



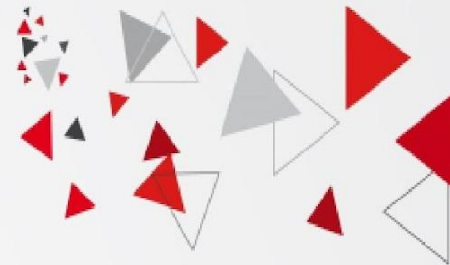
Introduction JavaScript Côté Serveur

Node JS

Année universitaire
2021-2022

4 TWIN

Présentation générale



- Unité pédagogique: Web
- Module: JavaScript côté serveur
- Cible : 4 TWIN
- Charge: 21 h
- Pré-requis: JavaScript



Objectifs

A la fin de ce module l'étudiant sera capable de :

- Définir Node.js
- Distinguer les principales caractéristiques du Node JS
- Mettre en œuvre un exemple pratique

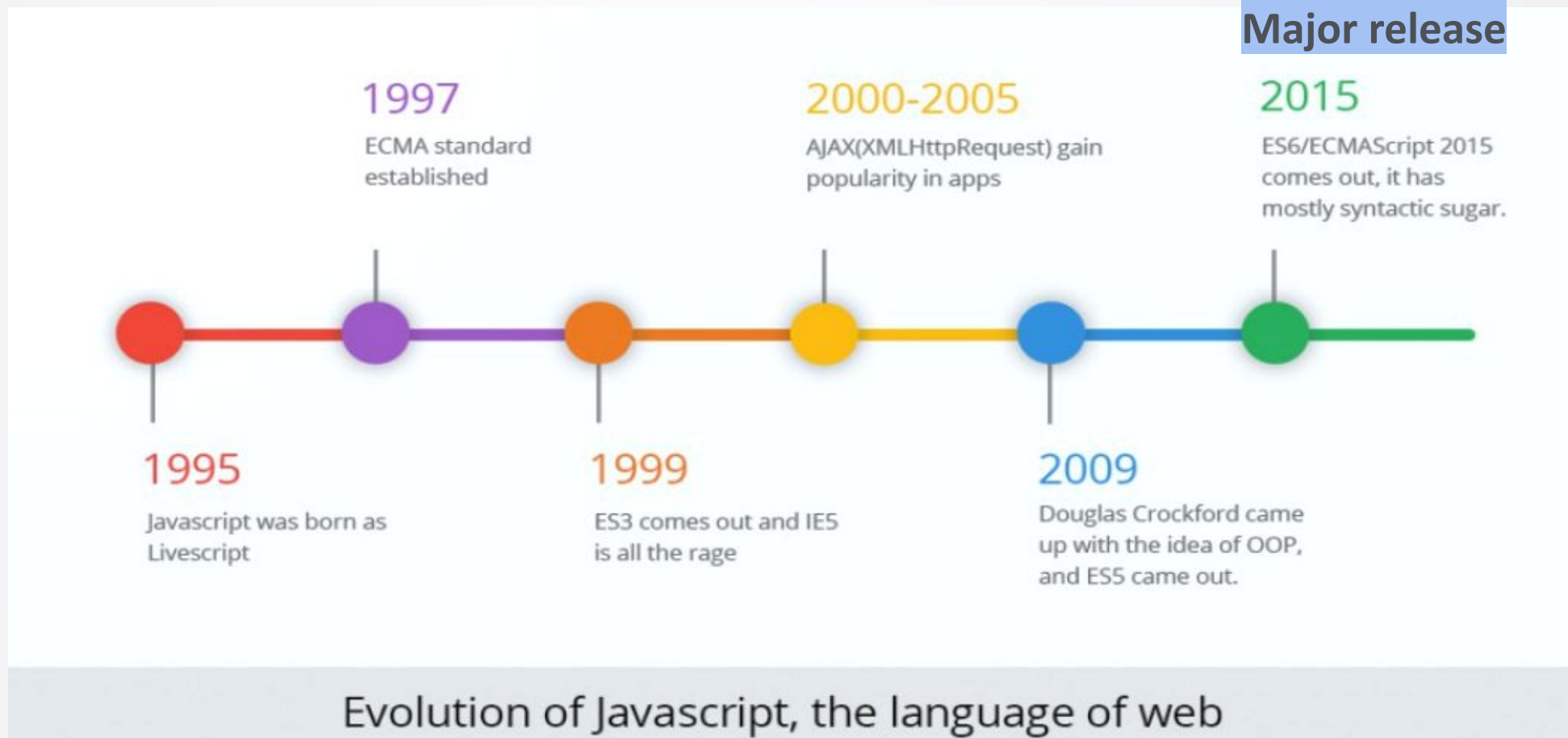
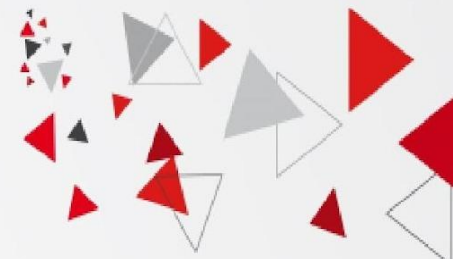




Plan

- ✓ Introduction
- ✓ Les caractéristiques du Node JS
- ✓ Architecture technique du Node JS
- ✓ Les modules Node JS
- ✓ NPM Node Package Manager
- ✓ Exemple pratique

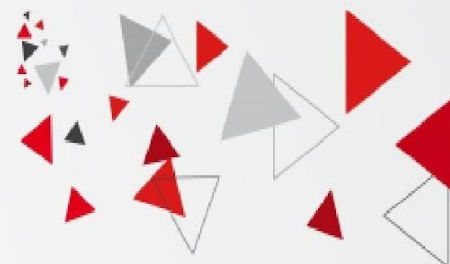
▶ Introduction



ES7/ECMAScript2016 ES8/ECMAScript2017 ES9/ECMAScript2018 ES10/ECMAScript2019 ES11/ECMAScript2020

<https://programming.vip/docs/new-features-of-es6-es7-es8-es9-and-es10.html>
<https://medium.com/codingtown/ecmascript-2020-aka-es-11-9c547f69d96f>

► Introduction



Les leaders de JavaScript

- « *JS is about as low as we can go. But it also has higher level facilities.* » – **Brendan Eich** (inventor of JavaScript)
- “The World's Most Misunderstood Programming Language Has Become the World's Most Popular Programming Language.” -**Douglas Crockford** (writer and speaker on JavaScript)
- “JavaScript has certain characteristics that make it very different than other dynamic languages, namely that it has no concept of threads. Its model of concurrency is completely based around events. ” - **Ryan Dahl**(inventor of Node.js)





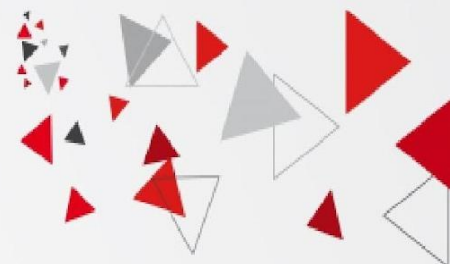
Introduction

L'ECMAScript et ses dérivés

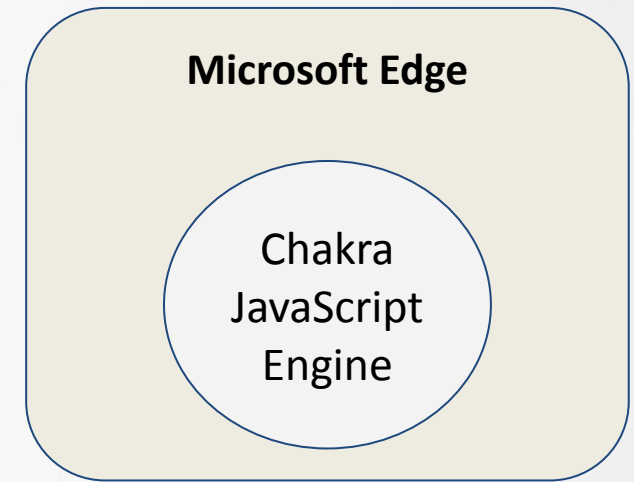
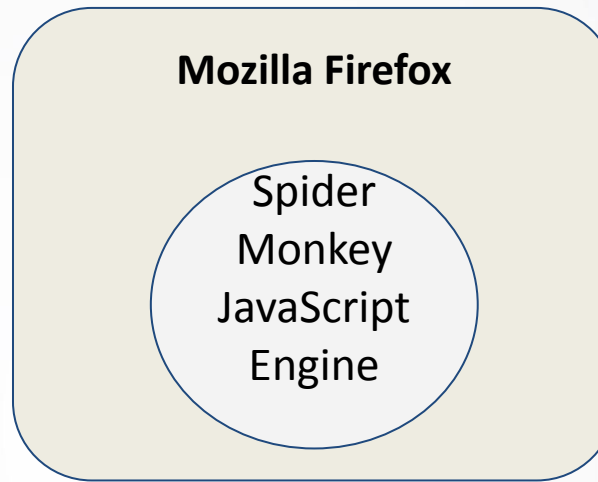
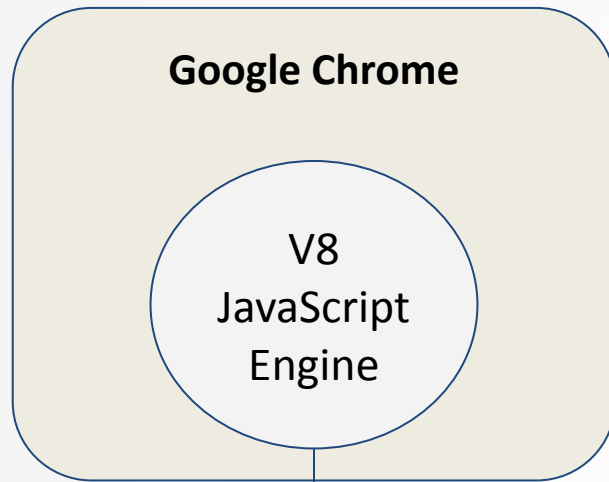
- L'ECMA: European association for standardizing information and communication systems (European Computer Manufacturers Association) [<http://ecma-international.org/>]
- Sa mission consiste à :
 - standardiser les langages
 - créer une référence du langage pour qu'il soit utilisé par d'autres personnes et embarqués dans d'autres logiciels.

L'ECMA International standardise le langage sous le nom d'ECMAScript

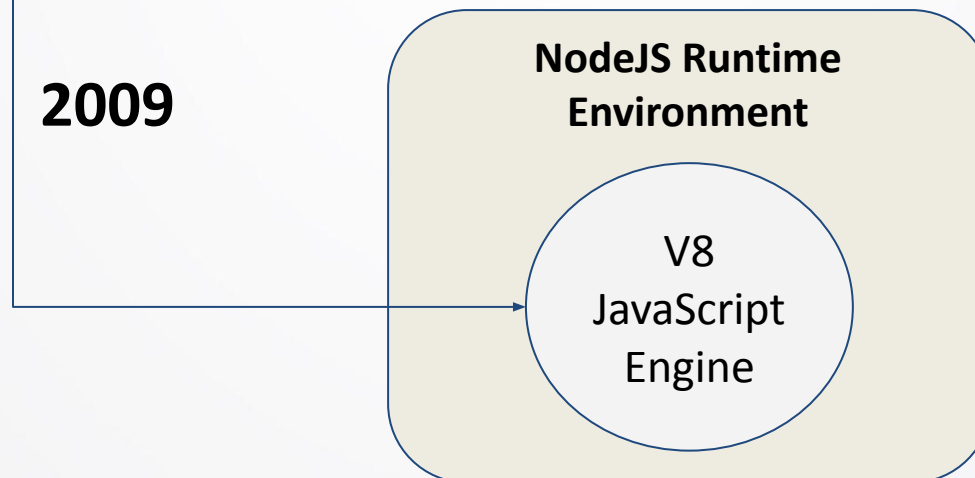
▶ Introduction



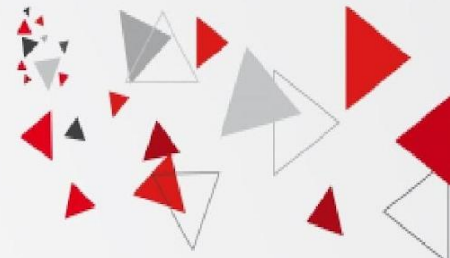
JavaScript côté client



JavaScript côté serveur

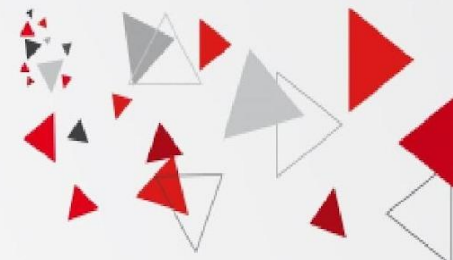


Introduction



- Node.js a été créé par Ryan Dahl en 2009.
- Plateforme logicielle libre en JavaScript orientée vers les applications réseau.
- Représente une alternative à des langages serveur comme PHP, Java ou Python
- Node.js est un moteur d'exécution JavaScript basé sur le moteur JavaScript V8 de Chrome.

► Introduction



Qui utilise NodeJS ?



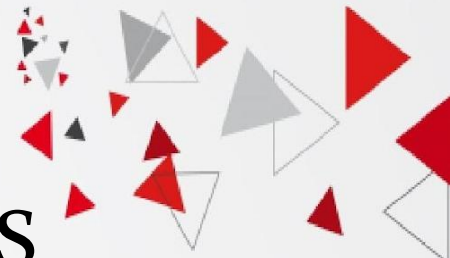
► Les caractéristiques du Node Js



Node.js® is a JavaScript runtime built on **Chrome's V8 JavaScript engine**. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world.

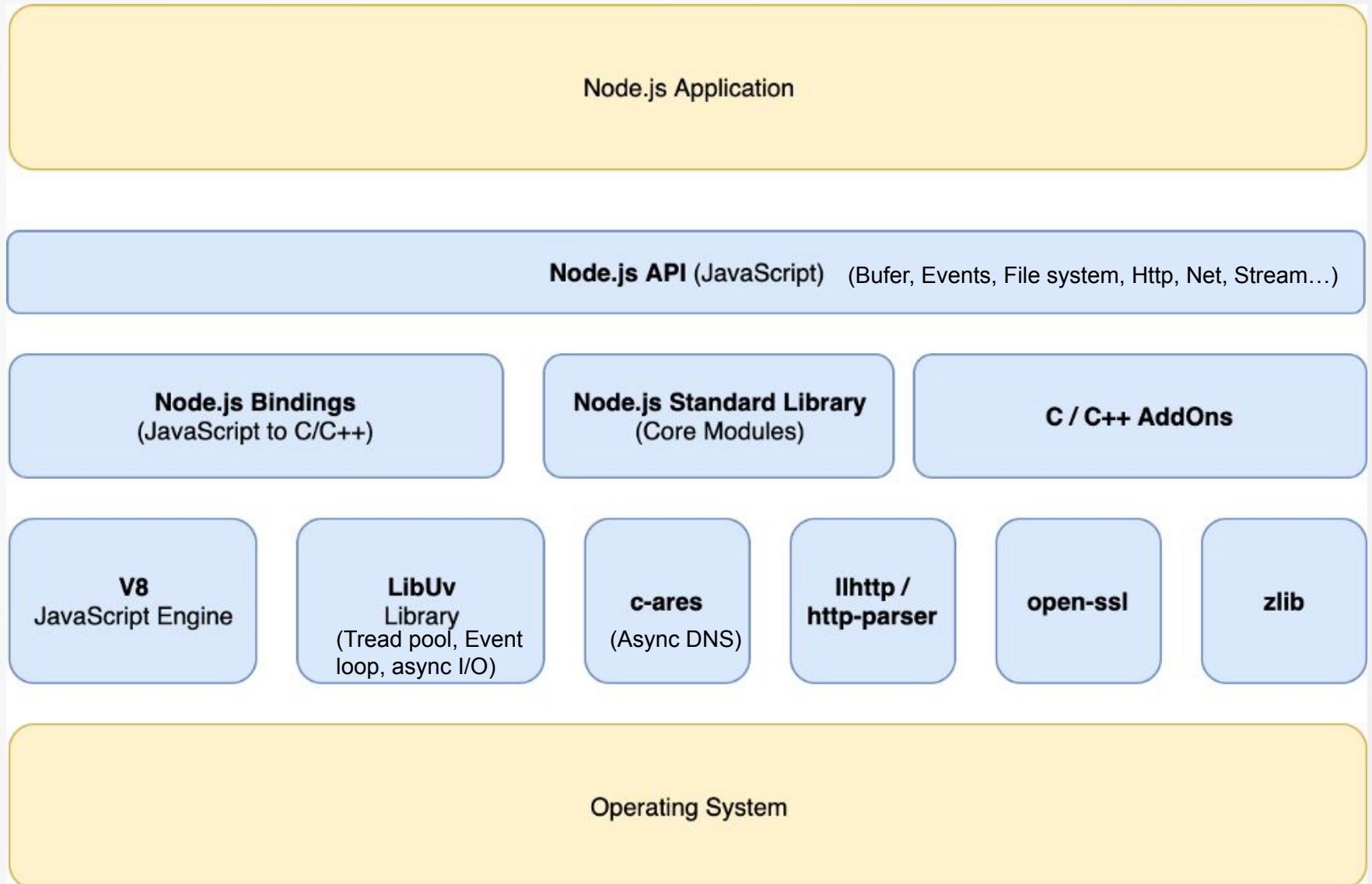
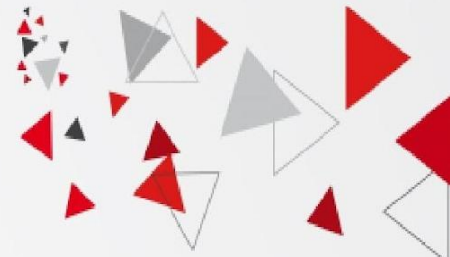


Les caractéristiques du Node Js

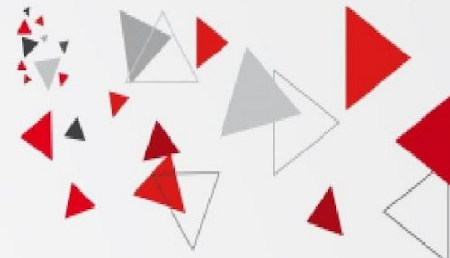


- Traitement asynchrone
- Model I/O non bloquant
- Modèle basé sur les événements (Event-driven Model)
- Rapidité et efficacité (Moteur V8)
- Utilise le gestionnaire des packages NPM (Node package Manager)
- Multiplateforme (Web, mobile, desktop)

► Architecture technique



▶ Les modules Node JS



- Node JS traite le code JavaScript comme des modules CommonJS
- Le développement en Node JS est basé sur l'utilisation des modules (modules intégrés ou bien les modules tiers).
- Le système de modules permet de charger des bibliothèques dans une application cela permet de développer des applications extensibles.
- Les modules peuvent être des modules pour se connecter à une base de données, créer des serveurs ou autre.

► Les modules Node JS



Exemple

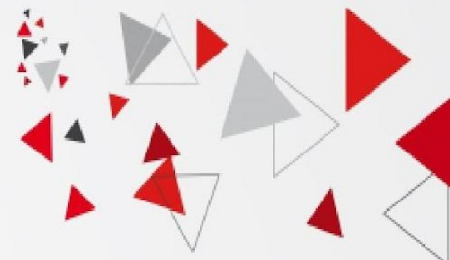
```
const fs = require ('fs');  
try {  
  fs.appendFileSync('message.txt', 'Bonjour Node JS');  
  console.log('Message "Bonjour Node JS"! ajouté');  
} catch (err) {  
  console.log(err);  
}
```

const : permet de créer une constante accessible seulement en lecture.

appendFileSync: permet d'ajouter des données à un fichier, en créant le fichier s'il n'existe pas encore.

Console.log(...): permet d'afficher un message dans la console.

▶ Les modules Node JS



- La fonction « **require** » permet aussi d'importer des fichiers créés par le développeur.
- Dans ce cas il faut mentionner le chemin relatif du fichier dans la fonction « **require** ».

```
const file = require ('../src/ressources.js');
```

- Il faut exporter le fichier « ressources.js », comme étant module externe afin d'être lisible par l'application.

► Les modules Node JS

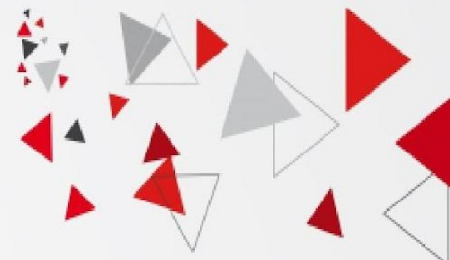


- Il faut exporter le fichier « **ressources.js** », comme étant module externe afin d'être lisible par l'application.

Exemple

```
const todo = function () {  
  console.log('Doing some tasks...')  
}  
module.exports = todo
```

► Les modules Node JS



Node.js ne partagent pas les variables □ les variables créées dans un script ne sont pas accessibles dans un autre script. La seule façon de partager des valeurs entre les scripts est d'utiliser « **require** » avec « **module.exports** ».

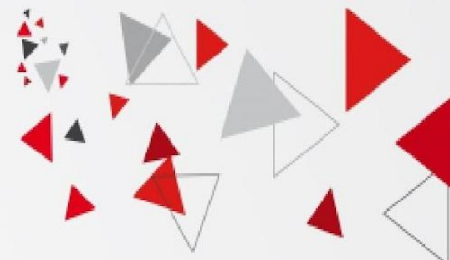


► NPM (Node Package Manager)

- NPM est le gestionnaire de paquets officiel de Node.js , il permet de lister les paquets disponibles de les télécharger, installer, mettre à jour et les désinstaller
- Il existe de nombreux gestionnaires de paquets, chacun dédié à un langage ou à un framework particulier : Maven (JAVA), Bundler (Ruby), Composer (PHP), YARN (JavaScript)...
- Exemples d'utilisation de npm
`npm install express` ou `npm i express`: (Express est le framework serveur de référence pour les applications web Node.js)

[<https://www.npmjs.com/>]

▶ Exemple 1



- La manière "classique" dans Node.js est de lire le contenu d'un fichier d'une manière non bloquante et asynchrone. Autrement dit, pour dire à Node de lire le fichier, puis d'obtenir un rappel(**callback**) lorsque la lecture du fichier est terminée.

⇒ traiter plusieurs demandes en parallèle.

→ Installer le package 'read-file' depuis npm: **npm i read-file**

→ `var read = require('read-file');`

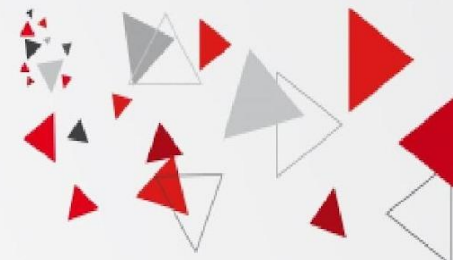
Mode Asynchrone:

```
→ read('file.txt', 'utf8',  
    function(err, buffer) {  
        console.log(buffer);  
    });
```

Mode Synchrone

```
read.sync('file.txt', 'utf8'); ou  
read('foo.txt', {encoding: 'utf8'});
```

► Exemple 2



- Le module **fs** est livré **nativement** avec Node js. Pour l'inclure on utilise la fonction **require**.
- Utilisons la méthode **readFile** du module **fs** :

Lire le fichier de manière asynchrone (non bloquante):

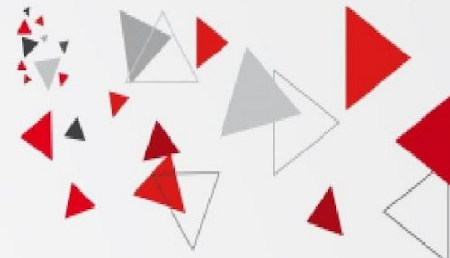
Soit un fichier “non-blocking-read-file.js” :

```
var fs = require('fs');

fs.readFile('filePath', 'utf8', function(err, content) {
  console.log(content);
});

console.log('after calling readFile');
```

▶ Exemple 2



- Pour lire un fichier, utilisons pour cet exemple la méthode **readFileSync** de la classe **fs** :

Lire le fichier de manière synchrone (bloquante):

Soit un fichier “blocking-read-file.js” :

```
var fs = require('fs');
```

```
var content = fs.readFileSync('filePath', 'utf8');  
console.log(content);
```

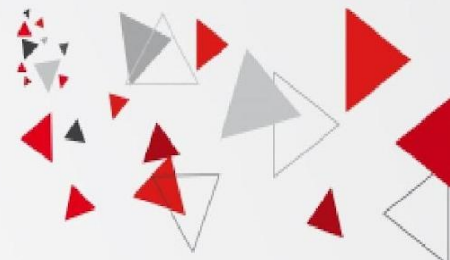
Dans le module **fs**, toutes les fonctions sont asynchrones par défaut (elles ne bloquent pas le code et fonctionnent en arrière-plan) mais peuvent aussi fonctionner synchroniquement. Il suffit d'ajouter Sync au nom de la fonction. (`appendFile()`, `open()`, `writeFile()`)



Conclusion

- NodeJS exécute le code JavaScript sur un seul thread. Si nous utilisons la version **Sync** d'une fonction d'**E / S**, ce thread est **bloqué** en attente d'E / S et ne peut rien faire d'autre.
- Si nous utilisons la version **asynchrone**, les **E / S** peuvent **continuer** en arrière-plan pendant que le thread JavaScript se poursuit avec d'autres tasks.

► Références



1. <https://www.ecma-international.org/publications-and-standards/standards/ecma-262/>
2. <https://medium.com/codingtown/ecmascript-2020-aka-es-11-9c547f69d96f>
3. <https://programming.vip/docs/new-features-of-es6-es7-es8-es9-and-es10.html>
4. <https://javascript.info/>
5. <https://nodejs.org/en/docs/>
6. <https://nodejs.dev/learn>
7. https://www.w3schools.com/nodejs/nodejs_intro.asp
8. <https://www.npmjs.com/>