GitHub

Nous allons utiliser l’intégration de Git sur VS Code pour gérer notre projet avec Git.

Routage

Fait référence à la gestion de la navigation entre différentes pages ou vues d'une application web du côté client,

nous allons utiliser un système de routage basique codé en JavaScript. Il est composé de 3 fichiers :

1. Le fichier **Route.js** définit une classe Route qui représente une route de l'application.

export default class Route {

constructor(url, title, pathHtml, pathJS = "") {

this.url = url;

this.title = title;

this.pathHtml = pathHtml;

this.pathJS = pathJS;

}

}

1. Le fichier **allRoutes.js** crée un tableau « *allRoutes* » contenant toutes les routes de l'application. Chaque route est créée en utilisant la classe « *Route* » avec les paramètres appropriés. Il définit également la variable « *websiteName* », qui représente le nom du site web.

import Route from "./Route.js";

//Définir ici vos routes

export const allRoutes = [

new Route("/", "Accueil", "/pages/home.html"),];

//Le titre s'affiche comme ceci : Route.titre - websitename

export const websiteName = "Quai Antique";

1. Le fichier **router.js**  C’est lui qui contient la logique de routage.

import Route from "./Route.js";

import { allRoutes, websiteName } from "./allRoutes.js";

// Création d'une route pour la page 404 (page introuvable)

const route404 = new Route("404", "Page introuvable", "/pages/404.html");

// Fonction pour récupérer la route correspondant à une URL donnée

const getRouteByUrl = (url) => {

let currentRoute = null;

// Parcours de toutes les routes pour trouver la correspondance

allRoutes.forEach((element) => {

if (element.url == url) {

currentRoute = element;

}

});

// Si aucune correspondance n'est trouvée, on retourne la route 404

if (currentRoute != null) {

return currentRoute;

} else {

return route404;

}

};

// Fonction pour charger le contenu de la page

const LoadContentPage = async () => {

const path = window.location.pathname;

// Récupération de l'URL actuelle

const actualRoute = getRouteByUrl(path);

// Récupération du contenu HTML de la route

const html = await fetch(actualRoute.pathHtml).then((data) => data.text());

// Ajout du contenu HTML à l'élément avec l'ID "main-page"

document.getElementById("main-page").innerHTML = html;

// Ajout du contenu JavaScript

if (actualRoute.pathJS != "") {

// Création d'une balise script

var scriptTag = document.createElement("script");

scriptTag.setAttribute("type", "text/javascript");

scriptTag.setAttribute("src", actualRoute.pathJS);

// Ajout de la balise script au corps du document

document.querySelector("body").appendChild(scriptTag);

}

// Changement du titre de la page

document.title = actualRoute.title + " - " + websiteName;

};

// Fonction pour gérer les événements de routage (clic sur les liens)

const routeEvent = (event) => {

event = event || window.event;

event.preventDefault();

// Mise à jour de l'URL dans l'historique du navigateur

window.history.pushState({}, "", event.target.href);

// Chargement du contenu de la nouvelle page

LoadContentPage();

};

// Gestion de l'événement de retour en arrière dans l'historique du navigateur

window.onpopstate = LoadContentPage;

// Assignation de la fonction routeEvent à la propriété route de la fenêtre

window.route = routeEvent;

// Chargement du contenu de la page au chargement initial

LoadContentPage();

J’ai crée le fichier index.html

Dans body j’ai ajoutée la balise header et main avec un « id=main-page» et la balise footer

Après footer j’ai inséré ce script

<script type="module" src="./Router/router.js"></script>

Ensuite j’ai crére le dossier Router comportant les 3 fichiers js

1. Le fichier **Route.js** définit une classe Route qui représente une route de l'application.
2. Le fichier **allRoutes.js** crée un tableau « *allRoutes* » contenant toutes les routes de l'application. Chaque route est créée en utilisant la classe « *Route* » avec les paramètres appropriés. Il définit également la variable « *websiteName* », qui représente le nom du site web.
3. Le fichier **router.js**  C’est lui qui contient la logique de routage.

Vous trouvez en annexe ecrit en orange le code associée aux fichiers ci-dessus

A la fin j’ai crée le dossier pages et dans ce dossier, le fichier home.html

On a fini le système de routage et maintenant c’est le moment de valider les modifications sur github

Je dois cree un souvegarde à l’aide de git on va réaliser un commit pour sauvegarder **l’ajout du système de routage sur GitHub**

CSS BOOTSTRAP

Nous allons ici décrire les étapes pas à pas pour installer Bootstrap sur un projet.

Avant de commencer, assurez-vous d'avoir Node.js installé sur votre système. Node.js inclut npm par défaut, ce qui facilitera l'installation de Bootstrap

Vous pouvez vérifier si Node.js est installé en ouvrant votre terminal () et en tapant la commande node -v.

Vous devez maintenant ouvrir un terminal de commande à la racine de votre projet. Vous pouvez installer Bootstrap en utilisant la commande npm install bootstrap.

COMPLEMENT

Choix de méthode

Pas facile de choisir la méthode pour intégrer Bootstrap. Pour vous aider, voici quelques points positifs et négatifs de chaque méthode. En fonction de votre projet, à vous de choisir avec ces arguments en tête :

* Installation manuelle en téléchargeant les fichiers Bootstrap :
  + Points forts : contrôle total sur les fichiers utilisés, possibilité de personnaliser les styles et de les intégrer facilement dans votre projet.
  + Points faibles : requiert une gestion manuelle des mises à jour et des dépendances.
* Utilisation d'un gestionnaire de paquets (comme npm) :
  + Points forts : gestion des dépendances simplifiée, facilité des mises à jour grâce à la gestion automatisée des packages.
  + Points faibles : peut nécessiter une configuration initiale plus complexe, surtout si vous n'utilisez pas déjà un gestionnaire de paquets.
* Utilisation d'un CDN (Content Delivery Network) :
  + Points forts : facilité d'utilisation, aucun téléchargement ou intégration de fichiers requis, mise en cache optimisée pour une meilleure performance.
  + Points faibles : dépendance externe, nécessite une connexion internet pour charger les fichiers Bootstrap, moins de contrôle sur les versions utilisées et les mises à jour.
  + J’ai crée le dossier node\_modules et dedans ajouter le dossier bootstrap

Une fois l'installation terminée, vous pouvez inclure Bootstrap dans votre projet en important les fichiers CSS et JS. Il suffit d’inclure les fichiers JS et CSS de Bootstrap, qui sont désormais présents dans le dossier « *node\_module* ».

* + On ajoute dans la balise head <link rel="stylesheet" href="node\_modules/bootstrap/dist/css/bootstrap.min.css">
  + Et dans body à la fin <script src="node\_modules/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

On a fini l’installation de bootstrap et maintenant c’est le moment de valider les modifications sur github

Je dois cree un souvegarde à l’aide de git on va réaliser un commit pour sauvegarder **l’ajout install bootstrap**

**La customisation de Bootstrap avec Sass**

Pour personnaliser notre projet nous allons modifier des éléments sur Bootstrap grâce à Sass, nous avons la possibilité de modifier les couleurs par défaut et font styles.

Pour personnaliser les couleurs de Bootstrap avec Sass, il est recommandé de créer un fichier « *\_custom.scss*

Vous devez avoir un fichier « *main.scss* », qui sera le fichier principal de votre site. Ce fichier main importera ce fichier custom, qui lui importe Bootstrap.

Nous pouvons aussi choisir d’installer et d’importer Bootstrap icons pour pouvoir afficher de jolies icônes

@import url("/node\_modules/bootstrap-icons/font/bootstrap-icons.css");

@import 'custom';