Anass El Ouahabi /
Mohammed Bouker

Rapport de formation : Front-end

I. Interface utilisateur et Expérience utilisateur UI/UX :

L'UI (Interface Utilisateur) et l'UX (Expérience Utilisateur) sont deux aspects fondamentaux du design.

L'UI s'attache aux éléments visuels et interactifs de l'interface utilisateur Cela inclut la disposition des écrans, les animations, les boutons, les icônes, et tout autre élément visuel.

Tandis que l'UX se concentre sur l'expérience globale vécue par l'utilisateur en utilisant un produit ou une plateforme, il a comme objectif d'améliorer la satisfaction des utilisateurs en améliorant l'usabilité, l'accessibilité, et l'interaction agréable avec l'application.

Les éléments clés dans Ui :

Les couleurs _ typographie _ mise en page (La disposition des éléments doit être logique et fluide) _ interactivité (comme les boutons et les liens doivent être clairement identifiables et réactifs)

Les éléments clés dans Ux :

Utilisateur (comprendre qui sont les utilisateurs possibles de plateforme) _ création des maquettes et wireframes (en utilisant figma).

Source : Ahmad Sekmani

II. Figma :

Figma est un outil de conception d'interface utilisateur (UI) et d'expérience utilisateur (UX) il permet aux concepteurs de créer des interfaces interactives des wireframe et des prototypes, tout en offrant des fonctionnalités robustes de collaboration en temps réel.

Wireframe : c'est un guide visuel basique qui représente la structure squelettique d'une page web ou d'une application, leur but principal est de définir l'agencement des éléments sur une page avant de détailler le design graphique et le contenu.

Source : Ahmad sekmani.

III. Javascript :

JavaScript est un langage de programmation dynamique qui est utilisé pour créer des interactions interactives sur les sites web. Il permet aux développeurs de mettre en œuvre des fonctionnalités complexes telles que la gestion d'événements, la manipulation de données et la communication avec les serveurs.

Les éléments clés que j'ai appris :

Les bases : syntaxe de base _ les objets _ les fonctions _ fichier JSON (leur rôle + syntaxe + passage de syntaxe de JSON à la syntaxe de javascript et vice-versa + utilisation de local Storage)

Document Object Model (DOM) :

Le DOM représente un document HTML comme une hiérarchie d'objets. Chaque élément du document, comme les balises, les attributs, le texte, et même les commentaires, est transformé en un objet distinct. Ces objets sont organisés en une structure arborescente qui reflète la structure du document.

JavaScript permet d'assigner des fonctions aux événements utilisateurs (comme des clics, des mouvements de souris, ou des frappes au clavier) pour manipulent le DOM en réponse à ces actions.

Source: SuperSimpleDev -JavaScript Full Course et mdn

IV. Vuejs

Vue.js est un Framework pour la construction d'interfaces utilisateur. Il utilise un système de réactivité qui rend le développement d'interfaces utilisateur interactives et plus facile. Lorsque les données changent, Vue met automatiquement à jour le DOM (Document Object Model) pour refléter ces changements.

Les éléments clés que j'ai appris :

Composants :

Vue.js est basé sur une architecture orientée composants. Chaque composant est une instance de Vue et peut être réutilisé dans différentes parties de l'application. Les composants permettent de structurer des applications complexes de manière modulaire et maintenable.

Il est toujours divisé en 3 parties : partie Template pour les balises html , parti script pour introduire la logique de composant et parti pour le style.

Directive :

Les directives sont des instructions spéciales que vous pouvez utiliser dans les Template HTML pour contrôler le comportement du DOM de manière déclarative.

Quelque directive utile :

v-model : Crée une liaison bidirectionnelle sur les éléments de formulaire pour simplifier la gestion des mises à jour des données. Par exemple, v-model="Text" synchronise le champ de formulaire avec la variable Text déclarer dans la partie script.

v-for : Utilisée pour rendre une liste d'éléments en se basant sur un tableau de données.

v-on : Attache un événement écouteur à un élément, ce qui permet d'exécuter du code JavaScript lorsque l'événement spécifié se produit. Par exemple, v-on:click="évènement" exécute la méthode évènement lorsque l'utilisateur clique sur l'élément.

v-if, v-else-if, v-else : Directives conditionnelles pour inclure ou exclure des blocs de Template HTML.

v-bind : Utilisée pour lier dynamiquement un ou plusieurs attributs de composant à une expression.

API Options :

Dans Vue.js, il existe deux approches principales pour créer des composants : l'API Options et l'API Composition. Chacune offre des moyens différents de structurer le code dans vos composants Vue. Dans ma formation J'ai intéressé juste par l'API options parce qu'il est plus accessible pour les débutants et elle suit une structure déclarative claire.

L'API Options est la méthode traditionnelle de création de composants dans Vue.js. Elle organise la logique du composant à travers des options spécifiques telles que data, Methods, Watch, props, et

Les hooks de cycle de vie comme mounted, created, etc. Chaque option représente un aspect distinct de la logique du composant.

Quelques options de l'Api options :

Data :il retourne un objet contenant les données locales du composant. Chaque propriété définie dans data est réactive, ce qui signifie que Vue suivra ses changements et mettra à jour la vue en conséquence.

Props : ils permettent de passer des données d'un composant parent à un composant enfant.

Méthodes : ils sont des fonctions associées à l'instance du composant. Elles peuvent être appelées depuis le Template ou liées à des événements.

Watchers : ils permettent de réagir à des changements spécifiques sur des données réactives. Un watcher peut exécuter une logique chaque fois qu'une donnée surveillée change.

Lifecycle hooks : Vue fournit plusieurs hooks de cycle de vie que vous pouvez utiliser pour exécuter du code à des moments spécifiques dans la vie du composant, tel que :

beforeCreate : Avant la création de l'instance du composant.

created : Après la création de l'instance.

beforeMount : Avant que l'instance ne soit montée.

mounted : Après que l'instance a été montée.

beforeUpdate : Avant la mise à jour du DOM.

updated : Après la mise à jour du DOM.

Router :

Vue Router est le système de routage pour Vue.js. Il permet de créer des applications web à page unique (SPA) en permettant le routage côté client, ce qui est crucial pour une navigation fluide sans rechargement de page.

SPA :

Une SPA est une application web qui interagit avec l'utilisateur en modifiant dynamiquement le contenu actuel de la page web avec les nouvelles données du serveur web, au lieu de charger de nouvelles pages entièrement. Cela permet une expérience utilisateur plus fluide et rapide, car les

ressources sont chargées une seule fois au début et seules les données nécessaires sont échangées avec le serveur.

Axios :

Axios est une bibliothèque populaire de JavaScript utilisée pour effectuer des requêtes HTTP depuis le navigateur ou depuis un environnement Node.js. Elle est particulièrement appréciée de l'utiliser à cause de sa simplicité d'utilisation, sa configuration facile, et son interface API prometteuse qui s'intègre parfaitement avec Vue.js.

Json-server :

C'est un outil qui permet de créer rapidement une fausse API RESTful. Il utilise un fichier JSON simple comme base de données pour stocker les données, et fournit automatiquement les routes et opérations CRUD (Create, Read, Update, Delete) sans que vous ayez besoin d'écrire le backend vous-même.

Il est extrêmement utile dans les phases initiales du développement lorsque le backend réel n'est pas encore disponible. Il aide Les développeurs front-end pour commencer à travailler sur la logique d'interaction avec l'API (comme l'affichage des données, l'envoi de formulaires, etc.) sans attendre que le backend soit prêt.

Vuex :

Vuex est une bibliothèque de gestion d'état centralisée pour les applications Vue.js. Elle sert de dépôt centralisé pour tous les composants dans une application, avec des règles assurant que l'état ne peut être muté que de manière prévisible. Vu sa conception, Vuex est particulièrement utile dans les applications SPA (Single Page Application) complexes, où le partage et la gestion de l'état entre plusieurs composants peuvent devenir problématiques sans une source de vérité centralisée.

Ses caractéristiques clés :

State : L'état global de l'application qui est stocké dans un seul objet.

Getters : Fonctions permettant d'accéder à des portions de l'état.

Mutations : Seules les mutations ont le pouvoir de modifier l'état. Elles sont synchrones et enregistrent explicitement ce qui change l'état.

Actions : Les actions commettent des mutations et peuvent contenir des opérations asynchrones.

Modules : Vuex permet de diviser le store en modules pour mieux gérer des applications complexes.

Source : Vuejs.org , Mohamed ghailani et Code Step By Step .

V. Autre

Tailwind css : Tailwind CSS est un Framework CSS utilitaire qui permet aux développeurs de construire rapidement des interfaces utilisateur en appliquant des classes de style atomiques directement dans le HTML. Chaque classe est conçue pour servir un seul objectif de style, offrant une flexibilité maximale et un contrôle précis du design sans écrire de CSS supplémentaire.

Css Avancé : SuperSimpleDev .