Partie 7
Frameworks JavaScript

Chapitre 7-1
Positionnement des frameworks JavaScript

1. Présentation générale des frameworks JavaScript

De très nombreux frameworks JavaScript existent, avec des positionnements fonctionnels différents.

Il ne peut être question, dans le cadre de ce livre réservé à des débutants en JavaScript, d'en faire une revue exhaustive.

Ils ont tous les points communs suivants : masquer la complexité du langage JavaScript, apporter de la robustesse dans les développements et aussi permettre, pour certains d'entre eux, d'interagir avec des bases de données.

1.1 Frameworks « front-end »

Les plus populaires des frameworks dits « front-end », c'est-à-dire gérant le côté interface utilisateur des applications web ou mobiles (téléphones mobiles, tablettes…) sont :

– Angular, framework développé par Google (la première version était connue sous l'appellation AngularJS)
– React JS (ou React), framework développé par Facebook
– Vue.js
– Svelte

1.2 Frameworks « back-end »

Pour les interactions avec les systèmes de gestion de bases de données, des frameworks dits « back-end » existent. Ils sont souvent eux-mêmes basés sur Node.js, qui est un environnement d'exécution multiplateforme Open Source exécutant du code JavaScript en dehors d'un navigateur (dans un runtime).

Node.js permet de concevoir des services d'accès à des Bases De Données et à des ressources disponibles sur Internet. Il fonctionne parfaitement sur Windows, Linux ou encore macOS.

Nous verrons par exemple que l'accès aux données pour le framework Svelte peut être assuré par les frameworks Express ou Sapper, tous deux basés sur Node.js.

1.3 Solutions de développement « hybride »

Pour terminer ce rapide panorama concernant les frameworks JavaScript, n'oublions pas les solutions dédiées aux développements pour périphériques mobiles (smartphones et tablettes). Dans un précédent livre, publié en 2020 aux Éditions ENI (Java et Ionic - Développement mobile pour Android : natif vs hybride), le framework Ionic a été présenté. Ce framework est lui-même basé sur d'autres frameworks, dont le très réputé Angular (soutenu par Google) et Apache Cordova.

Un chapitre du présent livre sera aussi consacré au Framework React Native (assez proche de React qui, lui, est réservé aux applications web). React Native permet très facilement à des développeurs ayant déjà une réelle expérience en React de concevoir des applications pour mobiles. Ce framework extrêmement utilisé et supporté par Facebook est une alternative plus que crédible à Ionic. Les applications React Native sont facilement déployables sur les mobiles fonctionnant sous systèmes d'exploitation Android et iOS (iPhone, iPad).

2. Les frameworks Node.js, Svelte, React et React Native

Comme indiqué précédemment, un court chapitre (Installation de Node.js) sera consacré à l'installation du framework Node.js, socle sur lequel fonctionnent les frameworks Svelte, React et React Native.

Le framework Svelte, relativement récent, est un challenger crédible pour React (React JS) et Vue.js. Svelte, présenté dans le chapitre Framework Svelte, possède de nombreux atouts techniques. Il bénéficie par contre pour l'instant d'une communauté de développeurs plus restreinte que celles des deux acteurs principaux (React et Vue.js).

Dans le cadre de ce livre, un choix éditorial a été fait de ne pas évoquer Vue.js. Vous trouvez, toujours aux Éditions ENI, des ouvrages dédiés exclusivement à Vue.js, notamment le livre Vue.js - Développez des applications web modernes en JavaScript de Yoann GAUCHARD.

Le chapitre Framework React sera celui dédié à React. Comme dans le chapitre consacré à Svelte, après une rapide présentation des concepts de base, de nombreux exemples seront proposés et largement commentés.

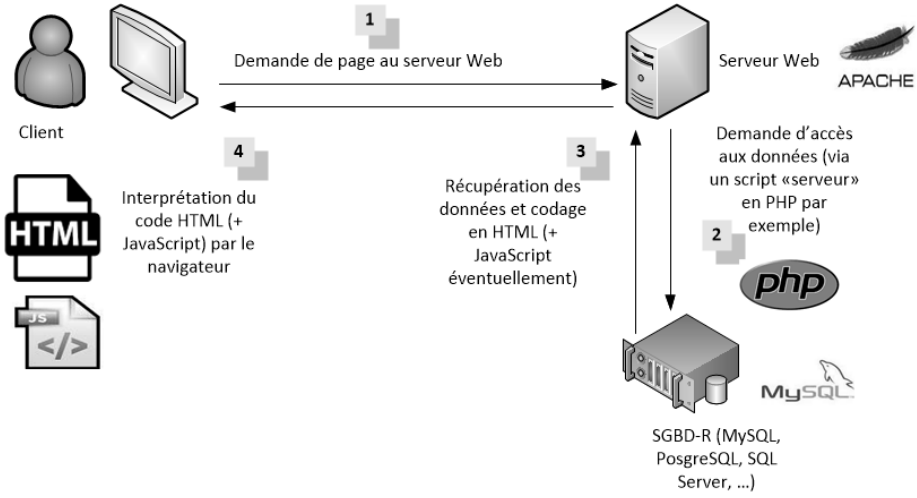
Le livre se terminera au chapitre Framework React Native avec un exposé consacré à React Native, la version du framework React permettant le développement d'applications pour mobiles.

Chapitre 7-2
Installation de Node.js

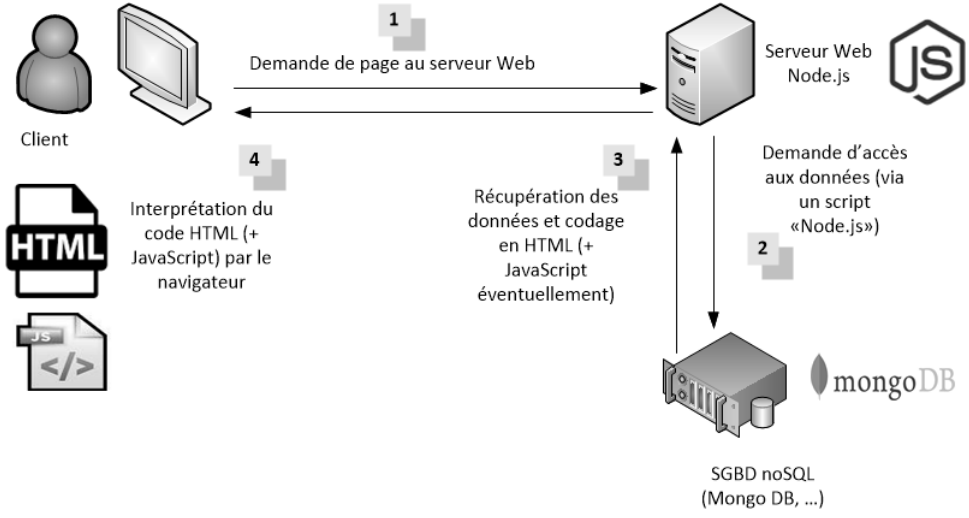
1. Présentation du framework Node.js

JavaScript a longtemps été cantonné à une utilisation côté client. On utilisait JavaScript seulement pour ajouter de l'interactivité dans les pages web (animations, contrôles de saisie...).

Pour les accès aux Bases De Données distantes, comme MySQL, les applications web intégraient par ailleurs des scripts orientés serveur, là aussi souvent développés en langage PHP.



Avec Node.js, il est bien sûr toujours possible d'utiliser JavaScript côté client pour manipuler les pages HTML. En plus, Node.js propose un environnement côté serveur qui permet aussi d'utiliser le langage JavaScript pour générer des pages web. En clair, il vient en remplacement de langages serveur comme PHP, Java EE, etc.



Chapitre 3
Comprendre les fondamentaux de Vue.js

1. Installation

1.1 Une version par environnement

Nous allons voir dans ce chapitre qu'il est possible d'installer Vue.js de différentes manières selon l'outillage d'un projet existant ou si vous démarrez un nouveau projet d'application SPA.

Il existe dans tous les cas deux versions de la librairie :

– une version de développement : elle comporte un mode debug permettant d'afficher des avertissements dans la console et d'interagir avec plusieurs outils d'aide au développement (extensions dans le navigateur) ;

– une version de production : c'est une version minifiée et allégée du mode debug et du code pour les outils d'aide au développement. Elle s'utilise lorsque votre site est en ligne pour que son chargement soit le plus rapide possible dans le navigateur.

Remarque

Vue.js n'est pas compatible avec IE 8 et ses versions antérieures. Il ne peut être utilisé que par des navigateurs compatibles avec les versions ES 5 et ultérieures de JavaScript.

1.2 Via téléchargement manuel

- Pour commencer à utiliser Vue.js, téléchargez la dernière version de la librairie sur le site officiel : https://vuejs.org/v2/guide/installation.html

- Téléchargez la version de développement. Une fois le fichier vue.js téléchargé, placez-le dans le répertoire de votre site web et chargez-le dans votre page HTML via la balise .`<script>`

```
<script src="vue.js"></script>
```

1.3 Via l'inclusion d'un CDN (le plus simple)

- Si vous ne souhaitez pas télécharger la librairie pour ne pas alourdir le poids de votre projet, vous pouvez également charger ce fichier depuis un serveur distant via le CDN (*Content Delivery Network*) suivant :

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

C'est la méthode d'installation la plus simple pour commencer à utiliser les fonctionnalités de Vue.js.

Remarque

Comme la librairie est hébergée sur un serveur distant, vous ne pourrez pas travailler sur votre projet hors ligne.

- Pour une mise en production, utilisez plutôt le CDN suivant pour la dernière version stable (v2.6.11) :

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.11"></script>
```

Si vous utilisez les modules ES natifs de JavaScript, il existe également un build compatible à partir de la version 2.6 de Vue.js :

```
<script type="module">
  import Vue from 'https://cdn.jsdelivr.net/npm/vue@2.6.0/dist/
vue.esm.browser.js'
</script>
```

1.4 Via npm ou yarn pour de plus gros projets

1.4.1 Téléchargement du package vue

npm (*Node Package Manager*) est le plus grand registre de librairies JavaScript du monde (plus de 800 000). Chaque librairie de ce registre se charge comme un module CommonJS ou ES, suivant la syntaxe de module dans laquelle elle a été développée.

C'est aussi un gestionnaire de dépendances qui s'installe sur votre machine avec le framework Node.js.

Pour des projets Vue.js de taille intermédiaire qui vont utiliser plusieurs librairies JavaScript il est recommandé d'utiliser un gestionnaire de dépendances comme npm ou yarn (surcouche de `npm`) pour les installer de manière optimisée. Il est nécessaire ensuite d'utiliser un empaqueteur de modules comme Webpack ou Browserify pour les utiliser.

- Vous pouvez télécharger Node.js à l'adresse suivante :
  https://nodejs.org/fr/download/
- Une fois installé, ouvrez le terminal de votre machine, tapez la commande suivante et suivez les instructions pour créer un nouveau projet :

```
npm init
```

Un fichier package.json est alors créé. Il servira de registre à npm pour connaître les différents packages qu'utilise votre projet.

- Tapez la commande suivante à la racine de votre projet web pour télécharger le package `vue` :

```
npm install vue
```

Cette commande crée un répertoire `vue` dans un répertoire `node_modules` de votre projet et ajoute la référence de ce module au fichier package.json.

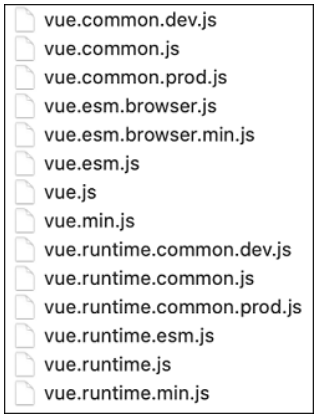
Si vous souhaitez ensuite partager votre projet, vous pouvez ainsi partager uniquement le code source sans le répertoire `node_modules`.

Avec la commande `npm install`, npm réinstalle automatiquement toutes les librairies dont votre projet a besoin en lisant le fichier package.json.

1.4.2 Explication des différents builds

Pour comprendre les différentes explications qui vont être fournies, il faut savoir que l'écriture de modules JavaScript a bien évolué ces dernières années. De plus, il est impossible de les utiliser directement dans le navigateur (excepté les modules ES natifs). Cela nécessite d'utiliser un empaqueteur capable d'analyser leur syntaxe pour charger ces fichiers dans un navigateur (cf. chapitre Notions essentielles JavaScript, section Utilisation des modules JavaScript).

Vous trouverez les différents builds de la librairie dans le répertoire `/node_modules/vue/dist` :



Voici un aperçu de leurs différences :

	UMD	CommonJS Module	ES Module (pour empaqueteurs)	ES Module (pour navigateurs)
Full	vue.js	vue.common.js	vue.esm.js	vue.esm.browser.js
Runtime-only	vue.runtime.js	vue.runtime.common.js	vue.runtime.esm.js	-
Full (production)	vue.min.js	-	-	vue.esm.browser.min.js
Runtime-only (production)	vue.runtime.min.js	-	-	-

Voici la définition des différents termes.

- **Full** : ce terme désigne des builds qui contiennent la partie Compiler et la partie Runtime.
- **Compiler** : il s'agit du code permettant de compiler les chaînes de caractères de la propriété `template` d'un composant Vue.js en une fonction de rendu. Ainsi, le code suivant :

```
// ceci a besoin d'un compilateur
new Vue({
  template: '<div>{{ hello }}</div>'
})
```

  deviendra après la compilation :

```
// ceci n'en a pas besoin
new Vue({
  render (h) {
    return h('div', this.hello)
  }
})
```

- **Runtime** : il s'agit du code qui s'occupe de créer l'instance Vue et les composants, de manipuler le DOM virtuel et de faire le rendu du DOM final, c'est-à-dire tout le reste. Si les composants Vue.js de l'application sont écrits au format monofichier avec l'extension .vue, la partie Compiler n'est pas utile dans le build. En revanche, cela nécessitera d'installer via npm ou yarn le module vue-loader sur votre projet si vous utilisez un build écrit avec la syntaxe ES Module, ou le module vueify si vous utilisez un build écrit avec la syntaxe CommonJS.
- **UMD (*Universal Module Definition*)** : c'est une syntaxe de module qui est compatible avec les modules CommonJS et les modules AMD. Ces fichiers ne nécessitent donc pas d'empaqueteur et peuvent être lus tels quels par le navigateur en les chargeant dans une balise `<script>`. Par défaut, la version téléchargeable manuellement sur le site officiel de Vue.js et via les CDN comprend la partie Compiler et la partie Runtime.
- **CommonJS Module** : il s'agit de l'ancienne syntaxe utilisée pour écrire des modules en JavaScript, avant l'apparition des modules ES natifs avec la version ES 6 de JavaScript. Ces fichiers sont utilisés si votre projet se sert de l'empaqueteur de modules Browserify ou de la première version de Webpack.
- **ES Module** : il s'agit de la nouvelle syntaxe utilisée nativement dans le langage JavaScript depuis la version ES 6 pour écrire des modules.
  - Il existe une version pour des empaqueteurs modernes comme Webpack ou Rollup. Cette version est optimisée pour être analysée plus rapidement par l'empaqueteur afin qu'il ne garde que les parties de la librairie qui sont utilisées dans votre application.

– La version "browser" permet aux fichiers d'être chargés directement dans le navigateur à l'aide de la balise `<script>` avec l'attribut `type` ayant pour valeur `module`.

Remarque

La version ES Module Browser permet de charger la librairie directement dans votre page. Cependant, pour éviter les problèmes de partage de ressources entre origines multiples (CORS - Cross-Origin Resource Sharing) dans votre navigateur, il est nécessaire d'utiliser un serveur local sur votre machine. Vous ne pourrez pas utiliser la librairie Vue.js sur une page index.html chargée dans le navigateur depuis votre explorateur de fichier (URL en file:///).

Sans la partie Compiler, la librairie est plus légère de 30 %. Il est donc recommandé, pour de gros projets, d'utiliser des composants monofichiers avec l'extension .vue et d'utiliser un empaqueteur moderne, comme Webpack, Rollup ou Parcel, avec le module vue-loader.

Si vous n'utilisez pas vue-loader ou vueify (utilise l'empaqueteur plus ancien Browserify) et que vous souhaitez utiliser un build avec la partie Compiler, il sera nécessaire de créer un alias vers le build approprié dans la configuration de votre empaqueteur.

1.5 Via Vue-CLI

Si vous ne souhaitez pas passer un ou deux jours à configurer un empaqueteur pour votre projet d'application SPA, Vue.js met à disposition un CLI (*Command Line Interface*) officiel pour démarrer rapidement, avec zéro configuration, un projet de développement d'application SPA.

Vue-CLI est un package npm construit au-dessus de l'empaqueteur Webpack et il fournit une commande `vue` dans votre terminal.

Vue-CLI est un système complet pour développer vos applications Vue rapidement et il fournit notamment :

– une interface graphique pour créer et gérer un projet Vue.js ;
– le rechargement à chaud (*hot reloading*), vous permettant de tester votre code sans attendre la compilation d'un build par Webpack pendant le développement ;
– une collection de plugins officiels intégrant les meilleurs outils frontend : transpileur, linter, préprocesseur CSS, outils de tests unitaires, etc.

Pour en savoir plus, vous pouvez vous reporter au chapitre Créer et déployer une application avec Vue CLI.

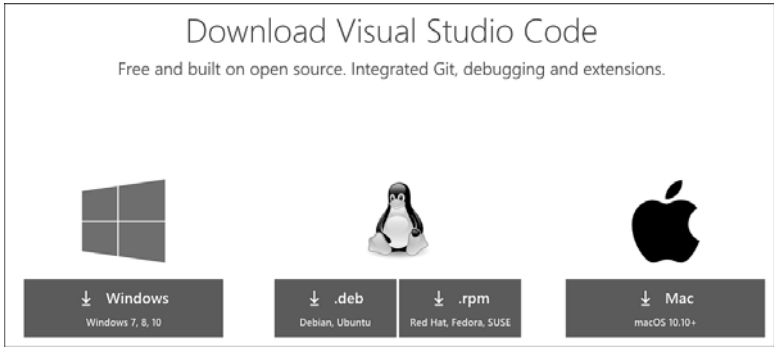
2. Outils de développements

2.1 VS Code et ses plugins

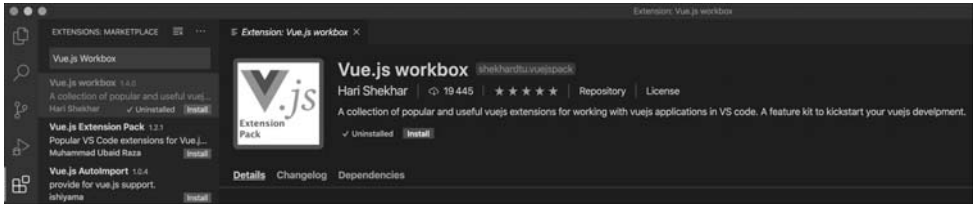
2.1.1 Installer et configurer VS Code

Visual Studio Code est un éditeur de code gratuit, édité par Microsoft, et l'un des plus utilisés aujourd'hui par les développeurs Vue.js. Il a la particularité d'être très léger et puissant à la fois. Il est également très extensible, grâce à ses milliers de plugins.

VS Code est disponible en téléchargement sur Windows, Mac et Linux à l'adresse suivante : https://code.visualstudio.com/Download



- Une fois installé, ouvrez-le.
- Cliquez sur le bouton  **Extensions** à gauche de la fenêtre.
- Tapez "**Vue.js Workbox**" :



- Cliquez ensuite sur le bouton vert **Install**.