

Back End JavaScript



"Le plus grand arbre est né d'une graine menue." (Lao-Tseu) - 2014 © Gilles Chamillard

JavaScript et le développement Web

FRONT-END



FULL-STACK



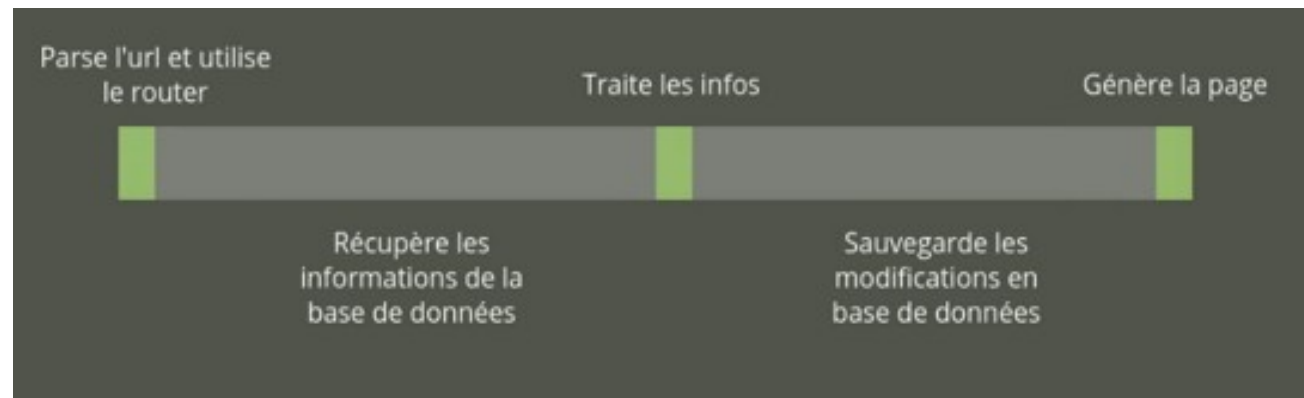
BACK-END

NodeJS : un serveur Web en JavaScript

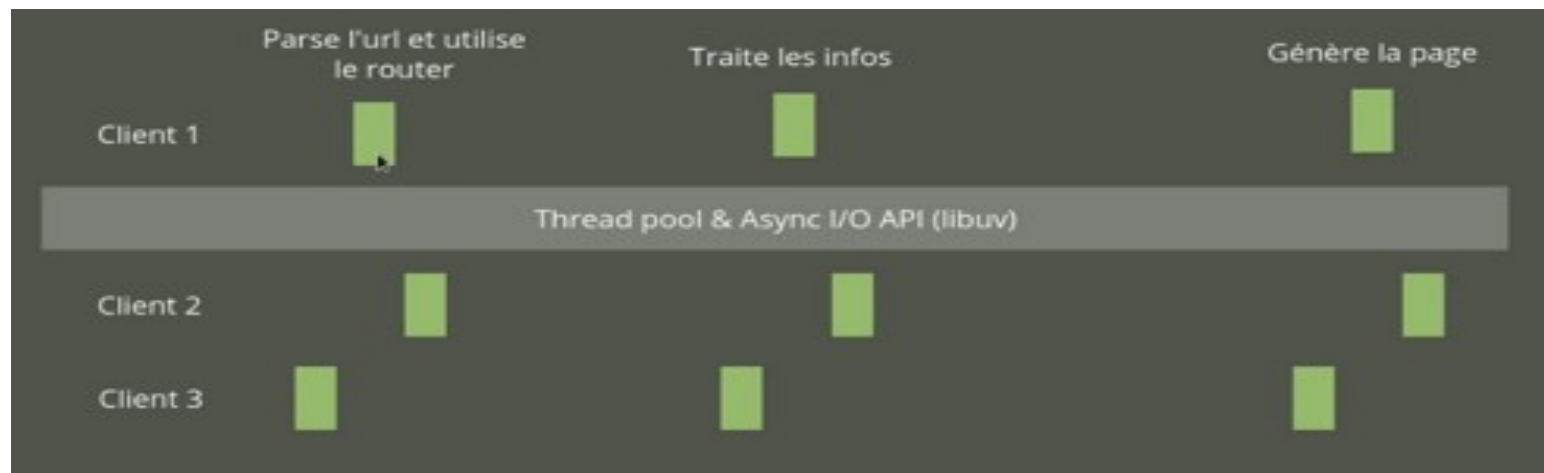
NodeJS utilise le langage JavaScript **côté serveur**, alors qu'il se trouve naturellement utilisé côté client dans un navigateur comme par exemple avec le framework **Angular**.

- **Fonctionnement**

Normalement, voici le traitement d'un script (par exemple PHP) effectuant une requête sur une base MySQL (avec ou sans PDO) et retournant une page HTML :



Voici la même chose, mais avec NodeJS :



NodeJS répond classiquement à des événements du navigateur mais avec des échanges **asynchrones**, donc très rapides car non bloquant pour le serveur.

NodeJS : bases du code (1)

- *Fonctions de « callback »*

NodeJS répond lors d'une requête à des fonctions dites de callback avec le principe d'une fonction anonyme en paramètre :

```
var nom_fonction_callback = function (response) {  
    // Traitement de la réponse;  
});  
request(URL,nom_fonction_callback);
```

Ou en écrivant plus simplement :

```
request(URL, function (response) {  
    // Traitement de la réponse;  
});
```

La requête du client porte sur une page (ou script) comme paramètre entrant (la variable **URL**) pour une demande HTTP et la réponse est traitée par une fonction sans nom (il n'y a rien entre le mot clé `function` et la première parenthèse) et dont le résultat est retourné seulement une fois le traitement effectué (comme pour un script).

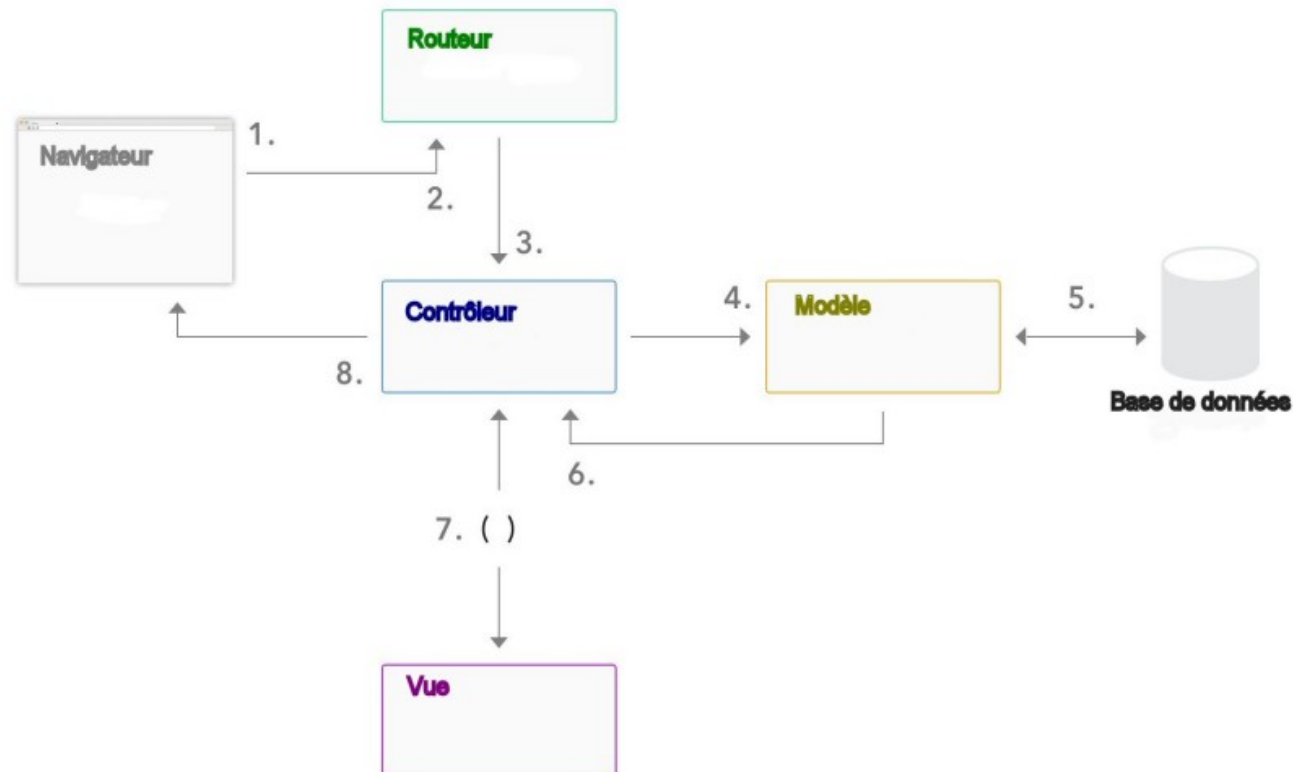
NodeJS est un langage dit de bas niveau, c'est-à-dire qu'il dispose de peu de librairies et qu'il faut donc tout écrire... Par exemple, le serveur HTTP ! NodeJS comporte heureusement un gestionnaire de paquets : **NPM** (comme Composer pour PHP) permettant l'installation des librairies existantes (ou modules).

NodeJS : bases du code (2)

- *Middleware et approche MVC*

Un **middleware** est un processus (ou traitement) qui se place entre la requête et l'envoi de la réponse. En fait, dans NodeJS, **tout est middleware**... **Express.js** est un module, ou plutôt un micro-framework permettant un développement plus facile d'une application NodeJS (utilisé dans ce cours).

L'approche **MVC** (Modèle/Vue/Contrôleur) repose respectivement sur la logique métier et données de l'application, le formatage et la disposition des données et le traitement des requêtes :



Deno : l'avenir ?

Deno a été créé par le créateur de NodeJS (noDe inversé) pour, selon ses dires, « combler les manques de NodeJS ».

Concrètement :

- Deno est plus sécurisé,
- Deno n'utilise plus NPM (la gestion des modules est directement intégré à Deno),
- Deno supporte pleinement JavaScript et TypeScript.

Né en 2018, **Deno** n'est pas encore tout à fait « mature » ni testé sur des systèmes en production. Il est très probable que les deux existeront conjointement...

