
node
JS
express
mongoDB

Node.js

- Node.js est une plate-forme construite sur le moteur d'exécution JavaScript de Chrome (V8 Engine) par Ryan Dahl en 2009.
- Il permet de créer facilement des applications réseau rapides et évolutives.
- Node.js utilise un modèle d'E / S non bloquant piloté par les événements qui le rend léger et efficace, parfait pour les applications en temps réel gourmandes en données qui s'exécutent sur des appareils distribués.
- Les applications Node.js sont écrites en JavaScript et peuvent être exécutées dans l'environnement Node.js sous Windows, Linux, etc.
- Node.js fournit également une riche bibliothèque de divers modules JavaScript qui simplifie le développement d'applications.

Qui utilise Node.js ?

Caractéristiques

- **Asynchrone et piloté par les événements** : toutes les API de la bibliothèque Node.js sont asynchrones, c'est-à-dire non bloquantes. Cela signifie essentiellement qu'un serveur basé sur Node.js n'attend jamais qu'une API retourne des données. Le serveur passe à l'API suivante après l'avoir appelée et un mécanisme de notification des événements de Node.js aide le serveur à obtenir une réponse de l'appel d'API précédent.
- **Très rapide** : construite sur le moteur JavaScript V8 de Google Chrome, la bibliothèque Node.js est très rapide dans l'exécution de code.

- **Single Threaded mais hautement évolutif** : Node.js utilise un modèle à thread unique avec boucle d'événements. Le mécanisme d'événement aide le serveur à répondre de manière non bloquante et rend le serveur hautement évolutif par opposition aux serveurs traditionnels, comme Apache, qui créent des threads limités pour gérer les demandes.
- **No Buffering** (pas de mise en mémoire tampon) : Les applications Node.js ne mettent aucune donnée en buffer. Ces applications sortent simplement les données par blocs.
- **License** : Node.js est publié sous la licence MIT.

Principe

- Node.js permet de concevoir des sites programmés en JavaScript côté serveur.
- Il ne s'agit pas d'un simple langage de script tournant sur un serveur préexistant ; en effet, il faut coder soi-même les différentes couches du serveur.
- Il faudra ainsi mettre en place :
  - le serveur HTTP lui-même, avec la gestion des différents messages HTTP
  - le routage des requêtes vers les bonnes pages du serveur
  - de quoi récupérer les paramètres saisis dans l'URL
  - un système permettant d'envoyer du contenu au client en fonction de l'URL
  - un dispositif permettant, en JavaScript, de gérer des accès au système de fichiers pour permettre non seulement la lecture, mais aussi l'écriture et même l'envoi de fichiers par l'utilisateur.

Installation et lancement

- Afin de télécharger Node.js, en vue de son installation, on a besoin d'accéder à :

https://nodejs.org

- Puis, il faut ouvrir la fenêtre CMD et exécuter les commandes ci-dessous pour inspecter la version de Node.js et celle de NPM :

  - node -v
  - npm -v

Application

- Créer un fichier, appelé salut.js, contenant : console.log("Bonjour"). Puis le lancer en tapant : **node** salut.js

Remarque :

nodemon qui s'installe à l'aide de npm, soit globalement avec npm install -g nodemon ou localement, permet de redémarrer automatiquement le serveur si un des fichiers constitutifs de l'application est modifié, ce qui est très commode en phase de développement.

Pour cela, il suffit de remplacer la ligne "start": "node index.js" par : "nodemon index.js".

Mise en place d'un serveur

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

var monServeur=function(requete, reponse){
  reponse.writeHead(200) ;
  reponse.end('Bonjour') ;
}

var serveur = http.createServer(monServeur) ;

serveur.listen(8080) ;
```

permet d'importer des modules

la requête envoyée par l'utilisateur et la réponse à lui renvoyer

permet d'écrire dans le header HTTP et de renvoyer un code 200 (« tout va bien »).

la chaîne Bonjour est envoyée au client

méthode de http, qui prend comme paramètre, une fonction anonyme (un callback) soit le nom d'une fonction. Cette méthode renvoie une référence à un objet serveur, qui permettra de le paramétrer et le contrôler.

le serveur écoute les requêtes et prend comme paramètre un numéro de port

- **Solution 2 :**

```
var http = require('http');
var monServeur= http.createServer(function(requete, reponse) {
    reponse.writeHead(200);
    reponse.end ('Bonjour');
    });
monServeur.listen(8080);
```

- **Remarque** :

Le client et le serveur communiquent en se basant sur la norme HTTP. Pour cela, le serveur doit indiquer le type de données qu'il s'apprête à envoyer au client. Ces différents types de données sont :

- Du texte brut : text/plain
- Du HTML : text/html
- Du CSS : text/css
- Une image JPEG : image/jpeg
- Une vidéo MPEG4 : video/mp4
- Un fichier ZIP : application/zip
- etc.

Ce sont ce qu'on appelle les types MIME. Ils sont envoyés dans l'en-tête de la réponse du serveur.

Exemple :

reponse.writeHead(200, {"Content-Type": "text/plain"}) : MIME de la réponse est de type texte.

Application

- Créer un mini-serveur web qui renvoie un message "Salut tout le monde", quelle que soit la page demandée. Le code de réponse est 200 en ajoutant un paramètre qui indique le type MIME de la réponse qui s’agit de HTML. Ce serveur est lancé sur le port 8080.

Routage

- Le routage est une association entre des URLs et des contenus que le serveur doit fournir.
- On va se servir d'une propriété de l'objet « requete », qui va permettre de récupérer des informations sur l'URL.
- Pour cela, il va falloir importer un autre module de node, le module **url** en ajoutant ***var url = require('url')***.
- Ensuite, il faut "parser" la requête du visiteur pour obtenir le nom de la page demandée : ***url.parse(requete.url).pathname;***
- ***reponse.end*** est une méthode qui permet de publier des données, et qui signale également la fin de l'envoi de ces données.
- Si le volume de données est important, il est préférable de passer par des ***reponse.write(donnees),*** qui peuvent être successifs, avant de conclure par un ***reponse.end*** pour terminer la publication.

Exemple :

```
var http = require('http');
var url = require('url');
var monServeur= http.createServer(function(requete, reponse) {
var page = url.parse(requete.url).pathname;
console.log(page);
reponse.writeHead(200, {"Content-Type": "text/plain"});
reponse.write('Bonjour tout le monde');
reponse.end();
});
monServeur.listen(8080);
```