Node.js est un environnement d'exécution **JavaScript** open source et multiplateforme. C'est un outil populaire pour presque tous les types de projets !

Node.js exécute le moteur JavaScript V8, le cœur de Google Chrome, en dehors du navigateur. Cela permet à Node.js d'être très performant.

Une application Node.js s'exécute en un seul processus, sans créer de nouveau thread pour chaque requête. Node.js fournit un ensemble de primitives d'E/S asynchrones dans sa bibliothèque standard qui empêchent le blocage du code JavaScript et, en général, les bibliothèques de Node.js sont écrites à l'aide de  paradigmes non bloquants, faisant du comportement de blocage l'exception plutôt que la norme.

**Un grand nombre de bibliothèques**

npm avec sa structure simple a aidé l'écosystème de Node.js à proliférer, et maintenant le registre npm héberge plus de 1 000 000 de packages open source que vous pouvez utiliser librement.

Le module http : <https://devdocs.io/node~8_lts-http/>

Exemple de création d’une instance de serveur :

const http = require("http");

// Créer une instance du serveur node

const server = http.createServer( (request, response) => {

    //On déclare les éléments du header de la réponse

    response.writeHead(200, {

    "Content-Type": "text/html; charset=utf-8", // permets d'interpreter les caractères spéciaux

    "mon-header-perso" : "Bonjour à tous"

    });

    // On génère la réponse à afficher à l'utilisateur

    response.end("Je suis généré depuis le serveur Node JS");

});

// On démarre l'instance du serveur

server.listen(9740, "localhost", () => {

    console.log("Serveur disponible à l'adresse http://localhost:9740");

});

// Autre manière de lancer l'instance du serveur

server.listen(9740);

Les framework de node js => Express

[[Express](https://expressjs.com/) est le framework actuellement le plus populaire dans Node et est la bibliothèque sous-jacente pour un grand nombre d'autres [cadres applicatifs web pour Node](https://expressjs.com/fr/resources/frameworks.html). Il fournit des mécanismes pour :

* Écrire des fonctions de traitement pour différentes requêtes HTTP répondant à différentes URI (par le biais des routes).
* Intégrer avec les moteurs de rendu de « vues » dans le but de générer des réponses en insérant des données dans des modèles (« templates »). Configurer certains paramètres d'applications comme le port à utiliser à la connexion et l'emplacement des modèles nécessaires pour la mise en forme de la réponse.
* Ajouter des requêtes de traitement « middleware » (intergiciel) où vous le voulez dans le tunnel gestionnaire de la requête.]

INSTALLATION

1ère étape : accéder à npm =

* dans terminal, taper « **npm init** » (un package json est créé. Je réponds aux questions 1 par 1).
* OU j’écris dans le terminal **npm init -y** (si je veux aller vite, je réponds aux questions par défaut).

2ème étape : modifier le fichier package.json =

* Supprimer la ligne : "test": "echo \"Error: no test specified\" && exit 1"
* Remplacer par : "start" : "node app.js" (nom du fichier de démarrage)

3ème étape : installer Express dans le terminal =

* Ecrire **npm install express**

(un dossier node\_modules apparaît dans le repo)

4ème étape : créer la dépendance =

* Taper dans le terminal = **npm i express**
* Une ligne est apparue dans le fichier package json =

"dependencies": {

"express": "^4.17.3"

}

A noter = lors du transfert (git) du repo, ne pas oublier d’ajouter node\_module dans le git.ignore (car dossier trop lourd).   
= lors de la récupération d’un projet nodejs, vérifier dans package json les lignes soulignées (modules absents) + écrire dans le terminal = **npm install**

5ème étape : installer nodemon =

* Dans le terminal écrire = npm i nodemon --save-dev
* Modifier la ligne start dans package.json =

"start": "nodemon app.js"

* Dans le terminal écrire = npm start (pour lancer le programme)

6ème étape : importer le module express =

* Dans le fichier js principal (exemple app.js) écrire les commandes pour instancier un serveur

//On importe le module express

const express = require('express');

// Création d'une instance de serveur (node = .createServeur)

const app = express();

//On définit le port d'écoute

const port = 9740;

Pour connaitre les différentes request du protocole http :  
<https://developer.mozilla.org/fr/docs/Web/HTTP/Methods>

Exemple en code dans ce repo :   
C:\Users\Morgane Tomas Ramboz\Documents\1- MES DOCUMENTS\PRO\ALT\_RH\02-BACKEND\04-NODE-JS\02-NODE-EXPRESS

Les requêtes HTTP avec express :

**Get :** récupérer des données

**Post :**  pour envoyer une entité (données) vers le serveur (base de données par exemple)

**Put :** remplace les données

**Delete :** supprime les données

Source : <https://developer.mozilla.org/fr/docs/Web/HTTP/Methods>

Requêtes de traitement (fonctions intermédiaires) les Middleware :

// EXEMPLE DE MIDDLEWARE

const exempleMiddleware = function (req, res, next) {

    console.log('LOGGED')

    next()

}

* C’est le **next()** qui transforme la fonction en middleware.

Source : <http://expressjs.com/en/guide/writing-middleware.html>

A NOTER : Express fournit des méthodes pour spécifier quelle fonction est appelée pour une méthode HTTP particulière (GET, POST, SET, etc.) et un modèle d'URL ("Route"), ainsi que des méthodes pour spécifier quel moteur de rendu de vues ("view") est utilisé, où sont les modèles de vues et quel modèle utiliser pour générer une réponse. Vous pouvez utiliser les fonctions intermédiaires d'Express pour prendre en charge les cookies, les sessions, les utilisateurs, obtenir les paramètres POST/GET, etc. Vous pouvez utiliser n'importe que système de base données supporté par Node (Express ne définit aucun comportement relatif aux bases de données).