node
$ npm install --save-dev typescript

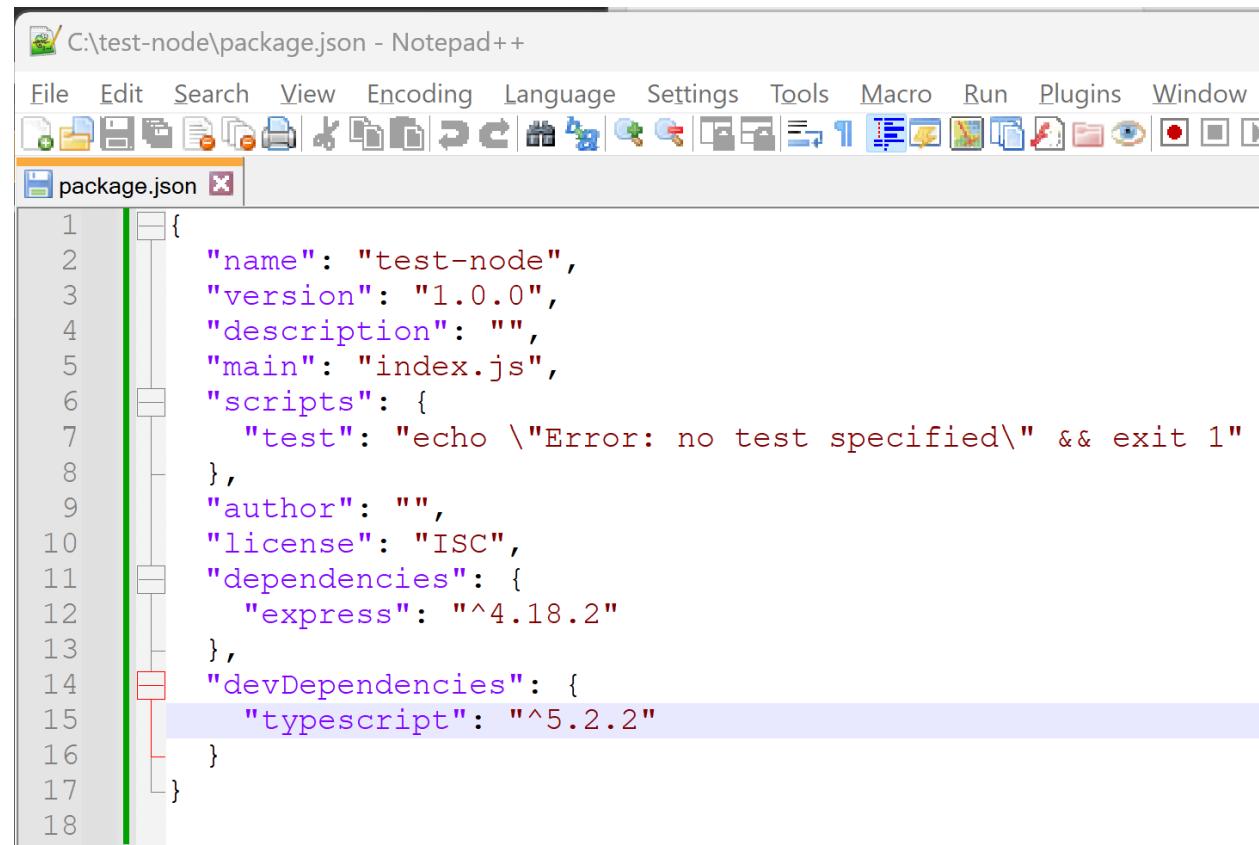
added 1 package, and audited 60 packages in 2s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

arnaud@DesktopArnaud /cygdrive/c/test-node
$ 
```

# devDependencies...



The screenshot shows a Notepad++ window displaying a `package.json` file. The file content is as follows:

```
1 {  
2   "name": "test-node",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC",  
11  "dependencies": {  
12    "express": "^4.18.2"  
13  },  
14  "devDependencies": {  
15    "typescript": "^5.2.2"  
16  }  
17}  
18
```

The `devDependencies` section is highlighted with a light purple background. The word `typescript` is also highlighted in purple.

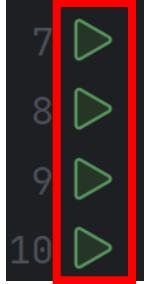
# Edit package.json « scripts »

```
package.json x

1  {
2      "name": "test-node",
3      "version": "1.0.0",
4      "description": "",
5      "main": "index.js",
6      "scripts": {  
7          "test": "echo \\\"Error: no test specified\\\" & exit 1",  
8          "compile-typescript": "tsc",  
9          "build": "tsc -w",  
10         "run": "node a.js"  
11     },  
12     "author": "",  
13     "license": "ISC",  
14     "dependencies": {  
15         "express": "^4.18.2"  
16     },  
17     "devDependencies": {  
18         "typescript": "^5.2.2"  
19     }  
20 }
```

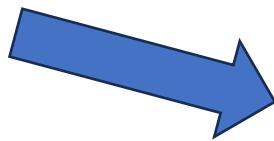
« **\$ npm run compile-typescript** »  
... is equivalent to « **\$ tsc** »

« **\$ npm run build** »  
... is equivalent to « **\$ tsc -w** »

**Notice:**  
**There are Run icons**  **for each script line**

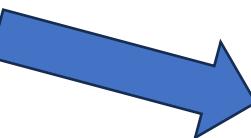
# Understanding Angular « ng » client internals

« npm run start»



Internally Call

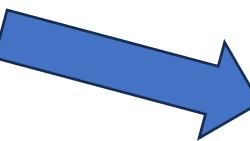
« **ng serve** »



Internally Call

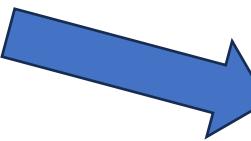
« **tsc -w** »

Compile all \*.ts to \*.js  
(maybe webpack all into a single app.js )



Internal call « **node** » with **express server**  
... http server running on port « http://localhost:4200 »

All commands « npm », « ng », « tsc »



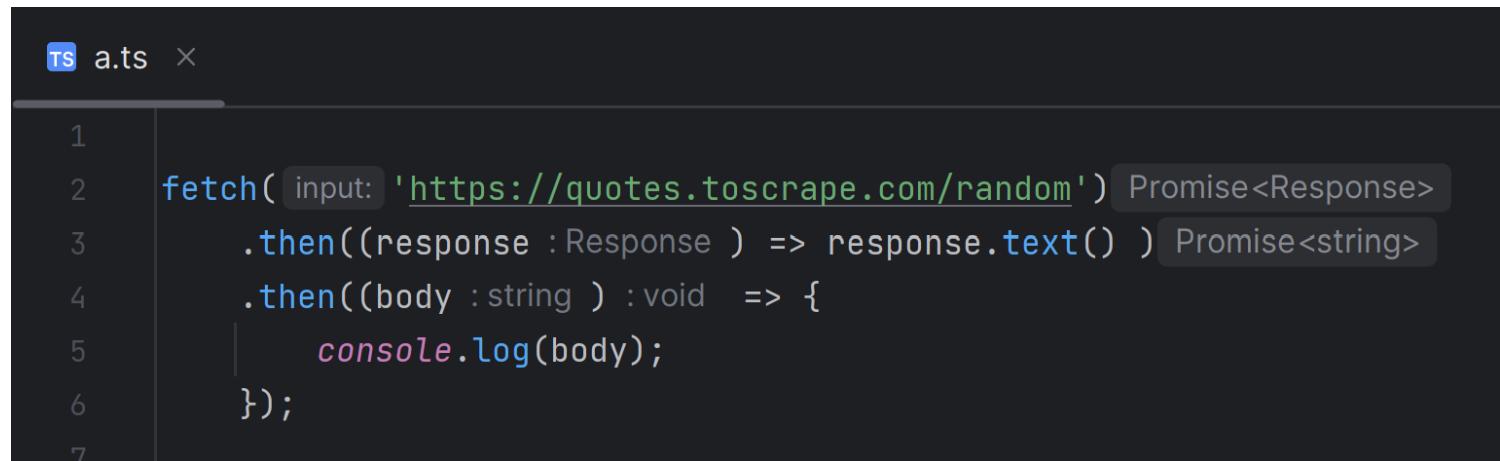
Run internally « **node** » !!!

Also Use « **live-reload.js** » ...  
so Web browser automatically refresh page on change

Coffe Break (?)

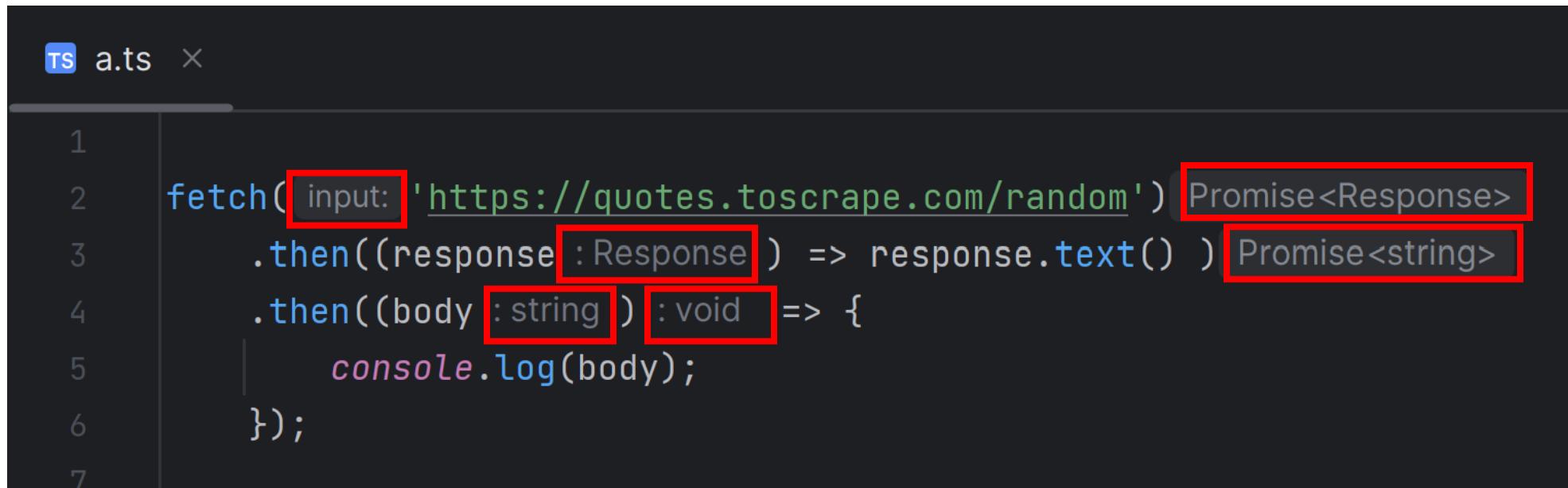
# Introducing difficulties with Async Code: Callbacks, Promises, async/await, ... Threads in a mono-threaded-user NodeJs model

A simple http GET fetch()  
.. The Hello world on http !



```
TS a.ts ×
1
2  fetch( input: 'https://quotes.toscrape.com/random') Promise<Response>
3    .then((response : Response ) => response.text() ) Promise<string>
4    .then((body : string ) : void  => {
5      console.log(body);
6    );
7
```

Noticed WebStorm automatically  
show « :type » and parameter « name: » in grey



The screenshot shows a code editor window for a file named 'a.ts'. The code uses the Fetch API to fetch a random quote from a specific URL. The code is annotated with type information, which is highlighted in grey. The annotations include:

- The URL 'https://quotes.toscrape.com/random' is annotated with a grey box labeled 'Promise<Response>'.
- The response object is annotated with a grey box labeled 'Promise<string>'.
- The body of the response is annotated with a grey box labeled 'void'.

```
1
2 fetch(input: 'https://quotes.toscrape.com/random') Promise<Response>
3   .then((response: Response) => response.text() ) Promise<string>
4   .then((body: string) : void => {
5     console.log(body);
6   );
7 }
```

# Adding console.log(...)

```
TS a.ts ×

8
9  console.log("1: calling (async) http GET ..");
10
11 fetch( input: 'https://quotes.toscrape.com/random' ) Promise<Response>
12     .then((response :Response ) => {
13         console.log('2: .. got response  http Status code: ' + response.status);
14         return response.text() // => cascade (+return) another async reading for consuming http response body ...
15     ) Promise<string>
16     .then((body :string ) :void  => {
17         console.log('3: .. got response text body');
18         console.log(body);
19     );
20
21 console.log('4: .. done launched async fetch() + attaching .then() callback ');
22
```

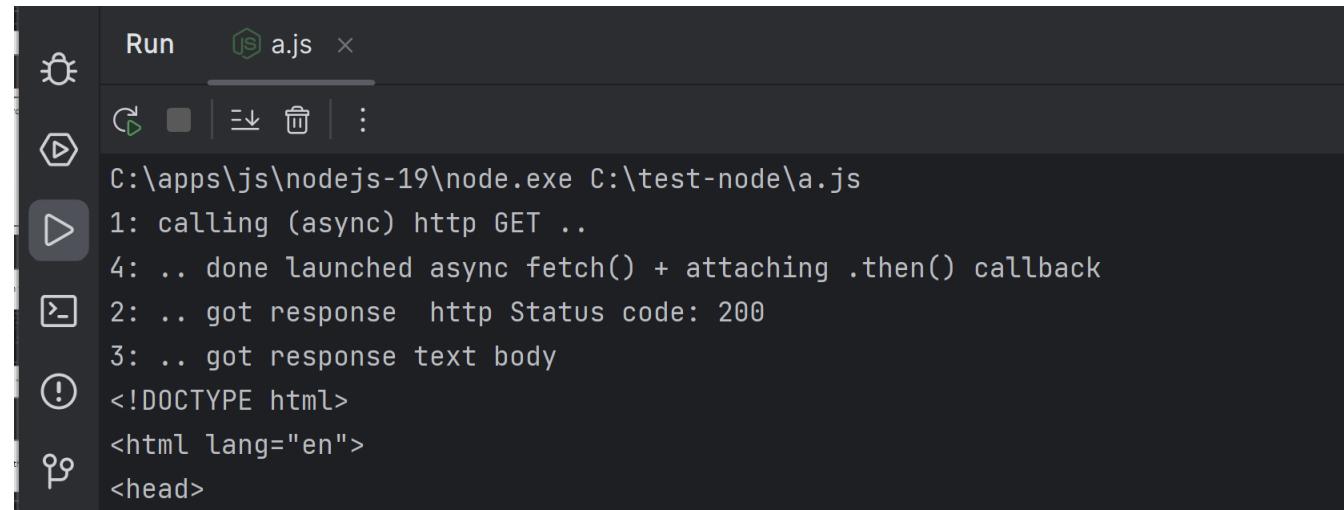
In which order are the console.log()**s** printed?

1, 2, 3, 4 ? ...

other? which order?



# Response: 1, 4, 2, 3 !!



The screenshot shows a terminal window with the following details:

- Tab Bar:** Run, a.js (highlighted), and a close button.
- Icons:** A vertical toolbar on the left with icons for file operations (refresh, new, save, delete, etc.), a play button, a minus sign, an exclamation mark, and a question mark.
- Output Area:** The text output of the script execution.

```
C:\apps\js\nodejs-19\node.exe C:\test-node\a.js
1: calling (async) http GET ..
4: .. done launched async fetch() + attaching .then() callback
2: .. got response http Status code: 200
3: .. got response text body
<!DOCTYPE html>
<html lang="en">
<head>
```

# Put Breakpoint and Debug

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Explorer:** Shows a file tree for a project named "test-node" located at "C:\test-node". The files listed include "c.ts", "a.ts", "b.ts", and "i-prefer-typescript-than-js.ts".
- Editor:** The file "a.ts" is open, showing TypeScript code. A red circular breakpoint icon is placed on line 17. The code uses the `fetch` API to make an asynchronous HTTP GET request to "https://quotes.toscrape.com/random".

```
8
9 console.log("1: calling (async) http GET ..");
10
11 fetch( input: 'https://quotes.toscrape.com/random' ) Promise<Response>
12     .then((response :Response ) => {
13         console.log('2: .. got response http Status code: ' + response.status);
14         return response.text() // => cascade (+missing return) another async reading for consuming http response body
15     } ) Promise<string>
16     .then((body :string ) :void => {   body: "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n<meta charset=\"UTF-8\">\n<title>Quotes to Scrape</title>\n<link rel=\"stylesheet\" href=\"https://unpkg.com/quotescraper@1.0.0/dist/quotescraper.css\">" });
17         console.log('3: .. got response text body');
18         console.log(body);   body: "<!DOCTYPE html>\n<html lang=\"en\">\n<head>\n<meta charset=\"UTF-8\">\n<title>Quotes to Scrape</title>\n<link rel=\"stylesheet\" href=\"https://unpkg.com/quotescraper@1.0.0/dist/quotescraper.css\">" );
19     });
20
21 console.log('4: .. done launched async fetch() + attaching .then() callback ');
22
23
24
```

A tooltip "callback for fetch(...).then()" is visible near the cursor.
- Debug View:** The "Debug" view is active, showing the "Threads & Variables" tab. It lists the "Main Thread" and several stack frames:
  - anonymous(), a.ts:18
  - processTicksAndRejections(), task\_queues
  - Async call from Promise.then
  - anonymous(), a.ts:16
  - Module.\_compile(), loader:1205
- Debugger Console:** Shows the expression "body" evaluated at the current frame, resulting in a large string of HTML content.
- Status Bar:** Displays "TypeScript 5.2.2 18:9 CRLF UTF-8 4 spaces".

# Can you spot the bug here ?

```
11  fetch( input: 'https://quotes.toscrape.com/random') Promise<Response>
12    .then((response :Response ) :void  => {
13      console.log(.. got response  http Status code: ' + response.status);
14      response.text() // => cascade (missing return) another async reading for consuming http response body ...
15    }) Promise<void>
16    .then((body :void ) :void  => {
17      console.log(.. got response text body');
18      console.log(body);
19    });
20
21  console.log(.. done launched async fetch() + attaching .then() callback ');
```

# Spot the Bug ... 4 Clues

- 1/ No Compile ERROR ... but Running => see in console logs
- 2/ Executing in Debug mode => check variables values
- 3/ WebStorm underline « .text() » as a possible warning
- 4/ Can you Spot the inconsistent Type infered by WebStorm  
(not a bug in WebStorm)

Clue 1/ Run, see in console log  
Understand the « undefined » instead of <html>... ?

# Clue 2 / Debug with Breakpoint ... indeed « body » is undefined

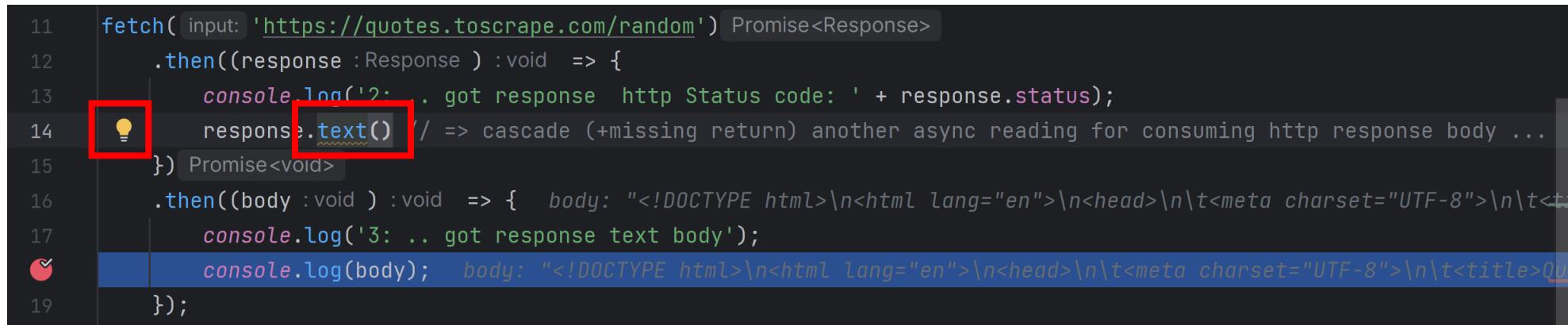
The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- Project Explorer:** Shows a file tree with a project named "test-node" containing files like "c.ts", "a.ts", "b.ts", and "i-prefer-typescript-than-js.ts".
- Code Editor:** An open file "a.ts" with the following code:

```
8 console.log("1: calling (async) http GET ..");
9
10 fetch( input: 'https://quotes.toscrape.com/random' ) Promise<Response>
11   .then((response :Response ) :void  => {
12     console.log('2: .. got response  http Status code: ' + response.status);
13     response.text() // => cascade (+missing return) another async reading for consuming http response body ...
14   ) Promise<void>
15   .then((body :void ) :void  => { body: undefined
16     console.log('3: .. got response text body');
17     console.log(body); body: undefined
18   );
19
20 console.log('4: .. done launched async fetch() + attaching .then() callback ');
21
22
23
24
```

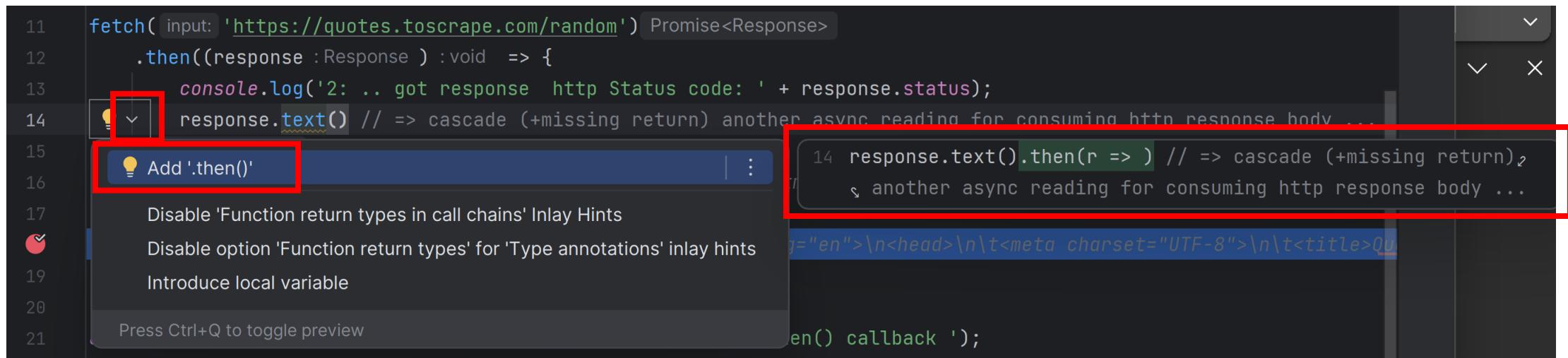
A red box highlights the line ".then((body :void ) :void => { body: undefined".
- Debug View:** A bottom panel titled "Debug" showing the current state of the "Main Thread".
  - The "Threads & Variables" tab is selected.
  - The "Local" variable list shows "body = undefined" highlighted with a red box.
  - The "Evaluate expression (Entrée) or add a watch (Ctrl+Maj+Entrée)" input field is empty.
  - The stack trace shows frames like "anonymous()", "processTicksAndRejections()", "Async call from Promise.then", and "anonymous()", all at line 18 of "a.ts".
- Status Bar:** Shows the file path "test-node > a.ts", the TypeScript version "TypeScript 5.2.2", and other system information like "18:9", "CRLF", "UTF-8", and "4 spaces".

# Clue 3/ .text() in underlined in WebStorm ?



```
11 fetch( input: 'https://quotes.toscrape.com/random') Promise<Response>
12     .then((response : Response) : void => {
13         console.log('2: .. got response http Status code: ' + response.status);
14         response.text() // => cascade (+missing return) another async reading for consuming http response body ...
15     ) Promise<void>
16     .then((body : void) : void => {   body: "<!DOCTYPE html>\n<html lang="en">\n<head>\n\t<meta charset="UTF-8">\n\t<t
17         console.log('3: .. got response text body');
18         console.log(body);   body: "<!DOCTYPE html>\n<html lang="en">\n<head>\n\t<meta charset="UTF-8">\n\t<title>Qu
19     });

```



```
11 fetch( input: 'https://quotes.toscrape.com/random') Promise<Response>
12     .then((response : Response) : void => {
13         console.log('2: .. got response http Status code: ' + response.status);
14         response.text() // => cascade (+missing return) another async reading for consuming http response body ...
15         |: Add '.then()' | ...
16             Disable 'Function return types in call chains' Inlay Hints
17             Disable option 'Function return types' for 'Type annotations' inlay hints
18             Introduce local variable
19
20             Press Ctrl+Q to toggle preview
21

```

The screenshot shows a tooltip for the `response.text()` call. The tooltip contains the following text:

- 14 response.text().then(r =>) // => cascade (+missing return),  
↳ another async reading for consuming http response body ...

# Check « .text() » : Browsing code -> definition !

The screenshot shows a code editor with two tabs: "a.ts" and "lib.dom.d.ts". The "lib.dom.d.ts" tab is active and highlighted with a red box. The code in "lib.dom.d.ts" defines the `BroadcastChannel` interface, which includes a `text(): Promise<string>` method. This method is also highlighted with a red box. A tooltip for this method provides MDN Reference links and a detailed description of the `BroadcastChannel` interface, stating it represents a named channel for communication between different documents of the same origin via a `message` event.

```
3151     formData(): Promise<FormData>;
3152
3153     json(): Promise<any>;
3154
3155     text(): Promise<string>;
3156 }
3157
3158 interface BroadcastChannelEventMap {
3159     "message": MessageEvent;
3160     "messageerror": MessageEvent;
3161 }
```

The `BroadcastChannel` interface represents a named channel that any browsing context of a given origin can subscribe to. It allows communication between different documents (in different windows, tabs, frames or iframes) of the same origin. Messages are broadcasted via a `message` event fired at all `BroadcastChannel` objects listening to the channel, except the object that sent the message.

**Note:** This feature is available in Web Workers

Supported by Chrome 54, Chrome Android 54, Edge 70, Firefox 38, Opera 41, Safari 15.4, Safari iOS 15.4

Body > text()

# Clue 4/ Inconsistent types

## ... was expecting « string », got « void »

```
TS a.ts x : 1 ^ v
8
9  console.log("1: calling (async) http GET ..");
10
11 fetch( input: 'https://quotes.toscrape.com/random') Promise<Response>
12     .then((response : Response) : void => {
13         console.log('2: .. got response http Status code: ' + response.status);
14         response.text() // => cascade (+missing return) another async reading for consuming http response body ...
15     }) Promise<void>
16     .then((body : void) : void => {
17         console.log('3: .. got response text body');
18         console.log(body);
19     });
20
21 console.log('4: .. done launched async fetch() + attaching .then() callback ');*/
22
```

Promise and nested Callbacks => Cascading Hell

async/await Solution

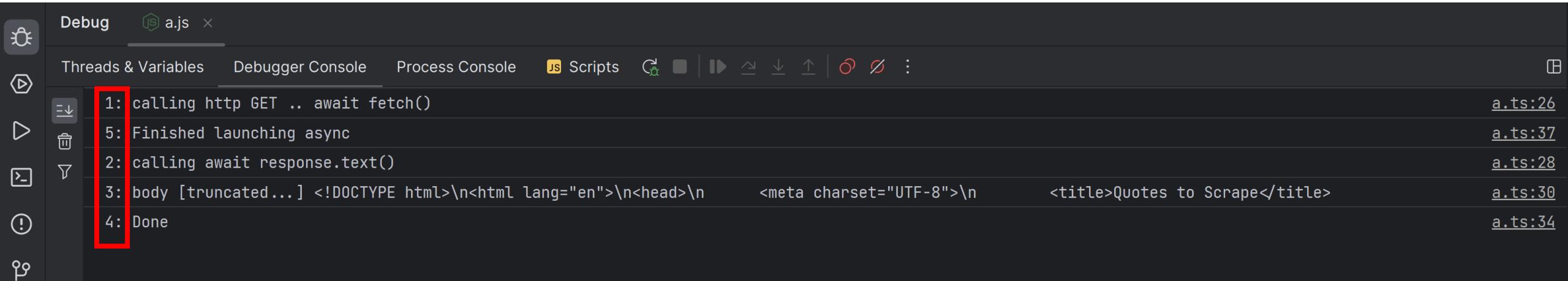
Cf also RxJS (out of scope here... But used in Angular)

# Transforming fetch().then() ... to await fetch()

```
TS a.ts ×
⚠ 1 ⚠ 1 ✅ 1 ^

24 // @ts-ignore
  1 usage
25 async function asyncFetchAndLog() :Promise<void> {
26   console.log("1: calling http GET .. await fetch()");
27   let response :Response = await fetch(input: 'https://quotes.toscrape.com/random');
28   console.log("2: calling await response.text()");
29   let body :string = await response.text();
30   console.log("3: body [truncated...] " + body.replace(searchValue: /\n/g, replaceValue: '\n').substring(0, 100));
31 }
32
33 asyncFetchAndLog().then(resp :void => {
34   console.log('4: Done');
35 });
36
37 console.log('5: Finished launching async');
```

# Run async/await ...



The screenshot shows the VS Code interface with the 'Debug' tab selected. The title bar indicates the file being debugged is 'a.js'. The left sidebar has several icons: a sun-like icon, a play/pause icon, a stop icon, a close icon, and a refresh icon. The main area is the 'Threads & Variables' tab of the Debug Console, which displays the following log output:

```
1: calling http GET .. await fetch() a.ts:26
5: Finished launching async a.ts:37
2: calling await response.text() a.ts:28
3: body [truncated...] <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Quotes to Scrape</title> a.ts:30
4: Done a.ts:34
```

The first two lines are highlighted with a red box.

Coffe Break / Lunch Break (?)

# Home Work

follow the simple doc provided in "express" NPM package documentation to build a Rest API Http Server

when calling "HTTP GET /hello", your server should answer  
HTTP OK 200, with http headers:

"Content-Type= application/json"

and http body response:

{ "message": "Hello" }

Then launch your server, and test it using chrome / curl / postman

When calling "HTTP PUT /hello" with request body { "message": "your new message" }  
then the server should store it for next GET request

Next Steps...