Formation Avancée de **NodeJS**

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

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

Plan

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

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 de la formation**
  - Présentation de la formation
  - Scénario de la formation
- **Node et le Web : HTTP, Request & Express**
  - Périmètre du module HTTP
  - Request
  - Serveur Web avec Express
  - Express : Router les requêtes
  - Express : Le gestionnaire de route
  - Express : les fichiers statiques
  - Express : Utiliser les middlewares
  - Express : Les moteurs de template
  - Side Project : ChatWithMe
- **L'asynchrone en détail**
  - Callback Hell & Pyramid of Doom
  - Async : Cours
  - Async : Correction d'exercice
  - Les promesses avec Q : Les fondamentaux
  - Les promesses avec Q : Gestion des erreurs et les accès directs
  - Les promesses avec Q : Flow, Timers et I/O
  - Les promesses : Exercices
- **Communication temps réel**
  - Socket.io
  - Intégration à Express
  - Projet : ChatWithMe Partie 1
  - Projet : ChatWithMe Partie 2

Plan de la formation

- Liaison avec la persistance des données
  - Intéraction avec une base de données relationnelle (Mysql)
  - Sequelize, l'introduction
  - Sequelize, les associations
  - Intégration avec une base de données NoSQL : Redis
  - Intégration avec une base de données NoSQL : Mongo, l'introduction
  - Intégration avec une base de données NoSQL : Mongo, les associations
  - Intégration de MySQL à ChatWithMe
  - Intégration de Sequelize à ChatWithMe
  - Intégration de MongoDB à ChatWithMe
- Bonus
  - PM2, le monitoring
  - Keymetric, la supervision
- Conclusion

Objectifs de la formation

- Apprendre et comprendre les différentes méthodes asynchrones de NodeJS.
- Apprendre à concevoir un projet important.
- Comprendre comment fonctionne un des frameworks les plus utilisés de NodeJS.
- Mettre en place un système MVC et une abstraction de base de données.
- Savoir développer une application web d'un niveau professionnel.

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
  - Connaissance en asynchrone
  - Connaissance en event-driven architecture
  - Bien connaitre les modules natifs de NodeJS

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

Scénario de la formation

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

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

Plan

- Formation
- Side project : ChatWithMe

Formation

- Cette formation est une seconde partie de la formation complète de NodeJs, les fondamentaux.
- Dans cette seconde partie, nous verrons ensemble :
  - Comment utiliser un des frameworks les plus utilisés de NodeJS : **Express**
  - **Deux méthodes** de code asynchrone.
  - Comment communiquer du code JavaScript côté client (navigateur web)
  - Comment sauvegarder nos données dans des bases de données ou dans des fichiers
- En plus, dans cette formation, nous développerons une plateforme web sur le modèle **MVC** et avec une abstraction de base de donnée.

Side Project : ChatWithMe

- À partir d'un squelette, nous allons développer une base solide d'un chat.
- Nous développerons par étape
  1. Le serveur web.
  2. La communication en temps réel avec le navigateur.
  3. La sauvegarde des données.
  4. Le monitoring de l'application
- Libre à vous d'améliorer ce projet après cette formation.

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**

- Prochain chapitre:
  - Node et le Web : HTTP, Request & Express

Node et le Web

Périmètre du module HTTP

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

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

Plan

- Rappel
- Une requête HTTP (partie client)
- Un serveur HTTP (partie serveur)

Rappel

- 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

Une requête HTTP (partie client)

- Créons notre premier script pour envoyer une requête avec '**http**'
- Dans '**FR_260_02_00/Exercices/**'
  - Faire un script qui se connectera à l'URL https://httpbin.org/get en **GET**
  - Faire un script qui se connectera à l'URL https://httpbin.org/post en **POST** avec en 'formData' :
    - **User** : 'bob'
    - **Message** : "Bonjour !"
- Correction :
- '**FR_260_02_00/Exercices/exercice_request_get_corrigé.js**'
- '**FR_260_02_00/Exercices/exercice_request_post_corrigé.js**'

Un serveur HTTP (partie serveur)

- Dans '**FR_260_02_00/Exercices/**'
- Développer un serveur HTTP et gérer différemment les requêtes GET et POST
- Bonus : Gérer deux routes différentes :
  - http://localhost:12000/users
  - http://localhost:12000/media
- Correction : '**FR_260_02_00/Exercices/exercice_server_http_corrigé.js**'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Un rapide rappel de NodeJS
  - Comment lancer une requête HTTP avec le module natif.
  - La difficulté de créer un serveur HTTP avec le module natif.
- Dans la prochaine vidéo, nous verrons le module Request, qui est une abstraction des modules natifs HTTP/HTTPS, et qui prend en charge énormément de fonctionnalités.

Alphorm.com

Node et le Web

Request

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

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

Plan

- Le module Request
- Les requêtes HTTP(s)

Le module Request

- Le module Request est une abstraction de différents modules natifs de NodeJs.
- Il est designé pour être le plus simple possible d'utilisation.
- Supporte :
  - **HTTP / HTTPS**
  - Suit les **redirections** 2xx
  - Gère les requêtes en **event** ou **stream/pipe**
  - **...**
- API https://github.com/request/request

Les requêtes HTTP(s)

- Créons notre premier script pour envoyer une requête avec '**request**'
- Dans '**FR_260_02_01/Exercices/**'
  - Faire un script qui se connectera à l'URL https://httpbin.org/get en **GET**
  - Faire un script qui se connectera à l'URL https://httpbin.org/post en **POST** avec en 'formData' :
    - **User** : 'bob'
    - **Message** : "Bonjour !"
- Correction :
- '**FR_260_02_01/Exercices/exercice_request_get_corrigé.js**'
- '**FR_260_02_01/Exercices/exercice_request_post_corrigé.js**'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment lancer des requêtes HTTP(s) avec request
- Prochaine vidéo :
  - Serveur Web avec Express

Alphorm.com

Node et le Web

Serveur Web avec Express

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

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

Plan

- Qu'est-ce qu'Express ?
- Installation
- Hello word
- le 'Generator'

Qu'est-ce qu'Express ?

- Express est une infrastructure web minimaliste, souple et rapide pour Node.JS
- Express fournit un ensemble de fonctionnalités robustes pour les applications web et mobiles.
- Grâce à une foule de méthodes utilitaires HTTP et de middleware mise à votre disposition, la création d'une API robuste est simple et rapide.
- Express apporte une couche fine de fonctionnalités d'application Web fondamentales, sans masquer les fonctionnalités de Node.js que vous connaissez et appréciez.

Installation

- Allez dans '**FR_260_02_02/Cours/hello_word**'
- Créer un projet NodeJS : **npm init**
- Puis installer express : **npm install --save express**

Hello word

Let's code

Cours : 'FR_260_02_02/Cours/hello_word/index.js'

Le 'Generator'

- Le generator est un outil pour créer un squelette rapidement.
- Dans '**FR_260_02_02/Cours/**'
- Installer le generator : $> npm install express-generator –g
- Générer un projet : $> express myapp
- Aller dans le projet : $> cd myapp
- Installer les modules nécessaires : $> npm install
- Lancer le projet : $> DEBUG=myapp:* npm start

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment créer un projet Express
  - Comment créer un serveur HTTP très light
  - Comment générer un squelette pour express
- Prochaine vidéo :
  - Express : Router les requêtes

Node et le Web

Express : *Router les requêtes*

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

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

Plan

- Qu'est-ce qu'une route ?
- Comment définir une route avec Express ?

Qu'est-ce qu'une route ?

- Une route est un 'path' contenu dans une URL :
  - http://example.com/users
  - http://example.com/contats
  - http://example.com/user/eferrari
  - http://example.com/user/eferrari/contacts

Comment définir une route avec Express ?

- On peut définir une route telle quelle :

```
app.METHOD(PATH, HANDLER)
```

  - **App** est l'instance d'Express
  - **METHOD** est une méthode de requête HTTP
  - **PATH** est la route définie
  - **HANDLER** est la fonction qui sera appelée.

- Cours : 'FR_260_02_03/Cours/routes/routes_basiques.js'

Comment définir une route avec Express ?

- Il est possible de mettre des **regex** dans le path.

- Cours : 'FR_260_02_03/Cours/routes/routes_regex.js'

Comment définir une route avec Express ?

- Il est possible de **chainer** les routes.

- Cours : 'FR_260_02_03/Cours/routes/routes_chainées.js'

Comment définir une route avec Express ?

- Il est possible de créer des modules pour mieux architecturer notre projet avec la class **express.Router**.
- Cette classe est une instance complétée avec les middlewares et son système de routing. Dans la doc, on peut le retrouver comme appellé une "mini-app".
- Cours : 'FR_260_02_03/Cours/routes/routes_miniapp.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Ce qu'était une route.
  - Comment définir une route avec Express de plusieurs façons.
- Prochaine vidéo :
  - Express : Le gestionnaire de route

Node et le Web

Express :
Le gestionnaire des routes

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

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

Plan

- Le gestionnaire des routes
- Exemple

Le gestionnaire des routes

- Avec le gestionnaire, on peut fournir plusieurs callback.
- À la fin de chaque fonction, on appelle '**next()**'
- On peut utiliser ce mécanisme pour imposer une précondition sur une route (Authentification, Autorisation …)
- Le gestionnaire peut être une fonction, un tableau de fonction, ou les deux.

Exemple

Let's code

Cours : 'FR_260_02_04/Cours/gestionnaire_de_route.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comme nous pouvions gérer les conditions à l'appel d'une route
- Prochaine vidéo :
  - Express : Les fichiers statiques

Alphorm.com

Node et le Web

Express :
Les fichiers statiques

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

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

Plan

- Comment les fichiers statiques sont gérés ?
- Exemple

Comment les fichiers statiques sont gérés ?

- Pour servir les fichiers statiques tels que les fichiers CSS ou JS, il faut faire appel à '**express.static**'

```
app.use(express.static('public'));
```

- Il est possible de définir plusieurs répertoires statiques.

```
app.use(express.static('public'));
app.use(express.static('files'));
```

- De plus, nous pouvons définir un répertoire statique dans un chemin virtuel.

```
app.use('/static', express.static('public'));
```

Exemple

Let's code

Cours : **'FR_260_02_05/Cours/*'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment gérer les fichiers statiques
- Prochaine vidéo :
  - Express : Ecrire un middleware

Node et le Web

Express : *Ecrire un middleware*

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

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

Plan

- Qu'est-ce qu'un middleware ?
- L'ordre des fonctions
- Les différentes possibilités des middlewares

Qu'est-ce qu'un middleware ?

- Express est une infrastructure web de routage.
- Il y a peu de fonctionnalités présentes par défaut.
- Une application développée avec Express, n'est ni plus ni moins qu'une succession d'appels de fonctions middleware.
- Un **middleware** est une fonction qui a 3 arguments :
  - **req** : Classe "Request", contient les informations de la requête
  - **res** : Classe "Response", permet de forger la réponse
  - **next** : fonction middleware, qui appelle le **middleware** suivant

Qu'est-ce qu'un middleware ?

- Les fonctions middleware effectuent les tâches suivantes :
  - Exécuter tout type de code.
  - Apporter des modifications aux objets de demande et de réponse.
  - Terminer le cycle de demande-réponse.
  - Appeler le middleware suivant dans la pile.
- Attention : Une fonction middleware doit appeler la fonction next() pour transmettre le **contrôle** à la fonction middleware **suivante**. Sinon, la demande restera **bloquée**.
- Cours : 'FR_260_02_06/Cours/middleware_logger.js'

L'ordre des fonctions

- L'ordre des fonctions est très important.
- Une fois qu'une fonction middleware finit le cycle, toutes les fonctions qui se trouvent après ne seront pas appelées.
- Cours : 'FR_260_02_06/Cours/middleware_logger.js'

Les différentes possibilités des middlewares

- Il est possible de transmettre stocker des informations dans **request** ou **response**.
- Par exemple pour :
  - Récupérer plus d'information sur le serveur (header)
  - Ajouter des éléments nécessaires à la réponse
  - Faire un middleware d'authentification (Oauth, ...)
  - Utiliser un moteur de modèle
  - Parse un body d'une requête POST
  - ...
- Cours : 'FR_260_02_06/Cours/middleware_time.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Qu'est ce qu'était un middleware
  - Nous avons pu voir comprendre comment Express fonctionne.
  - Les différentes possibilités offertes par les middlewares
- Prochaine vidéo :
  - Express : Les moteurs de template

Alphorm.com

Node et le Web

Express :
Les moteurs de template

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

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

Plan

- Intégrer un moteur de template
- **Consolidate.js** : Intégration dans Express

Intégrer un moteur de template

- Un moteur de template permet de générer du code HTML à partir d'un code plus simple.
- On peut y intégrer des variables et des objets, ce qui permet de rendre le template "dynamique".
- Le template est le squelette de la page.
- Suivant les données que nous recevons du Model (db, api, etc), le moteur de template va compléter le template et générer de l'HTML.
- Avec Express, on utilise beaucoup Jade.
- Cours : 'FR_260_02_06/Cours/express_jade.js'

Consolidate.js : Intégration dans Express

- Consolidate.js rassemble énormément de moteurs de template :
  - atpl
  - doT.js (website)
  - dust (unmaintained) (website)
  - …
- Cours : 'FR_260_02_06/Cours/express_consolidate.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Qu'elle est l'utilité d'un moteur de template.
  - Comment l'utiliser et comment envoyer des variables au template.
- Prochaine vidéo :
  - Side project : ChatWithMe

Node et le Web

Side project : *ChatWithMe*

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

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

Plan

- Créer le projet avec Express
- Créer un projet sur github
- Lier le projet sur github
- Initialiser NPM
- Copier le code

Hello word

Let's code

Ce qu'on a couvert

- Nous avons une bonne base pour commencer le développement de notre application !
- Prochain chapitre :
  - L'asynchrone en détail

L'asynchrone en détail

Callback
Hell & Pyramid of Doom

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

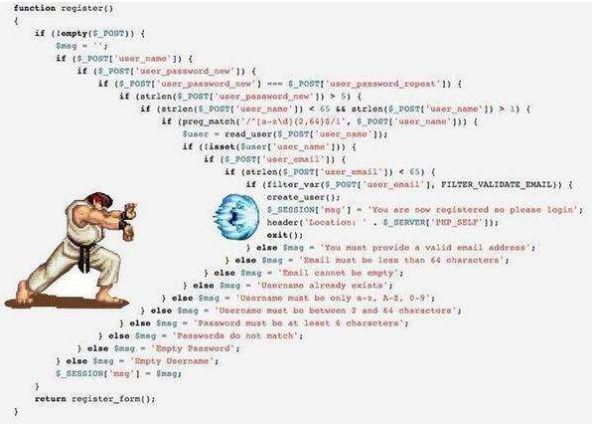
Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Plan

- Pyramid of doom
- Callback hell
- Comment éviter les pièges ?

Pyramid of doom

- La pyramide de la malediction
- C'est le fait d'enchainer les blocks en cascade



Callback hell

- En JavaScript, quand une tâche est faite, la callback définie est appelée
- On peut rapidement arriver à un code comme ci-dessous:

```
fs.readdir(source, function(err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function(filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function(err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function(width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(destination + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Comment éviter les pièges ?

- Nommer les fonctions anonymes
  - Le code est plus simple à lire
  - Quand une exception arrive, les fonctions dans la stacktrace seront nommées
- Garder le code peu profond
- Modulariser !
  - Node permet de créer des modules très facilement, profitez s'en !
  - Issac Schlueter "*Ecrire de petits modules qui font qu'une seule chose, puis les assembler dans d'autres modules qui feront de plus grande chose. Vous ne pouvez pas entrer dans l'enfer rappel si vous n'y allez pas.*"

Exemple

Let's code

Exercices: **'FR_260_03_00/Exercice/code_sale.js'**
Corrigé: **'FR_260_03_00/Exercice/code_sale_corrigé.js'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Quelles sont les plus gros pièges de JavaScript et de NodeJS
- Prochaine vidéo :
  - Async : Cours

Alphorm.com

L'asynchrone en détail

Async : *Cours*

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

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

Plan

- Qu'est-ce que Async ?
- Exemples

Qu'est-ce que async ?

- API : https://github.com/caolan/async
- Async est un module communautaire qui fournit de simple et de puissantes fonctions pour travailler avec les fonctions JS asynchrones.
- Async gère 3 categories :
  - Les collections
  - Le control Flow
  - Utils

Exemples

```
async.map(['file1','file2','file3'], fs.stat, function(err, results){
    // results is now an array of stats for each file
});

async.filter(['file1','file2','file3'], fs.exists, function(results){
    // results now equals an array of the existing files
});

async.parallel([
    function(){ ... },
    function(){ ... }
], callback);

async.series([
    function(){ ... },
    function(){ ... }
]);
```

Cours: 'FR_260_03_01/Cours/'

Exemple

Let's code

Cours: 'FR_260_03_00/Cours/code_sale.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Les principaux usages de async
    - Filter
    - Map
    - Parallel
    - Serie
    - waterfall
- Prochaine vidéo :
  - Async : Exercices

Alphorm.com

L'asynchrone en détail

Async : *Exercices*
Partie 1

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

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

Plan

- Async-you
  - Waterfall
  - Series objects
  - Each
  - Map

Async-you

- Async-you est un programme intéractif qui corrigera automatiquement vos exercices.
- Tous les exercices sont traduits en francais.
- Je commenterai moi-même le code corrigé
- Pour **installer** :
  - **$> npm install –g async-you**
- Pour **lancer** :
  - **$> async-you**
- Changer la langue en **Francais**, si vous le désirez

Async-you

Let's code

Exercices: **'FR_260_03_02/Exercices/'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Les principaux usages de async
    - Waterfall
    - Serie
    - Each
    - Map
- Prochaine vidéo :
  - Async : Exercices Partie 2

L'asynchrone en détail

Async : *Exercices* Partie 2

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

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

Plan

- Async-you
  - Times
  - Reduce
  - Whilst

Async-you

Let's code

Exercices: **'FR_260_03_02/Exercices/'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Les principaux usages de async
    - Times
    - Reduce
    - whilst
- Prochaine vidéo :
  - Les promesses avec Q: Les fondamentaux

L'asynchrone en détail

Les promesses avec Q : Les fondamentaux

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

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

Plan

- Qu'est-ce que sont les promesses?
- Module Q
- Installation de Q
- Faire une promesse basée sur une tâche
- Chainer les actions

Qu'est-ce que "Promise" ?

- **Promise** en anglais
- Les promises fournissent une alternative aux callbacks lorsqu'on traite un code asynchrone.
- Une promesse est une abstraction pour la programmation asynchrone.
- C'est un objet qui proxifie une valeur de retour ou une exception renvoyée par une fonction qui doit faire un traitement asynchrone.
- Malheureusement, la compréhension des promesses n'est pas facile.

Module Q

- Il existe plusieurs interprétations des promesses.
- Les promesses sont intégrées dans ECMAscript6
- API : https://github.com/kriskowal/q
- Cours: 'FR_260_03_03/Cours/introduction.js'

Installation de Q

- Avec nodeJS
  - $> npm install q
- Coté client, avec un navigateur web
  - <script src="q.min.js"></script>

Faire une promesse basée sur une tache

Let's code

Cours: **'FR_260_03_03/Cours/premiere_promesse.js'**

Chainer les actions

Let's code

Cours: **'FR_260_03_03/Cours/chainer_les_actions.js'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Les principaux usages des promesses avec Q
  - Comment créer une promesse
  - Comment chainer plusieurs actions
- Prochaine vidéo :
  - Les promesses avec Q : les conditions et les boucles

Alphorm.com

L'asynchrone en détail

Les promesses avec Q : Les conditions et les boucles

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

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

Plan

- Utilisation de condition dans une promesse
- Utilisation de boucle avec une promesse

Utilisation de condition dans une promesse

Let's code

Cours: **'FR_260_03_05/Cours/condition.js'**

Utilisation d'une boucle avec une promesse

Let's code

Cours: **'FR_260_03_05/Cours/boucle.js'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment gérer les conditions et les boucles
- Prochaine vidéo :
  - Les promesses avec Q : la gestions des erreurs

Alphorm.com

L'asynchrone en détail

Les promesses avec Q :
La gestion des erreurs et les accès directs

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

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

Plan

- La gestion des erreurs avec Q
- Gestion des objets avec les méthodes

La gestion des erreurs avec Q

- Q inclut une gestion “standard” des erreurs
- La fonction *then* peut prendre deux arguments, le second argument est une fonction anonyme qui sera appelée si une erreur intervient
- De plus avec *deferred*, on peut appeler *.reject(arg en erreur)*.
- En plus des *.then()*, on peut ajouter un *.fail()* qui récupérera les erreurs non traitées.
- Cours: ‘FR_260_03_06/Cours/erreurs.js’

Gestion des objets avec les méthodes

- Il est possible d'utiliser des méthodes de Q qui vont vous permettre de manipuler les données lors du flow des promesses.
- C'est un gain de temps
- Permettra de rendre votre code encore plus clair et concis.
- Cours: 'FR_260_03_06/Cours/manipulation_object.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment gérer les erreurs
  - Comment accéder en direct à certaines fonctions de Q
- Prochaine vidéo :
  - Les promesses avec Q : Flow, Timers et I/O

L'asynchrone en détail

Les promesses avec Q : *Flow, Timers et I/O*

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

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

Plan

- Composer un flow
- Les timers
- Les modules I/O

Composer un flow

- Les promesses de base supportent seulement un seul flow à la fois. Q permet de gérer plusieurs flow.
- Imaginons, nous avons deux flow :
  - Un qui fait une requête
  - Un qui log la tâche
- Grâce à Q, il est possible de regrouper plusieurs promesses distinctes sous une "grosse" promesse.
- Si un des deux flow a une erreur, l'autre tâche est quand même exécutée
- Si une des sous-promesses est en erreur, toutes les promesses sont arrêtées et mises en "fail".
- Cours: 'FR_260_03_07/Cours/composer_un_flow.js'

Les timers

- Les promesses ont deux systèmes de contrôle de temps :
  - .delay(ms)
  - .timeout(ms)
- Cours: 'FR_260_03_07/Cours/timers.js'

Les modules I/O

- Q nous propose des fonctions qui permettent de faire appel à des fonctions des modules I/O :
  - Q.nfapply(func, args)
  - Q.nfcall(func, arg1, arg2, ...)
  - Q.nfbind(func, arg1, arg2,...)
  - Q.npost(obj, name, args)
  - Q.nsend(obj, name, arg1, arg2, ...)
- Ne fonctionne pas avec le module HTTP
- Cours: 'FR_260_03_07/Cours/module_io.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment composer un flow complex
  - Comment utiliser les timers
  - Comment utiliser les modules I/O
- Prochaine vidéo :
  - Les promesses : Exercices Partie 1

Alphorm.com

L'asynchrone en détail

Les promesses : *Exercices Partie 1*

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

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

Plan

- promise-it-wont-hurt
  - Accomplir une promesse
  - Rejeter une promesse
  - Rejet ou ne pas rejeter
  - Toujours asynchrones
  - Valeurs et promesses
  - Lever une erreur

promise-it-wont-hurt

- Promise-it-wont-hurt est un programme intéractif qui corrigera automatiquement vos exercices.
- Tous les exercices sont traduits en français.
- Je commenterai moi-même le code corrigé
- Pour **installer** :
  - $> npm install –g promise-it-wont-hurt
- Pour **lancer** :
  - $> promise-it-wont-hurt
- Changer la langue en **Français**, si vous le désirez

promise-it-wont-hurt

Let's code

Exercices: **'FR_260_03_08/Exercices/'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons revu les principaux usages des promises.
- Prochain chapitre :
  - Les promesses : Exercices Partie 2

Alphorm.com

L'asynchrone en détail

Les promesses : *Exercices Partie 2*

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

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

Plan

- promise-it-wont-hurt
  - Utiliser qfcall
  - Une règle importante
  - Promesses multiples
  - Récurérer un JSON
  - Faire plus de boulot

promise-it-wont-hurt

Let's code

Exercices: **'FR_260_03_09/Exercices/'**

Ce qu'on a couvert

- Dans cette vidéo, nous avons revu les principaux usages des promises.
- Prochain chapitre :
  - Communication temps réel

Alphorm.com

Communication temps réel

Socket.io

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

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

Plan

- Un peu d'histoire
- WebSocket

Un peu d'histoire

- Avant, pour mettre à jour une information sur une page web :

```
setInterval(function(){
    $.ajax({ url: '/my/page', success: function(data){
        // do something with the data
    } });
}, 5000);
```

- Tous les 5 minutes, nous mettons à jour nos données.
- C'est aussi efficace que le protocole "Transmission de datagramme IP par pigeons"
- http://www.rfc-editor.org/rfc/rfc1149.txt

Un peu d'histoire

- Vu qu'on veut mettre à jour souvent, on baisse le rafraichissement

```
setInterval(function(){
    // now this should be fast!
    $.ajax({ url: '/my/page', success: function(data){} });
}, 100);
```

- En créant autant de requêtes, de nouveaux problèmes apparaissent :
  - Si la requête HTTP met plus de 100 millisecondes, deux requêtes sont lancées parallèlement.
  - Quand le serveur n'a pas de nouvelle donnée, ces requêtes sont toutes simplement de la consommation de ressource inutile.

Un peu d'histoire

- Puis, nous utilisions le "long pooling"

```
function load(){
    $.ajax({ url: '/my/page', success: function(){
        // do something with the data
    }, complete: load, timeout: 20000 });
}
```

- Cette technique garde la connexion ouverte jusqu'à avoir une réponse serveur.
- Cela a permis de réduire considérablement la latence et la bande passante.
- C'est la méthode qui se rapproche le plus des WebSockets.

WebScocket

- Les **WebSocket** permettent de fournir un mécanisme pour les navigateurs qui ont besoin de communiquer
  - Avec un navigateur
  - Des deux sens (upload, download)
  - et reste ouvert après avoir reçu les données
- Il est ajouté à partir du protocole HTTP/1.1
- Malgré ça, il reste deux problèmes majeurs :
  - Le serveur de destination et tous les serveurs en relais (*Cache, Reverse proxy, load balancer, ...*) doivent savoir comment gérer ce protocole.
  - Beaucoup d'ancien navigateur ne supporte pas ce protocole.

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - A partir de quelle problèmatique les websocket et socket.io ont été créés
  - Une rapide overview du protocole websocket.
- Prochaine vidéo :
  - Intégration dans Express

Alphorm.com

Communication temps réel

Intégration à Express

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

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

Plan

- Ajouter le code côté serveur
- Ajouter le code côté client
- Side project : ChatWithMe

Cours

Ajouter le code côté serveur

Cours : **'FR_260_04_01/Cours/'**

Cours

Ajouter le code côté client

Cours : **'FR_260_04_01/Cours/'**

Side project: ChatWithMe

- Intégrer socket.io dans ChatWithMe
- Side project : 'ChatWithMe/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu comment préparer un projet pour socket.io
- Prochaine vidéo :
  - Projet : ChatWithMe, le code backend

Alphorm.com

Communication temps réel

Side Project : ChatWithMe, le code Backend

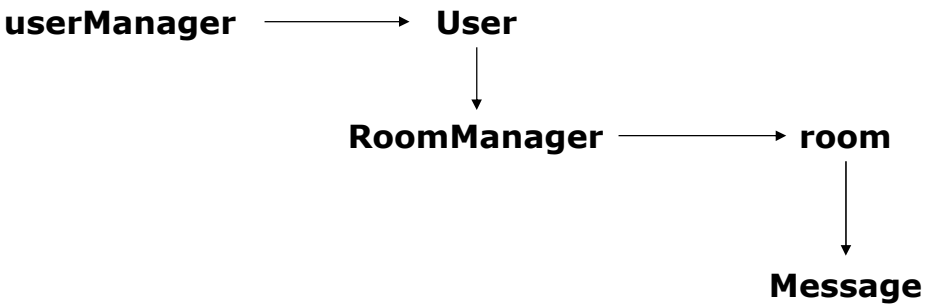
Site : http://www.alphorm.com
Blog : http://blog.alphorm.com

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

Plan

- Arborescence des objets et des Managers
- Liste des requêtes vers le backend
- Let's code !

Arborescence des objets et des Managers



Liste des requêtes backend

- Liste des requêtes backend
- JoinRoom : Le *user* joins une *room.*
- QuitRoom : Le *user* quitte la *room.*
- SetName : Le *user* change de nom
- NewMessage : Le *user* écrit un nouveau ***message*** dans une ***room***

Liste des requêtes backend

- Quand l'utilisateur se connecte au **serveur** :

1. Le serveur transmet la liste des **rooms** à l'utilisateur.

- Quand l'utilisateur rejoint une **room**

1. La **room** avertit tous ses **utilisateurs** qu'un nouvel **utilisateur** est arrivé.
2. La **room** transmet tous ses **informations** au nouvel **utilisateur**.

- Quand un **utilisateur** écrit un **message** dans une **room** :

1. L'information est reçue dans l'objet **user.**
2. Transmets l'information à l'objet **room.**
3. L'objet **room**, transmet l'information à tous les utilisateurs.

ChatWithMe, le code Backend

Let's code

Side project: 'ChatWithMe/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons développé le code backend pour les requêtes suivantes :
  - JoinRoom
  - QuitRoom
  - SetName
  - NewMessage

- Prochaine vidéo :
  - Side Project : ChatWithMe, le code frontend

Alphorm.com

Communication temps réel

Side Project : ChatWithMe, le code Frontend

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

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

Plan

- Arborescence des fichiers
- Liste des requêtes vers le frontend
- Let's code !

Arborescence des objets et des Managers

Realtime.js ⟷ **Room**

Liste des requêtes backend

- Liste des requêtes frontend :
  - **roomList** : Une fois connecté, le serveur envoie la liste des rooms disponibles.
  - **joinRoom** : Une fois que nous avons rejoint une *room*, on reçoit toutes les infos nécessaires à créer une nouvelle *room*.
  - **roomNewUser** : Un nouveau user rentre dans une *room.*
  - **roomUsers** : Si quelqu'un rentre ou sort de la room, on est averti par cette requête.
  - **roomMessages** : Pas implémenté.
  - **roomNewMessage** : Si un *user* écrit un message, on est averti par cette requête.

ChatWithMe, le code Front

Let's code

Side project: 'ChatWithMe/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons développé le code frontend pour les requêtes suivantes :
  - joinRoom
  - roomList
  - roomUsers
  - roomMessages
  - roomNewMessage
  - roomNewUser
- Prochain chapitre :
  - Liaison avec la persistance des données

Alphorm.com

Liaison avec la persistance des données

Intégration avec une base de données relationnelle (Mysql)

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

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

Plan

- Introduction à Mysql
- Mysql avec Cloud9
- Let's code

Introduction à Mysql

- MySQL est un système de gestion de bases de données relationnelles (SGBDR).
- Formation complète sur Alphorm :
  - http://www.alphorm.com/tutoriel/formation-en-ligne-mysql-1z0-883
- Savoir comment utiliser Node.JS et Mysql est plus que nécessaire.
- Mysql est une base de données la plus utilisée dans le monde et très performante.
- Tous les langages ont développé un driver pour interagir avec Mysql.

Mysql avec Cloud9

- $> mysql-ctl start
- $> mysql-ctl stop
- $> mysql-ctl cli

- Informations :
  - Hostname – process.env.IP
  - Port - 3306 (Port par défaut de Mysql)
  - User – process.env.C9_USER
  - Password - `` (Pas de mot de passe)
  - Database - c9 (Le nom de la base de données)

Exemple

Let's code

Cours : 'FR_260_05_00/Cours/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment utiliser mysql avec NodeJs
- Prochaine vidéo :
  - Sequelize, l'introduction

Alphorm.com

Liaison avec la persistance des données

Sequelize, l'introduction

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

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

Plan

- Qu'est ce qu'un ORM ?
- Sequelize
- Let's code

Qu'est-ce qu'un ORM

- Un **mapping objet-relationnel** (en anglais **object-relational mapping** ou **ORM**)
- C'est une technique de programmation informatique qui crée l'illusion d'une base de données orientée objet à partir d'une base de données relationnelle.
- L'ORM définit des correspondances entre cette base de données et les objets du langage utilisé.
- On pourrait le désigner par :
  - « correspondance entre monde objet et monde relationnel »

Sequelize

- **Sequelize** est un ORM basé sur les promesses pour NodeJS
- http://docs.sequelizejs.com/en/latest/
- Il supporte les bases de données suivantes :
  - PostgreSQL
  - MySQL
  - MariaDB
  - SQLite
  - MSSQL
- Et des fonctionnalités de transaction, relations, read replication, …

Sequelize

- La première chose à faire est de définir les modèles

```
var Project = sequelize.define('project', {
   title: Sequelize.STRING
   , description: Sequelize.TEXT
})
```

- Sequelize, nous permet de définir les champs que nous utiliserons.
  - http://docs.sequelizejs.com/en/latest/docs/models-definition/

Sequelize

- De plus, Sequelize fonctionne avec les promises.
- Cela ne fonctionne pas !

```
user = User.findOne();
console.log(user.name);
```

- Ce qu'il faut faire :

```
User.findOne().then(function (user) {
  console.log(user.name);
});
```


Let's code

- Maintenant, nous allons voir les grandes utilisations de Sequelize:
  - La configuration
  - La création de tables
  - L'insertion de données
  - Comment retrouver ses données
- Cours : 'FR_260_05_01/Cours/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment utiliser sequelize
  - Comment éviter les pièges
  - Et partir sur de bonne base
- Prochaine vidéo :
  - Sequelize, les associations

Alphorm.com

Liaison avec la persistance des données

Sequelize, les associations

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

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

Plan

- Les différents types de liaison
- Let's code

Les différents types de liaison

- Avec les ORMs, il existe plusieurs types de liaison.
- Les One-to-One
  - **BelongsTo**: "Appartient à", un **Post** appartient à un **User**.
  - **HasOne**: "à un" object
  - **HasMany**: "à plusieurs" object, un **User** peut avoir plusieurs **Post.**
  - ...
- http://docs.sequelizejs.com/en/latest/docs/associations/

Les différents types de liaison

- Les associations vont nous permettre de **lier** les données
- Par exemples, deux tables :
  - **Users** (id, firstname, lastname)
  - **Users-Posts** (id, **userId**, subject, content)

Exemple

Let's code

Cours : 'FR_260_05_02/Cours/association.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment faire des associations de table
- Prochaine vidéo :
  - Interaction avec une base de données NoSQL : Redis

Alphorm.com

Liaison avec la persistance des données

Interaction avec une base de données NoSQL : Redis

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

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

Plan

- Introduction à Redis
- Redis avec Cloud9
- Let's code

Introduction à Redis

- Redis (de l'anglais **REmote DIctionary Server** qui peut être traduit par « serveur de dictionnaire distant » et jeu de mots avec Redistribute).
- C'est un système de gestion de bases de données **clef-valeur scalable**, très hautes performances, écrit avec le langage de programmation C.
- Il fait partie du mouvement NoSQL et vise à fournir les performances les plus élevées possibles.
  - http://redis.io/commands

Redis avec Cloud9

- Pour lancer le serveur
  - $> sudo service redis-server start
- Pour se connecter au serveur
  - $> redis-cli

Exemple

Let's code

Cours : 'FR_260_05_03/Cours/'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment utiliser redis avec NodeJs
- Prochaine vidéo :
  - Interaction avec une base de données NoSQL : MongoDB, l'introduction

Alphorm.com

Liaison avec la persistance des données

Intégration avec une base de données NoSQL : MongoDB, l'introduction

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

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

Plan

- Introduction à MongoDB
- MongoDB avec Cloud9
- CRUD

Introduction à MongoDB

- MongoDB de l'anglais humongous qui peut être traduit par « énorme »
- C'est un système de gestion de bases de données :
  - Orientée documents.
  - Répartissable sur un nombre quelconque d'ordinateurs.
  - Ne nécessitant pas de schémas prédéfinis des données.
- Il est écrit en C++.
- Il fait partie du mouvement **NoSQL**.

MongoDB avec Cloud9

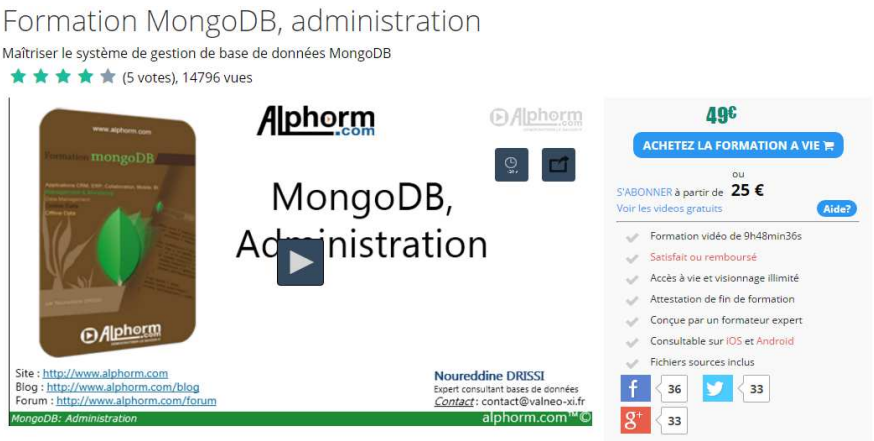
- Pour lancer le serveur
  - $> mongod --nojournal
- Pour se connecter au serveur
  - $> mongo

CRUD

- Nous allons voir les principales utilisations de **mongodb** avec **mongoose**.
  - Create : 'FR_260_05_05/Cours/create.js'
  - Read: 'FR_260_05_05/Cours/read.js'
  - Update: 'FR_260_05_05/Cours/update.js'
  - Delete: 'FR_260_05_05/Cours/delete.js

Pour aller plus loi

- Une formation complète sur Mongodb sur Alphorm : http://www.alphorm.com/tutoriel/formation-en-ligne-mongodb-administration



Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment utiliser MongoDB avec **mongoose**.
- Prochaine vidéo :
  - Interaction avec une base de données NoSQL : MongoDB, les associations

Liaison avec la persistance des données

Intégration avec une base de données NoSQL : MongoDB, les associations

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

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

Plan

- Les associations
- Let's code

Les associations

- De la même façon qu'avec MySQL, nous allons lier les **users** et les **posts.**

Exemple

Let's code

Cours : 'FR_260_05_05/Cours/associate.js'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment faire des associations de documents avec MongoDb
- Prochaine vidéo
  - Intégration de MySQL à ChatWithMe

Alphorm.com

Liaison avec la persistance des données

Intégration de MySQL à ChatWithMe

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

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

Plan

- Let's code

Exemple

Let's code

Side project: 'ChatWithMe'

Ce qu'on a couvert

- Dans cette vidéo, nous avons intégré Mysql à ChatWithMe
- Prochaine vidéo
  - Intégration de Sequelize à ChatWithMe

Alphorm.com

Liaison avec la persistance des données

Intégration de Sequelize à ChatWithMe

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

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

Plan

- Let's code

Exemple

Let's code

Side project: 'ChatWithMe'

Ce qu'on a couvert

- Dans cette vidéo, nous avons intégré Sequelize à ChatWithMe
- Prochaine vidéo
  - Intégration de MongoDB à ChatWithMe

Alphorm.com

Liaison avec la persistance des données

Intégration de MongoDB à ChatWithMe

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

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

Plan

- Let's code

Exemple

Let's code

Side project: 'ChatWithMe'

Ce qu'on a couvert

- Dans cette vidéo, nous avons intégré Mysql à ChatWithMe
- Prochaine chapitre
  - Intégration de Sequelize à ChatWithMe

Alphorm.com

Bonus

PM2, le monitoring

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

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

Plan

- Qu'est ce que PM2 ?
- La gestion d'un processus
- Les actions sur le processus
- Les logs
- Lancer un processus avec des arguments
- Les limites

Qu'est ce que PM2 ?

- PM2 est un gestionnaire de processus de production pour les applications Node.js avec un load balancer intégré.
- Il permet de garder les applications vivantes et les redémarrera si un problème intervient.
- Il permet aussi de recharger le code de l'application sans couper les files de description (Reload à chaud)
- https://github.com/Unitech/pm2
- Installation :
  - $> npm install –g pm2

La gestion d'un processus

- Démarrer le processus :
  - $> pm2 start bin/www
  - $> pm2 start bin/www --name ChatWithMe
- PM2 permet de démarer le processus en cluster mode, cela permettra de profiter de l'ensemble des processus de la machine.
  - $> pm2 start bin/www -i 0 # démarre avec autant de processus que de processeur.
- Pour lister les processus
  - $> pm2 list

La gestion d'un processus

- Afficher plus d'informations sur un processus en particulier
  - $> pm2 describe 0
- Monitorer les process lancés
  - $> pm2 monit
- Retire un processus de la liste
  - $> pm2 delete 0
  - $> pm2 delete all

Les actions sur le processus

- Différente actions :
  - $> pm2 stop all # Stop tous les processus
  - $> pm2 restart all # Restart tous les processus
  - $> pm2 reload all # Redémarrer un processus sans aucun downtime
  - $> pm2 gracefulReload all # Stop un processus, puis le relance
  - $> pm2 stop 0 # Stop un processus spécifique
  - $> pm2 restart 0 # Restart un processus spécifique

Les logs

- Gérer les logs
  - $> pm2 logs [--raw] # Affiche tous les logs
  - $> pm2 flush # Vide tous les logs

Lancer une processus avec des arguments

- Démarrer un processus avec des arguments
  - **$> pm2 start app.js -- -a 23** # Passe l'argument '-a 23' à app.js
- Spécifier les logs explicitement
  - **$> pm2 start app.js -e err.log -o out.log**

Les limites

- Le module **cluster** de NodeJS ne permet pas de faire fonctionner **socket.io**
- https://github.com/elad/node-cluster-socket.io
- Vu que **PM2** utilise le module natif pour les clusters, PM2 **ne supporte pas non plus les socket.io**.

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Un projet de monitoring de processus NodeJS pour une **mise en production** !
  - Lancer en cluster un processus
  - Gérer les logs
- Prochaine vidéo :
  - Keymetrics, la supervision

Alphorm.com

Bonus

Keymetrics, la supervision

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

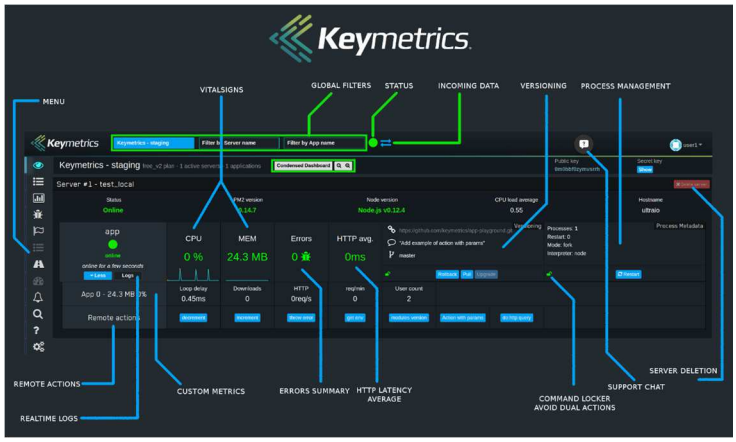
Édouard FERRARI
Formateur et Consultant indépendant
Contact : contact@ferrari.wf

Plan

- Qu'est-ce que Keymetrics ?
- Inscription
- Lien avec PM2
- Ajout de sondes

Qu'est-ce que Keymetrics ?

- Keymetrics permet d'afficher des statistiques remontées par pm2
  - https://keymetrics.io/



Inscription

- Inscription :
  - https://app.keymetrics.io/#/login

Lien avec PM2

- Pour lier Keymetrics avec un PM2
  - $> pm2 link xxxxxxxxxxx xxxxxxxxxx [hostname]

Ajouter des sondes

- C'est le module pmx qui permet d'ajouter des sondes :
  - npm install –save pmx
- Faire un wrapper (singleton) pour initialiser le module :

```
var pmx = require('pmx').init({
    network : true, // Network monitoring at the application level
    ports : true, // Shows which ports your app is listening on (default: false)
});
```

- Il existe **4 types** de sondes :
  - « **Simple metics** » : Remonte un nombre
  - « **Counter** » : Incrémente et décrémente un nombre
  - « **Meter** » : Remonte un nombre sur le temps, ex Nombre de requêtes par seconde.
  - « **Histogram** » : Garde un réservoir de valeur sur les 5 dernières minutes

Exemple

Ajout de nouvelle sonde

Side project : 'ChatWithMe'

Ce qu'on a couvert

- Dans cette vidéo, nous avons vu :
  - Comment superviser un projet NodeJS
  - Comment remonter certaines informations métier
- Prochain chapitre :
  - La conclusion !

Conclusion

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

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

Ce qu'on a couvert

1. Comment installer et mettre en place un **Express**
    1. Système de **route**
    2. Gestion des **fichiers statiques**
    3. Comment ajouter nos propres **middlewares**
    4. Comment utiliser un moteur de **template**
2. Deux différentes méthodes de développement **asynchrone:**
    1. **Async**
    2. Les **promesses** avec **Q**
3. La communication en temps raille avec les clients (front-end) grâce à **Socket.io**

1. Comment **sauvegarder des données** en couvrant les bases de donnée suivante :
    1. Mysql
    2. MariaDB
    3. SQLite
    4. Postgres
    5. MSSql
    6. Redis
    7. MongoDB
2. Développer un projet NodeJS en **MVC.**
3. Comment monitorer et superviser le projet avec **PM2** et **Keymetrics**

Avez-vous des Questions / Remarques / Commentaires ?

A bientôt ☺

Keep in touch !

E-mail : edouard@ferrari.wf
Linkedin : https://fr.linkedin.com/in/edouardferrari
Twitter : https://twitter.com/edouard_ferrari
Alphorm : http://www.alphorm.com/formateur/edouard-ferrari

