



Alphorm.com

Formation Les fondamentaux de **NodeJS**

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™ ©

Plan

- Présentation du formateur
- Plan de formation
- Objectifs de la formation
- Public concerné
- Les possibilités de NodeJS
- Les connaissances requises



Formation NodeJS, les fondamentaux

alphorm.com™ ©

Présentation du formateur

Edouard FERRARI

- contact@ferrari.wf
- Développeur full stack chez Summview
- Mission de conseil, d'architecture et de migration
- Mes références :
 - LinkedIn : <https://fr.linkedin.com/in/edouardferrari>
 - Alphorm : <http://www.alphorm.com/formateur/edouard-ferrari>
 - Github : <https://github.com/didouard>



Plan de la formation

- **Présentation et introduction de la formation**
 - Présentation de la formation
 - Scénario de la formation
 - Installation de NodeJS
 - Premier projet "Hello World"
 - Les IDE
- **Rappel des bonnes pratiques JavaScript**
 - Visibilité des variables
 - Les fonctions et les objets
 - Héritage
 - This / Context, Bind, Call, Apply
- **Introduction à NodeJS**
 - Origine de NodeJS
 - Le moteur d'interprétation Chrome V8
- **Architecture de Node.js**
 - Programmation par callbacks
 - Synchrone vs Asynchrone
 - Programmation par callback avancée
- **Module et gestion des dépendances**
 - L'approche modulaire
 - NPM: Le manager des modules
 - Le fichier package.json en détail
 - Publier un module sur NPM
 - Modules: Process, OS, Path et Util
 - Modules : Buffer / File System / ReadLine / Stream
 - Module : Console / Error / Timer
 - Module : Events / EventEmitter2 / EventEmitter3
 - Module : URL / HTTP / Https / Net / UDP
 - Module : Child Processes / Cluster

Objectifs de la formation

- Un court rappel du langage JavaScript.
- Voir comment NodeJS fonctionne
- Étude du développement par évènement, par callback et par fonction asynchrone.
- Bien comprendre les modules natifs de NodeJS.
- Comprendre la philosophie globale de NodeJS

Public concerné

- À qui s'adresse cette formation :
 - Aux étudiants
 - Aux développeurs
 - Aux chefs de projet
 - Aux amoureux des nouvelles technologies
 - Ceux qui veulent découvrir l'**event coding** et l'asynchrone
 - Ceux qui ont besoin d'une architecture robuste, scalable et modulaire
 - Pour un projet orienté Web ou pour un projet back
- Les possibilités sont infinies et immenses !

Les possibilités de NodeJS

- Serveur et site internet complexe
- Application console
- Service réseau sur mesure (Proxies, gestion réseau, ...)
- Application avec GUI (Graphical User Interface)
- Outils en ligne de commandes
- APIs
- Support des sockets
- Chargement du code node au premier appel HTTP

Les connaissances requises

- Au minimum :
 - Connaissance de base en Javascript
 - Autodidacte
 - Connaissance globale en programmation
 - Connaissance en réseau / internet
- Les plus :
 - Connaissance en asynchrone
 - Connaissance en event-driven architecture

Liens et ressources

- NodeJS API : <https://nodejs.org/api/> !!!
- Projet github en nodejs :
<https://github.com/search?utf8=%E2%9C%93&q=nodejs>
- Stackoverflow : <http://stackoverflow.com/tags/node.js>
- Nodecloud : <http://www.nodecloud.org/>

Are you ready ? 😊





Alphorm.com

Présentation et introduction
de la formation

Scénario de la formation

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Formation
- Side project : Blackhole



Formation NodeJS, les fondamentaux

alphorm.com™©

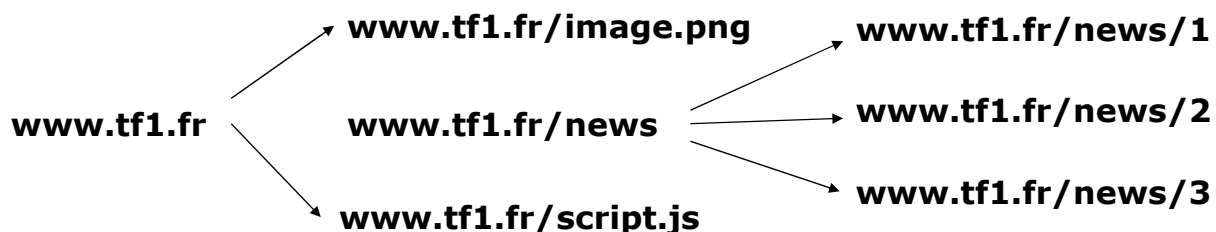
Formation

- La formation est une première partie de la formation complète de NodeJs.
- Dans cette première partie, nous verrons ensemble :
 - Comment installer NodeJS et l'utiliser
 - Un rapide rappel de JavaScript
 - Comprendre les origines de NodeJS et sa philosophie
 - L'architecture de NodeJS, comment composer nos futurs projets
 - Les modules natifs
- La seconde formation verra les modules plus utilisés et les plus connus. Tout pour être 100% indépendant en NodeJS.

Side Project : Blackhole

- Tout au long de la formation, nous travaillerons sur un projet
- Ce projet a pour but de reproduire le comportement d'un **aspirateur de site**.
- Avec NodeJs, ce genre de programme est très facile à développer et peut être fait en quelques heures.

Side Project : Blackhole



Ce qu'on a couvert

- Les cours, les exercices et le "side-project" sont complémentaires.
- Nous allons voir beaucoup de code.
- Vous pouvez télécharger toutes les sources dans votre espace utilisateur sur Alphorm.com
- Prochaine vidéo: L'installation de NodeJs





Alphorm.com

Présentation et introduction
de la formation

Installation de NodeJS

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™ ©

Plan

- Les versions
- Installation avec les installeurs
- Installation avec les sources



Formation NodeJS, les fondamentaux

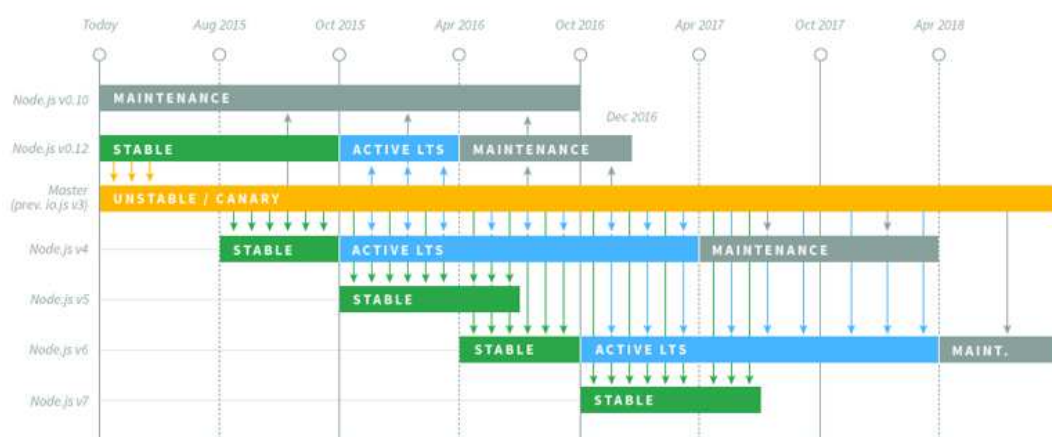
alphorm.com™ ©

Les versions

- Depuis le 12 octobre 2015, NodeJS LTS « Long Time Support » est disponible.
- 18 mois de support, soit jusqu'en avril 2017
- Cette année sera en **version 4.x.x**
- L'année prochaine en octobre 2015, une LTS 5.x.x sortira

Les versions

Node.js Long Term Support Release Schedule



COPYRIGHT © 2015 NODESOURCE, LICENSED UNDER CC-BY 4.0

Installation avec les installateurs

- Site officiel : <https://nodejs.org>
- <https://nodejs.org/en/download/>

Installation à partir des sources

- Site officiel : <https://nodejs.org>
- <https://nodejs.org/en/download/>

- Pourquoi ?
- Ubuntu / Debian

```
apt-get install make g++ libssl-dev git
```

Ce qu'on a couvert

- Les versions
- Installation avec les installeurs
- Installation avec les sources



Alphorm.com

Présentation et introduction
de la formation

Premier projet
« Hello World »

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Plan

- REPL
- Premier programme
- Premier programme HTTP



REPL

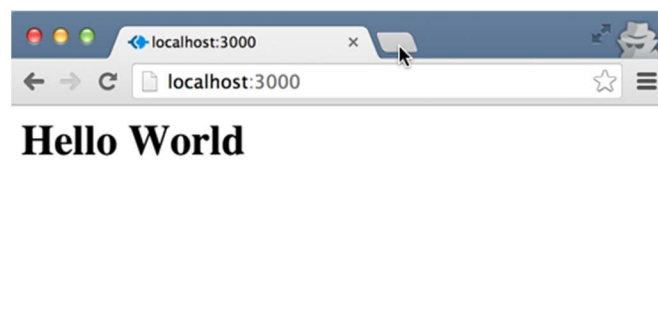
- NodeJS REPL « Read-Eval-Print-Loop »

```
$ node  
> console.log('Hello World');  
Hello World
```

Premier programme

Hello World!

Premier programme HTTP



Ce qu'on a couvert

Nous avons appris :

- Comment tester rapidement un portion de code
- Comment lancer un plus gros projet



Formation NodeJS, les fondamentaux

alphorm.com™©



Alphorm.com

Présentation et introduction
de la formation

Les IDEs

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Qu'est-ce qu'un IDE ?
- Les IDE les plus connus / utilisés
- 9 Cloud



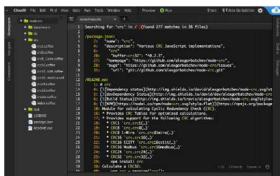
Qu'est ce qu'un IDE?

- « Integrated Development Environment »
- Est un software qui permet de faciliter le développement d'un programme.
- Il est composé :
 - D'un éditeur de code source
 - D'un « builder » automatique
 - D'un debugger
- La plupart des IDEs font de l'auto complétion.
- D'autres fonctionnalités : Etudes de l'exécution, Heap management, ...

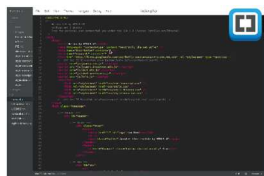
Les IDE

Les clients lourds (Desktop IDE)

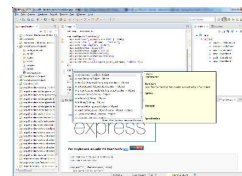
Atom



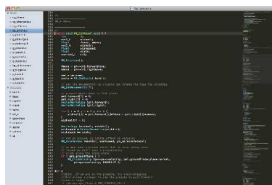
Brackets



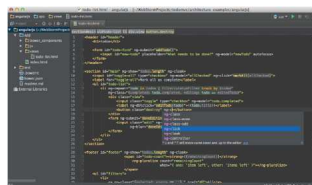
Eclipse IDE



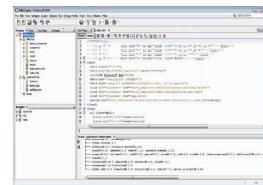
Sublime Text



JetBrains IntelliJ IDEA



Netbeans



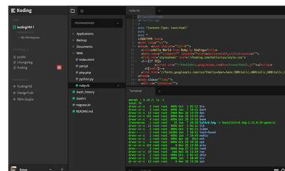
Les IDE

IDE en ligne (Online code editors)

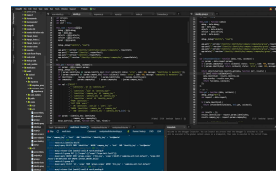
Codeanywhere.com



Koding.com



9Cloud



9 Cloud

Allez sur <http://c9.io>

- Le principe
- Le pricing
- Création d'un compte
- Décomposition de l'interface & debug
- Installation en local

Ce qu'on a couvert

- Nous sommes prêt pour réellement commencer la formation !
- Nous savons comment lancer et tester nos programmes
- Nous avons un IDE





Alphorm.com

Rappel des bonnes pratiques
Javascript

Visibilité des variables

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Les scopes
- Les closures
- Les problèmes liés aux boucles



Formation NodeJS, les fondamentaux

alphorm.com™©

Les scopes

- Un scope est un ensemble où les variables et les fonctions sont accessibles entre elles.
- Un scope peut être local ou global

Les closures

- Les closures sont des variables qui vont pouvoir être accédé depuis les fonctions définies en dessous d'elle-même.

Les problèmes liés aux boucles

- Dans le cas de fonction asynchrone, le temps que la **callback** soit rappelée, l'**itérateur** de la boucle est à son maximum.

Ce qu'on a couvert

- Nous avons :
 - Les scopes.
 - Vu que JavaScript nous permet d'accéder aux variables de fonctions parents grâce aux closures.
 - Vu comment gérer les problèmes liés aux boucles.

Prochaine vidéo :

- Les fonctions et les objets





Alphorm.com

Rappel des
bonnes pratiques Javascript

Les fonctions et les objets

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Les fonctions
 - basiques
 - Anonymes
- Les objets
 - Natifs
 - Instanciations
 - Constructeurs
 - Énumérer ses propriétés



Formation NodeJS, les fondamentaux

alphorm.com™©



Les fonctions

Basiques



Les fonctions

Anonymes

Les objets

Les objets natifs

Les objets

Les objets instanciés

Les objets

Les constructeurs

Les objets

Énumérer ses propriétés

Ce qu'on a couvert

- Nous avons passé en revue les fonctions et les objets en JavaScript
- Prochaine vidéo sur l'héritage



Alphorm.com

Rappel des
bonnes pratiques Javascript

L'héritage

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Plan

- Les instances
- Les prototypes
- Héritage par copie des prototypes
- Héritage par prototypes chaînés



L'héritage

Les instances

L'héritage

Les prototypes

L'héritage

Héritage par copie des prototypes

L'héritage

Héritage par prototypes chaînés

Ce qu'on a couvert

- L'héritage et ses limites
- Prochaine vidéo, les contextes.





Alphorm.com

Rappel des
bonnes pratiques Javascript

This / Context, Bind, Call, Apply

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Qu'est-ce que le contexte et 'this' ?
- Bind
- Apply / Call



Formation NodeJS, les fondamentaux

alphorm.com™©

Qu'est-ce que le contexte et 'this' ?

- Le contexte est un objet spécifique à JavaScript qui ne peut être instancié et qui reçoit, en début d'interprétation du code, une série de propriétés utilisables à travers l'ensemble du code.
- À chaque instantiation d'objet, crée un nouveau contexte.
- On peut accéder au contexte grâce au mot '**this**'

Bind

- Avec JavaScript, la fonction **bind()** permet d'attacher un **contexte** à la fonction appelée.
- Mais aussi de paramétrer la fonction appelée avec des arguments en dur.

Apply / Call

- Apply et Call sont deux fonctions qui peuvent être appelées à chaque fonction.
- Elles permettent d'appeler la fonction avec les arguments en argument.

```
fn.call( contexte, arg1, arg2,... );  
fn.apply( context, [ arg1, arg2,... ] );
```

Ce qu'on a couvert

- Nous avons fini avec les importants rappels sur le JS.
- Nous sommes prêts pour passer à NodeJS !





Alphorm.com

Introduction à NodeJS

Origine de NodeJS

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Histoire de NodeJS
- Description de NodeJS
- Avantages



Formation NodeJS, les fondamentaux

alphorm.com™©

Histoire de NodeJS

- 2009 : créé par **Ryan Dahl**
 - NodeJS utilise la V8 engine pour executer le code js
- 2011 : NPM est créé pour publier et partager les librairies de la communauté. **Windows** travaille avec NodeJS pour le rendre compatible
- 2012 : **Isaac Schlueter** passe en leader du projet
- 2014, Janvier : **Timothy J. Fontaine** devient nouveau leader
 - Décembre : **Fedor Industry** fork NodeJS, création de io.js
- 2015, merge de NodeJS et io.js



Descriptions

- Open-source
- Développé en C/C++
- Cross platform : OS X, Microsoft Windows, Linux, FreeBSD, NonStop, IBM AIX, IBM System z and IBM i.
- Pour développer en « server-side »
- Se base de l'interpréteur JS : V8 de Google
- Utilisé par de grand nom: IBM, Microsoft, Yahoo!, Walmart, Groupon, SAP, LinkedIn, Rakuten, PayPal, Voxer, GoDaddy, ...

Les avantages

- NodeJS est basé sur une architecture :
 - « event-driven »
 - « non-blocking I/O API »
- NodeJS est conçu pour optimiser les possibilités et les évolutions d'une application web en temps réel.
- Les possibilités de NodeJS sont sans cesse augmentées grâce à sa grande communauté.

Ce qu'on a couvert

À retenir :

- NodeJs et io.js merge durant l'été 2015
- NodeJs utilise le moteur de Google V8
- NodeJs est basé sur une architecture d'évènement et les IO ne sont pas bloquants

Prochaine vidéo :

- Le moteur V8





Alphorm.com

Introduction à NodeJS

Le moteur V8

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Histoire et description de V8
- Comment V8 fonctionne ?
- Les optimisations
- Comment V8 compile le code JS ?



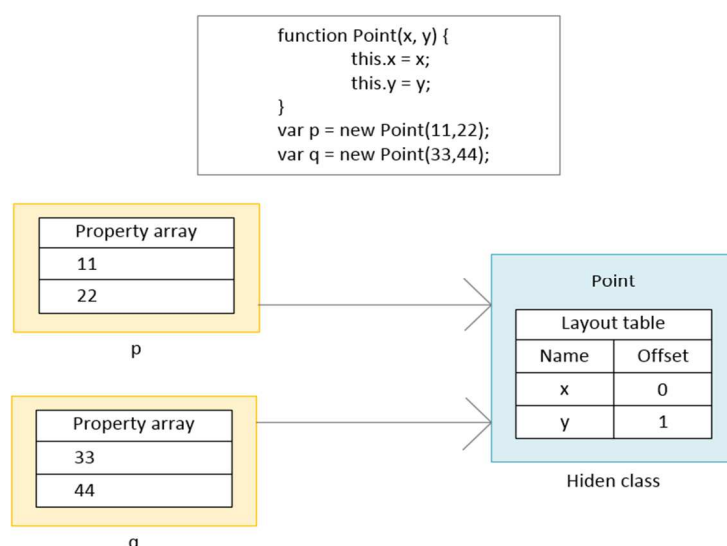
Formation NodeJS, les fondamentaux

alphorm.com™©

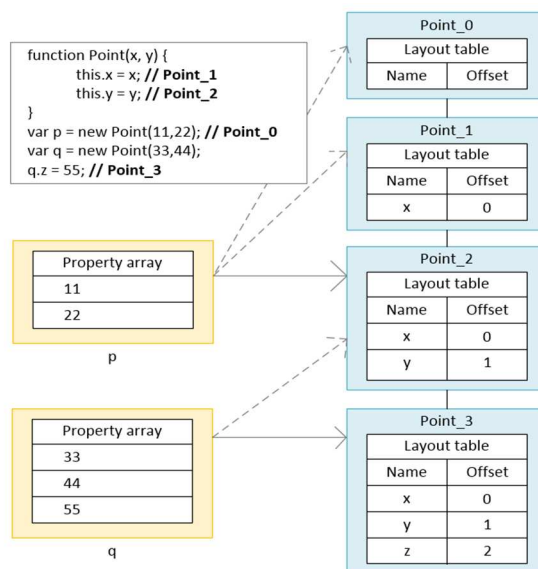
Histoire et description de V8

- Créé par **Google** en **2008** avec le projet Chromium. Depuis est utilisé par de nombreux projets comme **MongoDb** ou **NodeJs**
- Open-source et codé en C++
- V8 compile du Javascript en code machine avant de l'exécuter.
- Le code compilé est en plus optimisé et reoptimisé dynamiquement en "runtime"
- V8 gère **ECMAScript 6**

Comment V8 fonctionne?



Comment V8 fonctionne?



Les optimisations

Deux principales optimisations :

- **Tagged Values** : Pour éviter les effets d'expansion vers les doubles qui est chère en ressources.
- **Les tableaux** : Pour aider V8 à optimiser les tableaux, évitez les trous. Ajouter les nouveaux éléments à la suite. De plus la préallocation étant lourde, préférez augmenter/ajouter les éléments un à un plutôt que tout d'un coup.

Comment V8 compile le code JS ?

- V8 a deux **compilateurs** !
 - Un compilateur complet « **Full** »
 - Un compilateur **d'optimisation**

Ce qu'on a couvert

- Nous avons vu comment optimiser notre code pour V8.
- Pour plus d'informations :
 - Google I/O 2012 "Breaking the JavaScript Speed Limit with V8" avec Daniel Clifford : [vidéo](#) et [slides](#).
 - V8: Un moteur JavaScript open-source: [vidéo](#) de Lars Bak, V8 core engineer.
 - Un poste du blog de Nikkei Electronics Asia: [Why Is the New Google V8 Engine So Fast?](#)
- Prochain chapitre : L'architecture de NodeJS





Alphorm.com

Architecture de NodeJS

Programmation par callback

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™ ©

JS Plan

- Qu'est-ce que la programmation par **callback** ?
- Comment cela fonctionne ?
- « Error-first » callback



Formation NodeJS, les fondamentaux

alphorm.com™ ©

Qu'est-ce que la programmation par callback ?

- Les **callbacks** sont des fonctions qui sont appelées à la fin d'une tâche.
- Les **callbacks** peuvent aussi être désignées par abus de langage, une fonction anonyme passée en argument.
- Une très grande partie du code de NodeJS est développé avec des **callbacks**.

Comment cela fonctionne ?

- Imaginons l'algorithme suivant :
 - Je lance une recherche à partir d'une fonction avec les arguments :
 - Les données et le filtre
 - La callback
 - Dans la fonction, quand j'ai fini de traiter les données, j'appelle la fonction callback avec les potentielles erreurs et le résultat.

« Error-first » callback

- La communauté de NodeJS s'est mis d'accord pour une standardisation des callbacks :
 - Le premier argument est l'erreur.
 - Le second est le résultat.
- C'est important de garder cette normalisation pour faciliter la réutilisation de votre code.

Ce qu'on a couvert

- Nous avons compris comment fonctionnent les **callback**.
- La prochaine vidéo sera sur les méthodes **synchrones et l'asynchrone**.





Alphorm.com

Architecture de NodeJS

Synchrone vs Asynchrone

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Les méthodes synchrones
- Les méthodes asynchrones
- Synchrone VS Asynchrone



Formation NodeJS, les fondamentaux

alphorm.com™©

Les méthodes synchrones

- Chaque tâche est traitée une après l'autre

```
1 On téléchargement du file 1
2 On attend que la fonctionne retourne le résultat du fichier 1
3 On téléchargement du file 2
4 On attend que la fonctionne retourne le résultat du fichier 2
```

- Tant que le **fichier 1** n'est pas totalement téléchargé, le script bloque.
- L'exécution est complètement bloquée. La méthode de contournement est de développer un programme multithreadé.

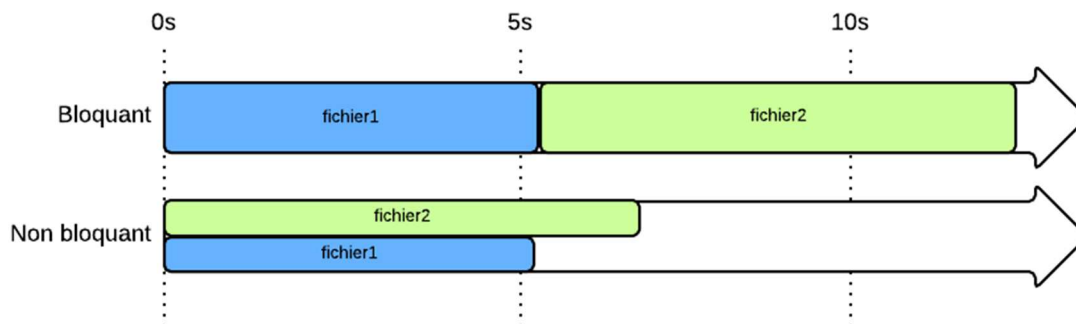
Les méthodes asynchrones

- Le programme lance plusieurs tâches en même temps, on est averti quand les tâches sont finies.

```
1 On lance le téléchargement du file 1
2 On lance le téléchargement du file 2
3 On attend que la tâche 1 et 2 finisse
```

- Malgré que le programme soit en train de télécharger les fichiers, le programme peut continuer à faire d'autres choses.

Synchrones vs Asynchrones



Ce qu'on a couvert

- Nous avons vu qu'elles sont les différences entre un code **Synchrone** et **Asynchrone**.
- Prochaine vidéo :
 - Compréhension de l'« **event loop** »





Alphorm.com

Architecture de NodeJS

L'Event Loop

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

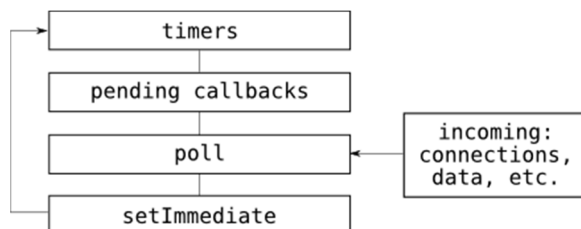
- Qu'est-ce que l'Event Loop ?
- Fonctions synchrones / asynchrones



Formation NodeJS, les fondamentaux

alphorm.com™©

Qu'est ce que l'évent loop ?



- L'évent loop est le cœur de V8 :
 - **Timer** : les fonctions de « timer » sont appelées (setInterval, setTimeout)
 - **Pending callbacks** : V8 va exécuter les fonctions qui sont en attente.
 - **Poll** : V8 vérifie s'il n'y a pas de nouvel événement sur les « files descriptors »
 - **setImmediate** : la fonction setImmediate est appelée.

Fonctions synchrones / asynchrones

- Nous allons voir ensemble ce qu'implique vraiment le synchrone et l'asynchrone dans notre code.

Ce qu'on a couvert

- Nous avons vu que malgré que des fonctions appellent des callback, elles ne sont pas forcément asynchrones.
- Nous avons bien vu que tant que nous ne rendons pas la main, le pointeur d'exécution reste dans notre code.
- Prochain chapitre:
 - Programmation par callback avancée



Alphorm.com

Architecture de NodeJS

Programmation par callback avancée : **Projet asyncMap**

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Plan

- Révisions
- Projet 1 : asyncMap
- Let's dev !



Révisions

- Les callbacks, comment elles fonctionnent et la norme « **Error-First** ».
- Les différences entre le synchrone et l'asynchrone.
- L'évent loop, comment nodejs fonctionne et ce qu'il faut prévoir

Projet 1 : asyncMap

- Sujet :
 - Télécharger 3 images à la fois.
 - Une fois que les 3 images sont téléchargées, afficher un message avec la taille totale des 3 images.
 - Les téléchargements doivent être asynchrones.
- Sujet: 'FR_235_04_03/Exercices/asyncMap.js'

Projet 1 : asyncMap

- Algorithme :
 - On crée un **Array** qui contient nos 3 images.
 - On appelle une fonction qui gèrera les 3 téléchargements.
 - On lance les 3 téléchargements, dans leur callback, on compte le nombre de retours et on garde la taille de l'image téléchargée.
 - Une fois qu'il y a eu 3 callbacks de retour, on appelle la callback pour indiquer que l'on a téléchargé nos 3 images, et on affiche le résultat.
- Correction : 'FR_235_04_03/Exercices/asyncMap_corrigé.js'

Projet 1 : asyncMap

A vous de coder !

Projet 1 : asyncMap

Correction

Ce qu'on a couvert

- Nous avons écrit notre premier vrai projet NodeJS.
- Dans le cas d'une boucle + fonction asynchrone, il ne faut pas oublier de réenglober la fonction afin de ne pas écraser les closures !
- Prochain Vidéo:
 - Programmation par callback avancée : Projet asyncWaterfall



Alphorm.com

Architecture de NodeJS

Programmation par callback avancée : **Projet asyncWaterfall**

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Plan

- Projet 2 : asyncWaterfall
- Let's dev !



Projet 2 : asyncWaterfall

- Sujet :
 - Exécuter une suite de fonctions consécutivement et appeler la callback une fois que la dernière est appelée.
 - Le projet DOIT gérer les fonctions asynchrones.
- Sujet : 'FR_235_04_03/Exercices/asyncWaterfall.js'
- Correction : 'FR_235_04_03/Exercices/asyncWaterfall_corrige.js'

Projet 2 : asyncWaterfall

- Algorithme :
 - On crée un array contenant 2 fonctions
 - Fonction1 : télécharge une image avec **http.get**
 - Fonction2 : donne la taille en octet de l'image
 - En récursif **asyncParallel** :
 - Je prends la première fonction et je la retire de l'Array
 - Je l'exécute en créant une callback
 - Dans la callback, s'il reste des jobs à faire, je rappelle **asyncParallel**
 - Une fois tous les jobs exécutés, on appelle la première callback avec le dernier résultat

Projet 2 : asyncMap

A vous de coder !

Projet 2 : asyncMap

Correction

Ce qu'on a couvert

- Nous avons vu comment gérer plusieurs fonctions asynchrones du même type.
- Nous avons fait un rapide rappel sur
 - Le keyword 'arguments'.
 - Sur la fonction Apply.
- Prochain chapitre:
 - Les modules natifs !





Alphorm.com

Modules et gestion
de dépendances

L'approche modulaire

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™ ©

Plan

- NodeJS et ses modules
- Les modules natifs
- Créer ses propres modules



Formation NodeJS, les fondamentaux

alphorm.com™ ©

NodeJs et ses modules

- Le cœur de NodeJS est petit, mais les modules qui le composent sont extrêmement nombreux.
- Un module encapsule le code d'un morceau fonctionnel du projet.
- Un module peut exporter un module.
- Le code NodeJS a besoin de la fonction « **require** » pour importer un nouveau module au code.
- On exporte un module en copiant l'objet dans « **module.exports** » ou dans « **exports** » directement.

NodeJs et ses modules

- Il y a **deux types de modules** :
 - Module **natif**, présent par défaut dans NodeJS.
 - Module **communautaire**, qu'il faut récupérer avec NPM.
- Un module peut exporter :
 - Une string
 - Un nombre
 - Un objet
 - Une fonction
 - Une « classe »
 - Un objet instancié

Les modules natifs

- NodeJs met à disposition des modules natifs.
- Leur liste et leurs API sont disponibles sur <https://nodejs.org/api/>

Créer ses propres modules

- Étude de cas
 1. Le plus simple des modules
 2. Un module doit avoir un but précis
 3. Exporter une string, un nombre et un objet
 4. Exporter une fonction anonyme
 5. Exporter une fonction nommée
 6. Exporter une classe
 7. Exporter un objet
- Bonus

Modulariser le projet : « Blackhole »

- Reprenons le code du projet « asyncMap » et divisons le code comme ci-dessous :
 - Module « **async.js** »
 - Module « **request.js** »
- Copions la liste des images dans un fichier de configuration "config.json".

Ce qu'on a couvert

- Nous avons vu comment créer nos modules, comment les exporter et les require.
- Les projets NodeJS sont créés autour de ce « Module Patherne ».
- Prochain chapitre:
 - NPM: Le manager des modules





Alphorm.com

Modules et gestion de dépendances

NPM : Le manager des modules

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Qu'est-ce que NPM ?
- Se familiariser avec NPM
- Module local
- Module global
- Usage



Formation NodeJS, les fondamentaux

alphorm.com™©

Qu'est-ce que NPM ?

- NPM est un gestionnaire de module pour NodeJs.
- NPM gère les dépendances des modules automatiquement.
- NPM est inspiré de PEAR (PHP) et CPAN (PERL)
- NPM fonctionne en ligne de commande.
- NPM permet de rechercher, installer et gérer.
- NPM est installé avec NodeJS.

Se familiariser avec NPM

- Ajouter l'autocomplétion (linux)
 - `$> npm completion >> ~/.bashrc`
 - `$> source ~/.bashrc`
- `npm help`
- `npm help <subcommand>`

Module local

- Les modules locaux sont des modules téléchargés et installés au **niveau** de votre **projet** :
 - npm install underscore
 - Installé dans ./node_modules/
 - Il suffit de require('module-name')
- Exemple :
 - npm install underscore
 - var underscore = require('underscore');

Module global

- Les modules globaux sont installés dans
 - /home/<user>/.npm/versions/node/v4.2.1/bin/<project>
- Les modules **ne peuvent pas** être accédés avec require()
- Mais ils sont accessibles depuis un **terminal**.
- Exemple :
 - \$> npm install -g greedy-snake
 - \$> greedy-snake

Usage

- Lister les modules
 - \$> npm ls
 - \$> npm ls -g
- Savoir où se trouve le répertoire node_module
 - \$> npm root
- Rechercher un module
 - \$> npm search <Module name>
- Désinstaller
 - \$> npm uninstall <Module name>
 - \$> npm uninstall -g <Module name>

Usage

- Pour « linker » un module global à un projet local
 - \$> npm link <Module name>
- Pour lister les paquets qui ne sont pas à jour
 - \$> npm outdated
- Pour mettre à jour les modules
 - \$> npm update
- Pour changer la configuration de NPM
 - \$> npm config

Ce qu'on a couvert

- Nous avons découvert comment utiliser le gestionnaire de module de NodeJS : NPM
- Sans NPM, NodeJS ne serait pas autant populaire.
- La prochaine vidéo :
 - Package.js
 - Publier un module sur NPM



Formation NodeJS, les fondamentaux

alphorm.com™©



Alphorm.com

Modules et gestion
de dépendances

Le fichier package.json

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- A quoi sert le fichier package.json ?
- Les différents champs
- Enregistrer les dépendances
- Les versions des modules



A quoi sert le fichier package.json ?

- Le fichier « package.json » résume toutes les informations pratiques de votre projet
- Pour générer le fichier :
 - `$> npm init`

Les différents champs

- **Name :**
 - Nom du projet
 - Limité à **214 caractères**
 - Ne peut commencer par un point ou un underscore.
 - Ne peut avoir de majuscule dans le nom
 - Ne pas ajouter « node » ou « js » dans le nom
 - Vu que le nom sera ajouté dans `require()`, autant que le nom soit court
 - Vérifier que le nom n'est pas déjà pris

Les différents champs

- **Version :** La version du projet. Obligatoire
- **Description :** Quelques mots pour décrire le projet. Ces mots seront ajoutés à la fiche de description de votre projet dans npm.
- **Keyword :** Sera utilisé par **npm search** pour retrouver votre module.
- **Homepage :** Pointe vers le site internet du projet.
- **Bugs :** Renseigner le bug tracker et / ou un email de contact.
- **Licence :** La licence du projet, ISC par défaut (<https://spdx.org/licenses/>)

Les différents champs

- **Author** : Les informations vous concernant

```
{ "name" : "Barney Rubble"  
  , "email" : "b@rubble.com"  
  , "url" : "http://barnyrubble.tumblr.com/"  
}
```

- **Repository** : Les informations pour accéder à votre projet

```
"repository" :  
  { "type" : "git"  
    , "url" : "https://github.com/npm/npm.git"  
  }
```

- Et plus encore !
 - <https://docs.npmjs.com/files/package.json>

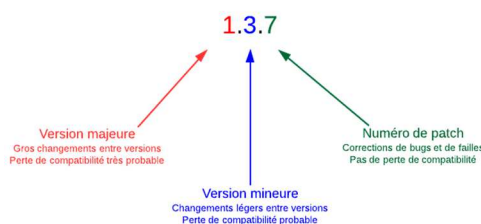
Enregistrer ses dépendances

- Imaginons que notre projet a besoin de plusieurs modules :
 - Async
 - Underscore
 - Moment
 - Mkdirp
- Solution rapide et pratique :
 - `$> npm install --save async`

Les versions des modules

- La version du projet ou des dépendances externes sont renseignées avec 3 chiffres :

```
"dependencies": {  
  "eventemitter3": "^1.1.1"  
}
```



Les versions des modules

- Pour les modules externes, on peut ajouter des flags pour préciser la version que l'on veut :
 - `<=1.2.3`
 - `=2.0.1`
 - `~1.2.3` va prendre toutes les versions jusqu'à 1.3.0 non compris
 - `^1.2.3` va prendre toutes les versions jusqu'à 2.0.0 non compris

Ce qu'on a couvert

- Désormais nous maîtrisons NPM et son fichier de « configuration »
- Nous avons vu les champs les plus importants
- Nous avons vu comment ajouter un module et le sauvegarder dans `package.json`
- Et nous avons compris comment renseigner les versions des modules externes
- La prochaine vidéo
 - Publier son propre module NPM



Alphorm.com

Modules et gestion
de dépendances

Publier un module
sur NPM

Site : <http://www.alphorm.com>

Blog : <http://blog.alphorm.com>



Édouard FERRARI

Formateur et Consultant indépendant

Contact : contact@ferrari.wf

Plan

- Pourquoi publier un projet sur NPM ?
- Créer un utilisateur NPM
- Ajouter le projet sur github
- Créer le module myasync
- Site project : Blackhole



Pourquoi publier un projet sur NPM ?

- Le rendre public votre module.
- Se faire aider pour l'améliorer ou pour le débbugger.
- Avoir de la visibilité sur internet ou pour faire valoir ses connaissances.
- Devenir célèbre et accéder à la gloire éternelle ... !

Créer un utilisateur NPM

- Première étape, créer un utilisateur NPM
 - `$> npm adduser`

Ajouter le projet sur github

- Sauvegarder le projet
- Avoir accès à un wiki pour son projet
- Avoir accès à un bug tracker

Créer le module myasync

- Créer le projet
 - Créer un dossier « **myasync-ferrari** » avec votre nom.
 - Copier le fichier **async.js**
 - Initialiser git
 - \$> git init
 - \$> git remote add origin <lien vers le repository>

Créer le module myasync

- Initialiser NPM
 - \$> npm init
 - **Name** : le nom de votre projet en minuscule (« myasync-eferrari »)
 - **Version** : 1.0.0
 - **Description** : Fonctions et modèles communs pour le code asynchrone
 - **Entry point** : async.js
 - **Test command** : mocha
 - **Git repository** : Normalement déjà renseigner, sinon prendre l'url de votre repository.
- En créant le projet dans cet ordre, les champs homepage et bugs doivent être remplis.

Publier le module

- Une seule commande :
 - \$> npm publish
- C'est tout ! Félicitation !

Site project : Blackhole

- Préparons notre projet, je vous propose de :
 - Créer un projet github pour sauvegarder et versionner le code
 - Initialisé git
 - Initialisé NPM
 - Ajouter notre module NPM « **myasync-xxxx** »
 - Adapter le code avec cette nouvelle disposition

Ce qu'on a couvert

- Nous avons vu ensemble comment créer un module et nous l'avons publié sur NPM
- Nous avons préparé notre projet Blackhole tel un vrai projet NodeJs !
- Prochaine vidéo :
 - Les modules natifs : process, os, path et util



Alphorm.com

Modules et gestion de dépendances

Modules : Process, Os,
Path, Util

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Plan

- Module : Process
- Module : OS
- Module : Path
- Module : Util



Module : Process

- Le module **process** est un objet global qui peut être accédé de n'importe où.
- C'est une instance d'**EventEmitter**.
- Il permet de gérer :
 - les états
 - les arguments
 - les signaux
 - et les informations du processus.
- Lien API : <https://nodejs.org/api/process.html>

Module : OS

- Le module OS est accessible à partir de `require("os")`
- Il permet de gérer quelques informations basiques sur l'os.
- Lien API : <https://nodejs.org/api/os.html>

Module : OS

Exercices

- Ecrire un script pour récupérer les informations suivantes :
 - L'architecture de la machine
 - Le nombre de CPU
 - Le hostname
 - Et la charge moyenne
- Correction : 'FR_235_05_04/Exercices/exercice_os_corrige.js'

Module : Path

- L'objet Path est accessible à partir de **require("path")**
- Il permet de manipuler les liens systèmes
- Lien API : <https://nodejs.org/api/path.html>

Module : Util

- L'objet Util est accessible à partir de **require("util")**
- Il permet de gérer beaucoup de différentes choses:
 - Construction de chaîne de caractères
 - Héritage
 - Inspection des objets
 - Et d'autres
- Lien API : <https://nodejs.org/api/util.html>

Module : Util

- Beaucoup de fonctions sont "**deprecated**" (obsoletes)
- En Février 2015, ils se sont rendu compte qu'un certain nombre de fonctions d'**util** ne retournaient pas de bonnes valeurs.
 - **Util.isObject** d'une fonction retourne 'false', alors qu'elle devrait retourner 'true'
- Lors de la réunion TSC "*Technical Steering Committee*" de Juin, vu qu'il n'était pas possible de modifier **util** sans créer d'important bugs dans les développements au sein de la communauté NodeJS. Ils ont décidé qu'ils supprimeraient les fonctions de test.
- En remplacement, il faut utiliser des modules de la communauté
 - **Exemple** : [core-util-is](#)

Ce qu'on a couvert

- Nous avons vu ensemble les 4 premiers modules natifs de NodeJS
- Vous savez désormais :
 - Comment gérer les arguments
 - Connaître le type de machine sur lequel votre script tourne
 - Gérer les Chemins
 - Et vous servir d'Util
- La prochaine vidéo sera sur :
 - Les modules : Buffer / FS / ReadLine / Stream





Alphorm.com

Modules et gestion de dépendances

Modules : Buffer / FS / ReadLine / Stream

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Module : Buffer
- Module : File System
- Module : ReadLine
- Module : Stream



Formation NodeJS, les fondamentaux

alphorm.com™©

Module : Buffer

- Le module **Buffer** est un objet global qui peut être accédé de n'importe où
- JavaScript gère très bien les strings, mais rien de spécial pour les données binaires. Par exemple : Les données provenant d'un service TCP.
- Les données brutes sont stockées dans des objets appelés "**Buffer**"
- Buffer ressemble à un Array (tableau) contenant des nombres.
- Un buffer a une taille fixe. Si la taille est modifiée, l'ensemble des données est recopié.
- Lien API : <https://nodejs.org/api/buffer.html>

Module : File System

- Tous les accès (I/O) sur les fichiers sont accessibles grâce au module "**FS**".
- Pour accéder au module, il faut faire un **require('fs')**.
- La plupart des fonctions existent en **Synchrone** et en **Asynchrone**.
- Lien API : <https://nodejs.org/api/fs.html>

Module : File System

Exercices

- À partir du fichier présent dans les ressources "lorem ipsum.txt"
- Lire le fichier en synchrone
- Lire le fichier en asynchrone
- Copier le fichier dans "lorem ipsum – copy.txt" en asynchrone
- Implémenter la fonction **watch** pour surveiller le dossier courant
 - `fs.watch(".", ...)`
- **Correction :** 'FR_235_05_05/Exercices/exercice_fs_corrige.js'

Module : ReadLine

- Le module **ReadLine** permet d'interagir avec les "file descriptor".
- <https://nodejs.org/api/readline.html>

Module : ReadLine

Exercices

- Ouvrir le fichier 'FR_235_05_05/Exercices/exercice_readline.js'
- Enregistrer tout ce qui est écrit dans le terminal dans un fichier "transcript.txt"
- Correction : 'FR_235_05_05/Exercices/exercice_readline_corrigé.js'

Module : Stream

- Stream est une implémentation de divers objets dans NodeJS.
- Les streams peuvent être 'readable' et 'writable', ou les deux.
- Tous les streams sont des instances d'EventEmitter.
- Lien API : <https://nodejs.org/api/stream.html>

Module : Stream

Exercices

- Ouvrir le fichier 'FR_235_05_05/Exercices/exercice_stream.js'
- Coder la fonction 'requestAndSave' qui téléchargera le contenu d'une URL et le stocker dans un fichier.
- Essayer de l'exercice faire avec les **streams**.
- Correction : 'FR_235_05_05/Exercices/exercice_stream_corrigé.js'

Ce qu'on a couvert

- Nous avons appris comment maîtriser complètement les fichiers et les stream sur NodeJS.
- Énormément de programmes vont utiliser ces modules natifs, c'est important de bien les maîtriser.
- Prochaine vidéo:
 - Module : Console / Error / Timer





Alphorm.com

Modules et gestion de dépendances

Modules : Console /
Errors / Timers

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Module : Console
- Module : Errors
- Module : Timers
- Site project : Blackhole



Formation NodeJS, les fondamentaux

alphorm.com™©

Module: Console

- **Console** est un module qui permet d'interagir avec **stdout** ou **stderr**.
- Le module est en **global**.
- API <https://nodejs.org/api/console.html>

Module: Errors

- Il existe deux types d'erreurs en NodeJS :
 - Les erreurs **JavaScript**
 - Les erreurs **NodeJS**
- API <https://nodejs.org/api/errors.html>

Module: Timers

- **Timers** est un module qui permet d'interagir avec l'**Event loop**.
- Il nous permet d'ajouter des fonctions à un temps donné.
- 3 types fonctions :
 - `setImmediate`
 - `setTimeout`
 - `setInterval`
- API <https://nodejs.org/api/timers.html>

Module : Timers

Exercices

- Créer une boucle avec `setInterval` et nettoyer la boucle au bout de 10 interactions.
- Correction : 'FR_235_05_06/Exercices/exercice_timers_corrige.js'

Side project : Blackhole

- Reprendre le projet 'Blackhole'
 - Ajouter aux options à notre fichier de configuration
 - log_file: "<nom du fichier>"
 - error_file: "<nom du fichier>"
 - Gérer les logs à partir du module Console
 - Si les options sont présents dans la config, écrire les logs dans les fichiers
 - Sinon, afficher les logs sur la console

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
 - Comment écrire des logs
 - Qu'elles sont les erreurs que l'on peut rencontrer
 - Comment gérer les timers
- Prochaine vidéo
 - Module : Events / EventEmitter2 / EventEmitter 3





Alphorm.com

Modules et gestion de dépendances

Modules : Events /
EventEmitter2 / EventEmitter3

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Module: Events
- Module externe : EventEmitter2
- Module externe : EventEmitter3
- Side project : Blackhole



Formation NodeJS, les fondamentaux

alphorm.com™©

Module: Events

- Une méthode de développement de NodeJS est par l'émission d'évènements.
- On inscrit une **fonction** à un **événement**.
- A chaque fois qu'on appelle cet **événement**, la fonction est **appelée**.
- API <https://nodejs.org/api/events.html>

Module externe: EventEmitter2

- EventEmitter2 est un module externe, développer par la communauté.
- Le code a été amélioré et de nouvelles fonctionnalités ont été ajoutées
 - Wildcard
 - GetAll
- API <https://github.com/asyncly/EventEmitter2>

Module externe: EventEmitter3

- EventEmitter3 est un module externe, développé par la communauté.
- Le code a été encore amélioré et extrêmement allégé.
- API <https://github.com/primus/eventemitter3>

Ce qu'on a découvert

- Nous avons vu une méthode de développement très utilisée sur NodeJS.
- Prochaine vidéo:
 - Les modules réseaux





Alphorm.com

Modules et gestion de dépendances

Modules : Url / http /
Https / Net / UDP

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

Plan

- Module : URL
- Module : Http
- Module : Https
- Module : Net
- Module : UDP
- Site project : Blackhole



Formation NodeJS, les fondamentaux

alphorm.com™©

Module: URL

- URL est un module qui permet de **parser** et de **formater** une URL.
- API <https://nodejs.org/api/url.html>

Module: HTTP

- Le module HTTP permet de :
 - Créer un client pour interagir avec un serveur HTTP.
 - Créer soit même un serveur HTTP.
 - Manipuler les headers et user-agents
- Ce module est une surcouche au module **NET**.
- API <https://nodejs.org/api/http.html>

Module: HTTP

Exercices

- Créer un objet qui permet de récupérer un contenu à partir d'une URL en HTTP ou HTTPS.
- Correction : 'FR_235_05_08/Exercices/exercice_request_corrige.js'

Module: HTTPS

- Le module HTTPS gère la couche TLS/SSL du protocole HTTP.
- L'API est exactement la même que HTTP.
- API <https://nodejs.org/api/https.html>

Module : NET

- Le module **NET** permet d'interagir avec le protocole TCP/IP en asynchrone.
- Le module permet de :
 - Créer un client.
 - Créer un serveur.
- API <https://nodejs.org/api/net.html>

Module : UDP

- Le module **UDP** permet d'interagir avec le protocole UDP/IP et multicast en asynchrone.
- Le module permet de :
 - Créer un client.
 - Créer un serveur.
- API <https://nodejs.org/api/dgram.html>

Side project : Blackhole

- Reprendre le projet 'Blackhole'
 - Intégrer l'abstraction du protocole HTTP / HTTPS

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu les différents modules réseaux que NodeJS nous propose.
- Prochaine vidéo
 - Module : Child Process / Cluster





Alphorm.com

Modules et gestion de dépendances

Modules : ChildProcess / Cluster

Site : <http://www.alphorm.com>
Blog : <http://blog.alphorm.com>



Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Formation NodeJS, les fondamentaux

alphorm.com™©

JS Plan

- Module : ChildProcesses
- Module : Cluster



Formation NodeJS, les fondamentaux

alphorm.com™©

Module: ChildProcess

- Le module **ChildProcess** permet de gérer :
 - Le lancement de nouveaux processus.
 - La communication interprocessus :
 - Au 'pipe' (Stdin, Stdout, Stderr).
 - Par message.
 - Les signaux.
- L'objet **ChildProcess** hérite d'un EventEmitter.

Module: ChildProcess

- 3 fonctions principales :
 - **Spawn** : Détache le processus.
 - **Exec** : Ne détache pas le processus, attend un retour. Buffer de 200k max.
 - **Fork** : Détache et embarque v8 dans le processus.
- API https://nodejs.org/api/child_process.html

Module: Cluster

- Le module Cluster est un nouveau module qui va spawn automatiquement les processus fils et va faire du load balancing sur les ports TCP.
- Il offre une surcouche de gestion de processus fils :
 - Spawn
 - Kill
 - Algorithme de load balancing
- API <https://nodejs.org/api/cluster.html>

Ce qu'on a couvert

- Nous avons vu comment exploiter tous les **core** de notre machine physique.
- Je vous recommande les modules communautaires 'Pm2' ou 'forever'



- Prochaines vidéos, **Blackhole** !!

