

# Node

“Exécuter du JavaScript partout”



# Node

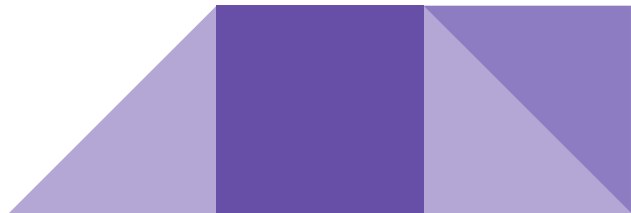
Crée par Ryan Dahl en 2009.

Node apporte le développement javascript sans navigateur (en utilisant le moteur js (moteur “v8”) de chrome)

Permet donc de créer des outils CLI, serveur backend,..

Des concurrents sont apparus ces dernières années:

Deno (par le créateur de node..), bun



# Exécuter node: écrire du js dans un terminal

```
$ node // Pour entrer en mode interactif  
> console.log("hello world")  
hello world  
  
> const a = 10  
undefined // undefined car l'instruction ne retourne rien  
  
> const b = 10  
undefined  
  
> a + b  
20  
  
$ .exit // Pour quitter le mode interactif
```

# Première app

```
$ echo "console.log('Hello world !')" > index.js
```

```
// Si on passe un argument à la commande node (vs slide  
// précédente), node exécute le fichier.
```

```
$ node index.js  
> Hello world !
```

# Installation

- à la “main”: <https://nodejs.org/en/download/prebuilt-installer>
  - penser à mettre à jour le \$PATH de l'os
- selon votre os: <https://nodejs.org/en/download/package-manager>

Pour tester l'installation:

```
$ node --version
```

## Download Node.js®


Download Node.js the way you want.

[Package Manager](#) [Prebuilt Installer](#) [Prebuilt Binaries](#) [Source Code](#)

Install Node.js  on  using

```
1 # installs nvm (Node Version Manager)
2 curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
3
4 # download and install Node.js (you may need to restart the terminal)
5 nvm install 20
6
7 # verifies the right Node.js version is in the environment
8 node -v # should print 'v20.18.0'
9
10 # verifies the right npm version is in the environment
11 npm -v # should print '10.8.2'
```

Bash

 Copy to clipboard



# Let's go

installer node + première appli

# Node - module

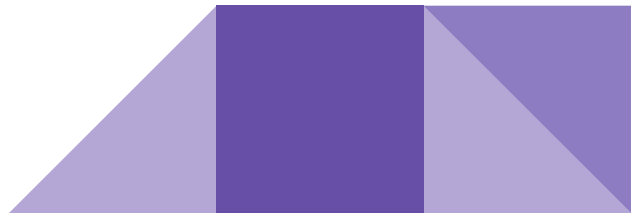
Un module regroupe plusieurs objet JS (variable, fonction, class,...)

Pour simplifier, un module = un fichier (qui peut regrouper d'autre module)

Il existe plusieurs façon d'importer/exporter un module: CommonJS et ESM

Pour activer l'un ou l'autre :

- fichier en .js => ajouter "type": "module" dans le package.json (par défaut dans les versions récentes de node (à partir de la 23 avec un flag)
- fichier en .mjs => module esm
- fichier en .cjs => module commonjs

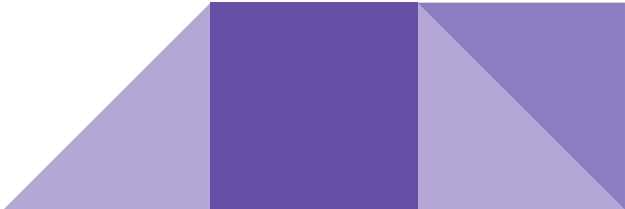


# Node - Commonjs

## import:

- `const moduleA = require('./moduleA');`

## export:

- `module.exports = { moduleA };`
  - `exports.moduleA = moduleA;`
- 



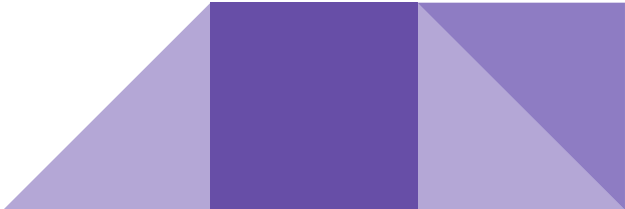
# Node - ESM (es6)

en 2025, c'est la façon de faire ! commonjs c'est bien pour le legacy

## import:

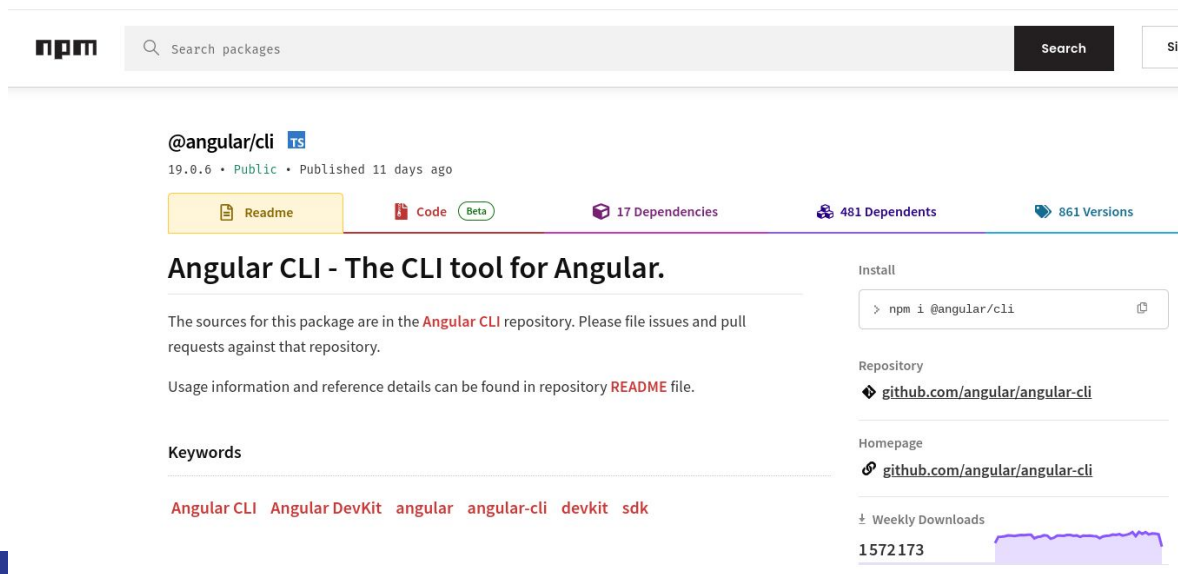
- `import moduleA from './moduleA';`
- `import * as moduleB from './moduleB';`
- `import { moduleB } from './moduleB';`
- `import moduleA, { moduleA1 } from 'moduleA';`

## export:

- `export default moduleA`
  - `export { moduleA as default }`
  - `export const moduleB = () => { console.log('toto') }`
- 

# NPM

- Gestionnaire de paquet par défaut de node
- Outil de gestion de dépendance, version, publication,...
- Alternatives: yarn, pnpm,...



The screenshot shows the NPM package page for `@angular/cli`. At the top, there's a search bar with the text "Search packages" and a "Search" button. Below the search bar, the package name `@angular/cli` is displayed with a TypeScript icon. The version `19.0.6` is shown as "Public" and "Published 11 days ago". There are buttons for "Readme", "Code" (with a "Beta" badge), "17 Dependencies", "481 Dependents", and "861 Versions". The main heading is "Angular CLI - The CLI tool for Angular." Below this, a paragraph states: "The sources for this package are in the **Angular CLI** repository. Please file issues and pull requests against that repository." Another paragraph says: "Usage information and reference details can be found in repository **README** file." There is a "Keywords" section with the following terms: "Angular CLI", "Angular DevKit", "angular", "angular-cli", "devkit", and "sdk". On the right side, there is an "Install" section with a code block: `> npm i @angular/cli`. Below that is the "Repository" section with a link to `github.com/angular/angular-cli`. The "Homepage" section also has a link to `github.com/angular/angular-cli`. At the bottom, there is a "Weekly Downloads" section showing a bar chart and the number `1572173`.

npm Search packages Search

@angular/cli TS  
19.0.6 • Public • Published 11 days ago

Readme Code Beta 17 Dependencies 481 Dependents 861 Versions

### Angular CLI - The CLI tool for Angular.

The sources for this package are in the **Angular CLI** repository. Please file issues and pull requests against that repository.

Usage information and reference details can be found in repository **README** file.

**Keywords**

Angular CLI Angular DevKit angular angular-cli devkit sdk

Install

```
> npm i @angular/cli
```

Repository

github.com/angular/angular-cli

Homepage

github.com/angular/angular-cli

Weekly Downloads

1572173

# NPM

node et npm s'appuient sur un fichier de config: package.json

Pour le générer: npm init

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

## Les commandes principales:

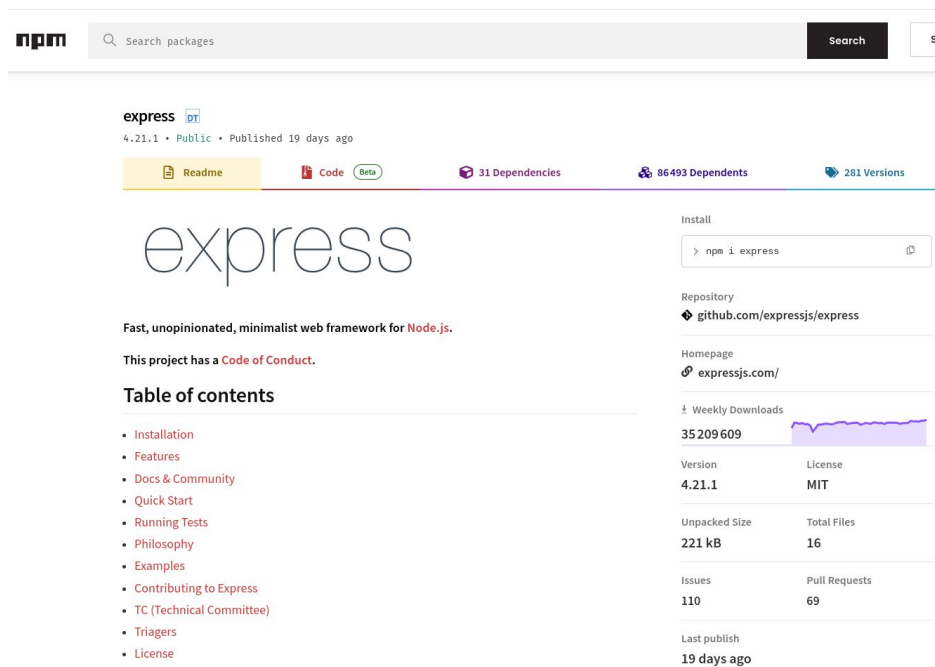
- npm install express
- npm install jasmine -D (-D = dépendance de dev)
- npm uninstall
- npm run <script> (script défini dans package.json)
- npm test (= npm run test)
- npm build (= npm run build)

```
"scripts": {  
  "test": "echo \\\"Error: no test specified\\\"",  
  "hello": "echo hello world"  
},  
"author": ""
```

```
daniel@penguin:~/dev/gi$ npm run hello  
  
> test@1.0.0 hello  
> echo hello world  
  
hello world  
daniel@penguin:~/dev/gi$
```

# NPM - packages

Exemple: <https://www.npmjs.com/package/express>



The screenshot shows the npm package page for 'express'. At the top, there's a search bar with 'npm' and a search button. Below the search bar, the package name 'express' is displayed with a version '4.21.1', a status 'Public', and a note 'Published 19 days ago'. There are links for 'Readme', 'Code' (with a 'Beta' badge), '31 Dependencies', '86 493 Dependents', and '281 Versions'. The main section features the 'express' logo and the tagline 'Fast, unopinionated, minimalist web framework for Node.js'. Below this, it says 'This project has a Code of Conduct.' and a 'Table of contents' with a list of links: Installation, Features, Docs & Community, Quick Start, Running Tests, Philosophy, Examples, Contributing to Express, TC (Technical Committee), Triagers, and License. On the right side, there's an 'Install' section with a command 'npm i express', a 'Repository' link to 'github.com/expressjs/express', a 'Homepage' link to 'expressjs.com/', a 'Weekly Downloads' bar chart showing '35 209 609' downloads, and a table with package details.

Version	License
4.21.1	MIT

Unpacked Size	Total Files
221 kB	16

Issues	Pull Requests
110	69

Last publish  
19 days ago

# NPM - le versionning

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

Exemple de version: 18.9.2

18 => version majeur

9 => version mineur

2 => version patch

- version exacte: 18.9.2
- dernière vers. avec vers. minimale: : ~18.9.2
- dernière version de patch: 18.9
- dernière vers. mineur avec vers. minimale: ^18.9.2
- dernière version mineur: 18
- dernière version : \*

# Let's go

- initialiser un repo npm avec `npm init` et installer la dépendance `chalk`
- afficher un message en couleur:
  - `console.log(chalk.blue("Hello"))`
- script npm qui permet de lancer le programme avec `npm start`

# Node - cli tools

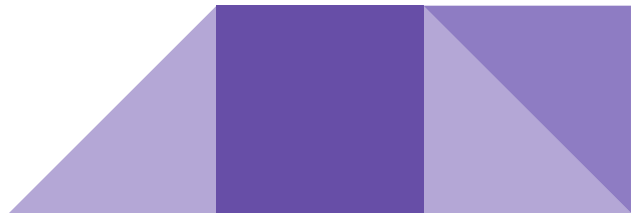
Exemple d'utilisation: créer un fichier, télécharger un fichier, executer un programme,...

Les arguments: `process.argv`

`argv[0]` = chemin executable node

`argv[1]` = fichier source (index.js)

`args[2]` (et +): arguments





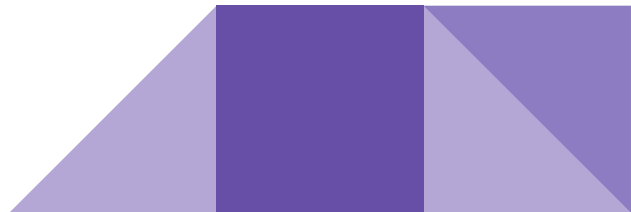
# Node process

Outils “core” de node qui se trouve dans le paquet natif 'node:process'

Mais pas besoin de l'importer, c'est le cas par défaut.

On trouve notamment les méthodes :

- quitter le programme:
  - `process.exit(0)`
- récupérer les variables d'environnement
  - `process.env` // toutes les variables (attention au mot de passe..)
  - `process.env.TOTO`
- chemin courant
  - `process.cwd`



# Node fs

Outils qui permet de gérer les fichiers.

Attention trois formes de l'api une **“synchrone”**, une **“asynchrone classique”** et **“asynchrone via promesse”**



# Node fs - read

```
import fs from 'node:fs';

try {
  const data = fs.readFileSync('/Users/joe/test.txt', 'utf8');
  console.log(data);
} catch (err) {
  console.error(err);
}
```

# Node fs - read

```
import fs from 'node:fs';

fs.readFile('/Users/joe/test.txt', 'utf8', (err, data) => {
  if (err) {
    console.error(err);
    return;
  }
  console.log(data);
});
```

## Node fs - read

```
import fs from 'node:fs/promises'; // 'node:' est optionnel
try {
  const data = await fs.readFile('package.json');
  console.log(data);
} catch (err) {
  console.log(err);
}
```

# Let's go

Créer un programme cli qui prend un argument en entrée :

- si l'argument = 0
  - on affiche en bleu "ok"
- sinon
  - on affiche en rouge "erreur"
  - enregistre un fichier avec l'argument erreur-{timestamp}.txt
    - timestamp = new Date().getTime()
  - forcer la sortie du programme

## Node - créer un serveur

```
import http from 'http';  
const server = http.createServer((request, response) => {  
  response.writeHead(200);  
  response.end("Hello world");  
});  
server.listen(3000);
```



# Let's go

Créer un serveur qui permet de récupérer la liste complète des joueurs d'une équipe



# Node - serveur

La lib la plus connue avec node: express.js (<https://expressjs.com/>)

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

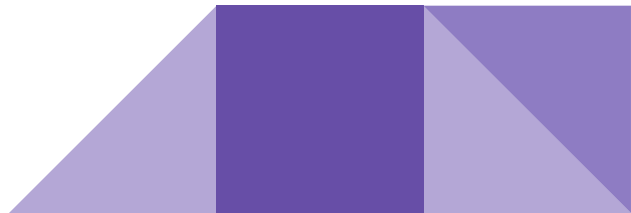
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

# Express

Pour créer un serveur express :

```
const app = express();
```

```
app.listen(3000);
```



# Express

Créer une route :

```
app.get('/', (requete, response) => {  
    response.send('Hello world');  
})
```

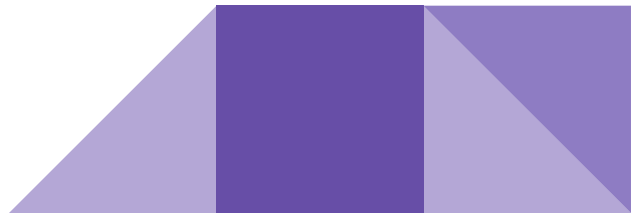
On peut aussi avoir toutes les méthodes http : post, put, delete,...



# Express

## Les middlewares

```
app.all("/", (request, response) => {  
    response.send('Hello all');  
})
```



# Express

Envoyer du HTML :

```
response.sendFile(path.join(process.cwd(), "index.html"));
```

Au lieu de `process.cwd()`, en commonjs on peut aussi utiliser `__dirname` (il existe aussi `_filename`), mais ne fonctionne pas en ESM.



# Let's go

Créer un serveur avec les fonctions suivantes:

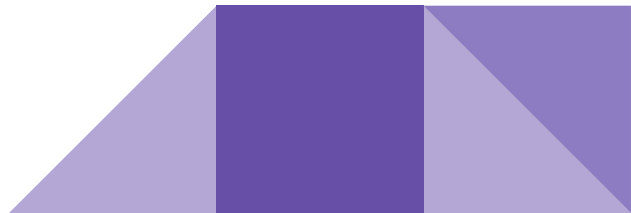
- récupère la liste complète des joueurs d'une équipe.
- récupère via leur identifiant chaque membre de l'équipe
- ajoute un membre dans l'équipe
- supprime un membre

# sqlite

```
import sqlite3 from "sqlite3"
```

```
const db = new sqlite3.Database("data.db");
```

Pour avoir une db en mémoire DataBase(":memory:");



# sqlite

Créer une table:

```
const sql = `CREATE TABLE IF NOT EXISTS players ....`
```

```
db.exec(sql, (err) => { if (err) console.error(err) });
```

```
db.close();
```

A decorative graphic in the bottom right corner consisting of three overlapping purple shapes: a triangle on the left, a square in the middle, and another triangle on the right, all in different shades of purple.



# sqlite

Récupérer un élément

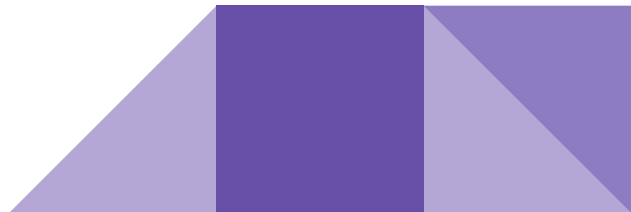
```
const sql = `SELECT * FROM players WHERE id = ?`;  
db.get(sql, [1], (err, row) => { if (err) console.error(err) });  
db.all(SELECT * FROM players, [], (err, rows) => { if (err) console.error(err) });  
db.close();
```



# sqlite

Ajouter, mettre à jour, supprimer un élément

```
const sql = `INSERT INTO players(name) VALUES(?)`;
db.run(sql, ['Daniel'] (err) =>{ if (err) console.error(err) });
db.close();
```





# Let's go

Mettre à jour le serveur pour utiliser sqlite3

# Les tests

Plusieurs framework: jest, vitest, jasmine, mocha, ..

Le nommage des fichiers : mafeature.spec.js ou mafeature.test.js (selon le framework et la config)

```
describe("ma feature", () => {  
  test("should be true", () => { // on peut utiliser "it" aussi à la place de test  
    expect(getValue()).toBeTrue();  
  });  
});
```

