Chapitre 3
Développement frontend

1. Introduction au développement frontend

Le développement frontend est une discipline cruciale dans la création d'applications web modernes. Il se concentre sur tout ce qui est visible et interactif pour l'utilisateur final. Contrairement au backend qui gère les opérations en coulisses, le frontend se concentre sur la création de la partie visible des sites web et des applications web, assurant ainsi une expérience utilisateur fluide et esthétique.

Dans cette section, nous allons explorer les concepts fondamentaux du développement frontend. Nous commencerons par définir ce qu'est le développement frontend et son importance. Ensuite, nous discuterons des raisons pour lesquelles React.js est devenu un choix populaire parmi les développeurs pour construire des interfaces utilisateur dynamiques. Enfin, nous introduirons SCSS, une extension du CSS classique, en mettant en lumière ses avantages et son utilité dans le développement de styles plus maintenables et modulaires.

1.1 Qu'est-ce que le développement frontend ?

Le développement frontend est la discipline qui concerne la création et la gestion de l'interface utilisateur d'un site web ou d'une application web. Cette partie du développement se focalise sur ce que l'utilisateur voit et ce avec quoi il interagit, incluant la mise en page, le design, les animations et les interactions.

Ses trois piliers principaux

Le frontend repose sur trois piliers principaux :

- **HTML (*HyperText Markup Language*)** : il s'agit du langage de balisage utilisé pour structurer le contenu sur le Web. Il permet de définir des éléments tels que les titres, les paragraphes, les listes, les liens et les images.
- **CSS (*Cascading Style Sheets*)** : c'est le langage de feuille de style utilisé pour décrire la présentation d'un document HTML. CSS contrôle l'apparence des pages web, notamment les couleurs, les polices, les espacements et les dispositions.
- **JavaScript** : il s'agit du langage de programmation qui permet d'ajouter de l'interactivité et des fonctionnalités dynamiques aux pages web. Grâce à JavaScript, il est possible de créer des applications web réactives qui réagissent aux actions de l'utilisateur en temps réel.

Les développeurs frontend utilisent également des frameworks et bibliothèques pour améliorer leur productivité et créer des interfaces utilisateur plus complexes et interactives. Parmi les plus populaires, on trouve React.js, Next.js, Vue.js, Nuxt.js et Angular.

Des pratiques et concepts importants

En plus de ces technologies de base, le développement frontend implique des pratiques et concepts importants tels que :

- **Responsive Design** : l'objectif est d'assurer que les interfaces s'adaptent et offrent une expérience optimale sur différents appareils et tailles d'écran (grand ou moyen écran PC, tablette ou smartphone).
- **Accessibilité (a11y)** : il s'agit de rendre les applications web utilisables par le plus grand nombre, y compris les personnes en situation de handicap.

– **Performance** : son but est d'optimiser le chargement et le fonctionnement des pages web pour offrir une expérience utilisateur fluide et rapide.

En somme, le développement frontend est un domaine essentiel qui lie l'esthétique à la fonctionnalité, garantissant que les utilisateurs finaux bénéficient d'une expérience agréable et intuitive lorsqu'ils interagissent avec une application web.

1.2 Pourquoi choisir React.js pour le frontend ?

React.js est une bibliothèque JavaScript développée par Facebook, utilisée pour construire des interfaces utilisateur dynamiques et interactives. Depuis sa création, React.js a gagné une popularité considérable parmi les développeurs frontend en raison de ses nombreux avantages et de sa flexibilité. Évoquons les raisons pour lesquelles React.js est souvent le choix privilégié pour le développement frontend.

Composants réutilisables

React.js repose sur une architecture de composants, où chaque partie de l'interface utilisateur est construite comme un composant autonome. Ces composants peuvent être réutilisés à travers l'application, ce qui permet de réduire la redondance du code et de faciliter la maintenance.

Virtual DOM

L'un des aspects innovants de React.js est l'utilisation du Virtual DOM. Au lieu de manipuler directement le DOM, React.js maintient une copie virtuelle de celui-ci. Lorsqu'une mise à jour est nécessaire, React.js compare le Virtual DOM à l'original et applique uniquement les modifications nécessaires. Cela améliore considérablement les performances et rend les mises à jour de l'interface utilisateur plus rapides et plus efficaces.

JSX

JSX est une extension syntaxique de JavaScript qui permet d'écrire du HTML directement dans le code JavaScript. Il rend le code plus lisible et facilite la création de composants React. Grâce à JSX, les développeurs peuvent voir la structure de l'interface utilisateur et la logique de l'application au même endroit, ce qui améliore la productivité et la compréhension du code.

Écosystème riche et communauté active

React.js bénéficie d'un écosystème vaste et en croissance constante, comprenant de nombreuses bibliothèques et outils complémentaires tels que Redux pour la gestion de l'état, React Router pour la navigation et Next.js pour le rendu côté serveur. De plus, React.js est soutenu par une communauté active et un large éventail de ressources, de tutoriels et de documentation, facilitant l'apprentissage et le dépannage.

Compatibilité et intégration

React.js peut être facilement intégré avec d'autres bibliothèques ou frameworks, ce qui le rend très flexible. Il peut être utilisé pour ajouter de nouvelles fonctionnalités à une application existante ou pour développer une application complète à partir de zéro. Cette compatibilité rend React.js adaptable à divers projets et besoins de développement.

Adoption par l'industrie

De nombreuses grandes entreprises, y compris Facebook, Instagram, Airbnb et Netflix utilisent React.js pour leurs applications web. Cette adoption à grande échelle témoigne de la fiabilité et de l'efficacité de React.js dans le développement d'applications modernes.

En conclusion, React.js offre une combinaison puissante de performances, de modularité et de flexibilité, ce qui en fait un excellent choix pour le développement frontend. Son architecture de composants, son utilisation du Virtual DOM et sa communauté active sont autant d'atouts qui permettent aux développeurs de créer des interfaces utilisateur performantes et maintenables.

1.3 Introduction à SCSS : avantages par rapport au CSS

SCSS (*Sassy CSS*) est une extension de CSS qui ajoute des fonctionnalités puissantes et des outils permettant de gérer les feuilles de style de manière plus efficace et maintenable. SCSS fait partie de Sass (*Syntactically Awesome Style Sheets*), l'un des préprocesseurs CSS les plus populaires.

Voici pourquoi SCSS est souvent préféré par les développeurs par rapport au CSS classique :

– **Variables** : SCSS permet d'utiliser des variables pour stocker des valeurs telles que les couleurs, les tailles de police ou les espacements. Les variables rendent le code plus maintenable et facilitent les modifications globales. Par exemple, changer une couleur de thème dans un fichier SCSS se répercute automatiquement sur tous les éléments utilisant cette variable.

```
$primary-color: #3498db;

body {
  color: $primary-color;
}
```

– **Imbrication** : SCSS permet l'imbrication des règles CSS, reflétant mieux la structure HTML et rendant le code plus lisible. Cela permet de regrouper des styles similaires et d'éviter la répétition des sélecteurs.

```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
    li {
      display: inline-block;
      a {
        text-decoration: none;
      }
    }
  }
}
```

– **Mixins** : les mixins sont des ensembles de règles CSS qui peuvent être réutilisés tout au long du fichier SCSS. Ils permettent de réduire la duplication du code et d'appliquer des styles complexes de manière cohérente.

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}
.box { @include border-radius(10px); }
```

– **Inheritance** : SCSS offre une fonctionnalité d'héritage qui permet de partager un ensemble de propriétés CSS d'un sélecteur à un autre en utilisant le mot-clé `@extend`. Cela réduit la redondance et permet de maintenir une structure de style plus propre.

```
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}
.message { @extend %message-shared; }
.success { @extend %message-shared; background-color: #cff;}
.error { @extend %message-shared; background-color: #fcc;}
```

– **Partials et importations** : SCSS permet de diviser le fichier CSS en plusieurs fichiers partiels, chacun contenant des fragments de code spécifiques. Ces fichiers partiels peuvent ensuite être importés dans un fichier principal, ce qui facilite la gestion et l'organisation des styles.

variables.scss :

```
_variables.scss
$primary-color: #3498db;
```

base.scss :

```
base.scss
body {
  color: $primary-color;
}
```

main.scss :

```
@import 'variables';
@import 'base';
```

– **Fonctions** : SCSS offre la possibilité de définir des fonctions, permettant d'effectuer des opérations et de renvoyer des valeurs. Ces fonctions peuvent être utilisées pour des calculs dynamiques dans les feuilles de style.

```
@function calculate-rem($px) {
  @return $px / 16px * 1rem;
}
.container {
  padding: calculate-rem(32px);
}
```

En conclusion, SCSS apporte une multitude d'améliorations par rapport au CSS classique, rendant les feuilles de style plus puissantes, modulaires et maintenables. L'utilisation de variables, l'imbrication, les mixins, l'héritage, les partiels et les fonctions permettent aux développeurs de gérer plus efficacement les styles de leurs applications web et de maintenir un code propre et cohérent.

2. Présentation de React.js

React.js est une bibliothèque JavaScript open source créée par Facebook, utilisée pour construire des interfaces utilisateur dynamiques et performantes. Depuis son lancement en 2013, React.js a révolutionné la manière dont les développeurs conçoivent et maintiennent les interfaces utilisateur, grâce à son approche basée sur les composants et son utilisation efficace du Virtual DOM.

Cette section offre une vue d'ensemble de React.js, en commençant par les concepts fondamentaux tels que les composants, le JSX et le Virtual DOM. Nous examinerons ensuite les étapes nécessaires pour installer et configurer un environnement de développement React.js, afin de permettre aux développeurs de démarrer rapidement avec cette bibliothèque puissante.

Nous illustrerons ces concepts par un exemple pratique : la création d'une application "Hello World" en React.js. Ce premier projet vous guidera à travers les bases de la création et du rendu de composants React.

Enfin, nous proposerons des exercices pratiques pour renforcer les concepts appris. Vous serez amené à modifier l'application "Hello World" pour y inclure un compteur de clics, ce qui vous permettra de vous familiariser davantage avec les fonctionnalités de React.js et d'acquérir une expérience pratique précieuse.

En somme, cette section vise à fournir une compréhension complète et pratique de React.js, en vous préparant à développer des applications web modernes et interactives avec confiance et efficacité.

2.1 Fondamentaux de React.js : composants, JSX et Virtual DOM

2.1.1 Composants

Les composants sont la pierre angulaire de React.js. Ils permettent de diviser l'interface utilisateur en segments indépendants et réutilisables, chaque segment gérant son propre état et sa propre logique. Un composant en React.js peut être une simple fonction ou une classe JavaScript.

Voici un exemple de composant fonctionnel basique :

```
function Welcome(props) {
  return <h1>Bonjour, {props.name}</h1>;
}
```

Et voici un exemple de composant basé sur une classe :

```
class Welcome extends React.Component {
  render() {
    return <h1>Bonjour, {this.props.name}</h1>;
  }
}
```

Les composants peuvent être imbriqués, permettant ainsi de construire des interfaces utilisateur complexes à partir de composants simples.

2.1.2 JSX

JSX (*JavaScript XML*) est une extension syntaxique de JavaScript, utilisée en React.js pour décrire ce à quoi l'interface utilisateur devrait ressembler. Avec JSX, il est possible d'écrire des balises HTML directement dans le code JavaScript, ce qui rend le code plus lisible et facile à comprendre.

Voici un exemple de JSX :

```
const element = <h1>Bonjour, monde</h1>;
```

JSX peut également intégrer des expressions JavaScript en les entourant d'accolades `{ }` :

```
const name = 'Sara';
const element = <h1>Bonjour, {name}</h1>;
```

Il est important de noter que l'utilisation de JSX n'est pas obligatoire dans les projets React. Cependant, il est largement adopté en raison de sa simplicité et de sa capacité à rendre le code plus lisible et expressif. En fonction des préférences ou des exigences du projet, vous pourriez travailler sur des fichiers avec l'extension .jsx ou .js en même temps.

2.1.3 Virtual DOM

Le Virtual DOM est l'une des innovations clés de React.js. Il s'agit d'une représentation légère du DOM réel qui permet à React.js de déterminer les changements nécessaires de manière efficace. Lorsque l'état d'un composant change, React.js crée un nouvel arbre de Virtual DOM, puis le compare avec l'arbre précédent. Cette comparaison permet de déterminer les modifications minimales nécessaires à appliquer au DOM réel pour mettre à jour l'interface utilisateur.

Cette approche offre plusieurs avantages :

- **Performance améliorée** en minimisant les opérations sur le DOM réel, qui sont coûteuses en termes de performance ;
- **Mises à jour prévisibles** : le processus de comparaison garantit que seules les parties de l'interface utilisateur qui ont changé sont mises à jour, rendant les mises à jour plus prévisibles et plus faciles à comprendre.