INF4375 - Paradigmes des échanges Internet

Node.js

Jacques Berger

Objectifs

Présenter le Javascript côté serveur

Prérequis

Javascript

Node.js

Javascript côté serveur

Entrées/sorties asynchrones

Événementiel

"Event-driven asynchronous server-side Javascript"

Node.js

Version actuelle : 8.6.0

Projet créé en 2009

Popularité en hausse depuis 2011

Node.js

Conçu pour les applications web, notamment les serveurs web

Reconnu pour sa rapidité d'exécution

Utilise le moteur Javascript de Chrome : V8

Majoritairement écrit en C++

Syntaxe

C'est du Javascript

Pour les fonctionnalités d'I/O, on utilise des modules supplémentaires

Voir un exemple d'Hello World

Concepts

Programmation fonctionnelle

Programmation événementielle

Programmation fonctionnelle

Javascript permet des mécanismes de programmation fonctionnelle

Les fonctions sont des données

On peut stocker des fonctions dans des variables, les passer en paramètre à une autre fonction, etc.

Programmation fonctionnelle

Javascript permet de créer des fonctions anonymes

Ces fonctions peuvent être passées directement en paramètre à une autre fonction

```
fs.readFile("textfile", function(err, data) {
  console.log("Voici les données du fichier : " + data);
});
```

Programmation fonctionnelle

Node.js fait une utilisation intensive de cette possibilité

Notamment, toute la gestion des entrées et sorties est faite de façon asynchrone à l'aide de fonctions passées en paramètre, servant ainsi de callback

Callback

Un callback est une fonction qui sera invoquée par une autre fonction

Dans l'exemple précédent, c'est la fonction readFile qui fera l'appel à la fonction anonyme qu'elle reçoit en paramètre

Lors de l'appel à readFile, la fonction anonyme n'est qu'une donnée, elle sera exécutée plus tard

Traitement asynchrone

Les entrées et sorties sont des opérations généralement lentes :

- Requête sur une base de données
- Lecture/écriture d'un fichier
- Requête HTTP
- Appel de service web

Traitement asynchrone

Node.js a été conçu pour traiter ces opérations de façon asynchrone

Ainsi, le serveur n'a pas besoin d'attendre la fin de ces opérations pour continuer son exécution

Traitement asynchrone

On donne un callback à la fonction qui fait l'opération lente et c'est cette fonction qui exécutera le callback

Voir un exemple avec la lecture d'un fichier texte

Programmation événementielle

Node.js est single-threaded

C'est-à-dire qu'il n'y a qu'un seul thread qui exécute du code utilisateur

On utilise plutôt la programmation événementielle

Programmation événementielle

Les tâches asynchrones sont placées dans une file en attendant qu'un événement survienne

- L'arrivée de données d'un serveur web
- La fin de la lecture d'un fichier
- L'arrivée du résultat d'une requête SQL
- Etc.

Programmation événementielle

Lorsqu'un événement débloque une tâche dans la file, le callback que nous lui avions donné est exécuté

Cette méthode change complètement la façon dont on gère les entrées/sorties et leurs erreurs possibles

Serveur web

Node.js sert très souvent de serveur web

Le module http lui permet d'écouter des connexions sur un port en particulier

Le callback qui traite une connexion prend en paramètre un objet correspondant à la requête et un objet correspondant à la réponse à fournir au client

Serveur web

Nous devrons alors générer un résultat et l'envoyer au client

Observons quelques exemples

Modules

On peut séparer les fonctionnalités dans des modules

Favorise la modularité
Favorise la réutilisation
Améliore la maintenabilité

Modules

La portée globale des variables et des fonctions en Javascript est limitée à un module, c'est-à-dire un fichier .js

Pour permettre à d'autres modules d'utiliser les fonctionnalités d'un module, il faut exporter ces fonctionnalités

Modules

Pour exporter une fonction ou une variable, on l'affecte à l'objet exports

```
exports.extractNumberOfPosts = extractNumberOfPosts;
```

Modules

Pour utiliser les fonctionnalités exportées d'un autre module, on utilise require

```
var jberger = require("./jberger.js");
```

API

Plusieurs modules de base viennent automatiquement avec Node.js (ex.: fs, http)

Plusieurs autres modules peuvent être téléchargés à l'aide du logiciel npm

npm est un programme qui vient avec Node.js

API

Pour utiliser un package externe, il faut d'abord l'installer avec npm

```
npm install async
```

API

npm téléchargera le module externe et l'installera dans un répertoire node_modules

Tous les modules externes de l'application doivent être dans ce répertoire

package.json

Lorsqu'on crée un projet Node.js, on crée également un document package.json à la racine du projet

On y spécifie :

- Le nom du projet
- La description du projet
- La version
- L'auteur
- Etc.

package.json

On peut également y spécifier les dépendances du projet envers des modules externes, favorisant ainsi la publication des sources du projet

Si on installe un module avec l'option --save, il sera ajouté au manifest automatiquement

```
npm install async --save
```

package.json

L'option --save ajoutera le module et sa version à la propriété dependencies du package.json

Ensuite, nous n'avons qu'à faire une commande pour installer toutes les dépendances du projet

```
npm install
```

Modules globaux

Certains modules peuvent être installés globalement (pour tous les projets) plutôt que localement (pour un seul projet)

On fait cette installation avec une option globale

```
npm install -g moment
```

Plus loin...

Node.js

http://nodejs.org/

npm

https://npmjs.org/