Démarrer avec Node.js

1. Objectifs

- Être capable de créer un serveur node.js

2. Présentation

- Node.js est un environnement d'exécution JavaScript open source et multiplateforme. C'est un outil populaire pour presque tous les types de projets!
- Node.js a été développé en 2009 par Ryan Dahl en réponse à la frustration causée par problèmes de concurrence, en particulier lorsqu'il s'agit de services Web.
- Google venait de sortir le moteur JavaScript V8 pour le navigateur Web Chrome, qui était hautement optimisé pour le Web circulation. Dahl a créé Node.js au-dessus de V8 comme un environnement côté serveur qui correspond à la environnement côté client dans le navigateur.
- Le résultat est un environnement côté serveur extrêmement évolutif qui permet aux développeurs de combler facilement le fossé entre le client et le serveur.
- Le fait que **Node.js** soit écrit en JavaScript permet aux développeurs de naviguer facilement entre le code client et le code serveur et même réutiliser le code entre les deux environnements.
- **Node.js** a un excellent écosystème avec de nouvelles extensions écrites tout le temps. Le Node.js l'environnement est propre et facile à installer, à configurer et à déployer. Littéralement en seulement une heure ou deux, vous pouvez avoir un serveur Web Node.js opérationnel.

3. Notre premier serveur Web

- Une fois que nous avons installé Node.js, construisons notre premier **serveur Web**.
- Pour tester **Node.js**, vous allez dans un premier temps créer un code JavaScript qui va être le plus simple possible, et le faire exécuter par Node.js.
- Créez un fichier nommé **app.js** contenant le contenu suivant:

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

const hostname = '127.0.0.1';
const port = 3000;
```

apcpedagogie.com

```
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

4. Ou le contenu suivant:

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

5. Maintenant, exécutez votre serveur Web en utilisant node app.js. Visitez http://localhost:3000et vous verrez un message disant "Hello World".
6. Reportez-vous à l' introduction à Node.js pour un guide plus complet sur la prise en main de Node.js.

7. ## Explication détaillée

   - Passons à l'explication de ce programme, ligne par ligne (ou presque).
   - `const http = require('http');`
   - Node.js
   - permet d'utiliser des outils tiers appelés modules. Il inclut de nombreux modules de base, parmi eux le module `http` permettant de créer un **serveur web**.
   - Par la fonction `require`, nous demandons à **Node.js** d'inclure le module http, et nous décidons d'accéder à ses méthodes via un objet http.
   - La méthode `createServer` de l'objet http permet comme son nom l'indique de créer un serveur, en l'occurrence un serveur web (HTTP). Nous reviendrons sur la fonction qu'elle prend en paramètres dans quelques instants.
   - La méthode `listen` du serveur permet de lancer l'écoute, ici sur le port 1337 de l'hôte 127.0.0.1 (localhost). Autrement dit, c'est grâce à cette méthode qu'on lance notre serveur.

apcpedagogie.com

- La fonction qui est passée en paramètre à `createServer` est celle qui sera appelée à chaque requête sur notre serveur. Elle prend elle-même deux paramètres : **la requête (`req`)** et **la réponse (`res`)**. (Nous aurions pu appeler les paramètres autrement évidemment, le principal étant qu'ils soient là et qu'ils soient deux.)
- La requête est un objet permettant d'accéder aux détails sur la requête envoyée au serveur : en-têtes HTTP, paramètres **GET** ou **POST**, etc. Nous ne l'utilisons pas ici.
- La réponse (`res`) est l'objet qui nous permet d'envoyer une réponse à la requête. Dans notre cas, nous effectuons deux actions sur cette réponse :
  - nous définissons l'en-tête HTTP Content-type à _text__/plain_, ce qui permet d'indiquer que le contenu que nous renvoyons est de type texte. Il pourrait être text/html, application/json, bref ce que l'on veut, tant qu'il s'agit d'un type MIME standard ;
  - nous terminons la requête en envoyant le texte "Hello World" (suivi d'un retour à la ligne, qui n'est pas indispensable ici).
- La fonction `console.log` permet d'afficher un message dans la console (la sortie de la ligne de commande), ce qui nous sert ici à indiquer que le serveur a bien démarré.

Sommaire du cours Node.js

Source : https://apcpedagogie.com/demarrer-avec-node-js · téléchargé le 04/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com