Documentation du code :

Version 1

Chef de qualité du code :

I. Au niveau du code :

1. documentation d'un component ReactJS :

Exemple :

```
/**
 * @component Header
 * @description Affiche l'en-tête de l'application
 *
 * @props {string} title - Le titre de l'application
 * @props {boolean} showMenu - Indique si le menu doit être affiché ou non
 *
 * @example
 * <Header title="Mon application" showMenu={true} />
 */
import React from 'react';
import PropTypes from 'prop-types';
import Menu from './Menu';

const Header = ({ title, showMenu }) => {
  return (
    <header>
      <h1>{title}</h1>
      {showMenu && <Menu />}
    </header>
  );
};

Header.propTypes = {
  title: PropTypes.string.isRequired,
  showMenu: PropTypes.bool.isRequired,
};
```

```
export default Header;
```

Dans cet exemple, nous avons documenté un component ***Header*** qui affiche l'en-tête de l'application. Nous avons utilisé des commentaires pour expliquer le rôle du composant, ses propriétés et son utilisation. Voici ce que chaque partie de la documentation signifie :

@component Header : Cette annotation indique que la documentation décrit le composant Header.

@description Affiche l'en-tête de l'application : Cette annotation décrit brièvement ce que fait le composant.

@props {string} title - Le titre de l'application : Cette annotation décrit la propriété title, qui est une chaîne de caractères.

@props {boolean} showMenu - Indique si le menu doit être affiché ou non : Cette annotation décrit la propriété showMenu, qui est un booléen.

@example : Cette annotation montre un exemple d'utilisation du composant avec ses propriétés.

Nous avons également utilisé la méthode PropTypes pour valider les propriétés du composant, ce qui aide à éviter les erreurs de type. Enfin, nous avons exporté le composant Header pour qu'il puisse être utilisé dans d'autres parties de l'application.

En suivant cette méthode de documentation, vous pouvez créer une documentation claire et précise pour chaque composant de votre application ReactJS. Cela facilitera la compréhension et l'utilisation de vos composants par les autres développeurs travaillant sur le projet.

II. Documentation PDF :

Voici quelques étapes pour documenter les composants dans ReactJS :

Nommez vos composants de manière significative : Utilisez des noms qui reflètent la fonctionnalité ou la responsabilité du composant.

Utilisez des commentaires : Ajoutez des commentaires à votre code pour expliquer ce que fait chaque partie du composant.

Décrivez les propriétés : Pour chaque propriété que vous utilisez dans votre composant, expliquez sa signification, son type et sa valeur par défaut, le cas échéant.

Incluez des exemples : Fournissez des exemples de code pour montrer comment utiliser le composant et ses propriétés.

Expliquez les méthodes : Si votre composant utilise des méthodes, expliquez leur fonctionnement et leur utilisation.

Fournissez des informations sur l'état : Si votre composant utilise l'état, expliquez comment il est mis à jour et comment cela affecte le comportement du composant.

Fournissez des informations sur les événements : Si votre composant gère des événements, expliquez comment ils sont déclenchés et comment ils affectent le comportement du composant.

Écrivez une documentation générale : Écrivez une documentation générale pour votre composant, qui explique son fonctionnement global, sa responsabilité et son utilisation.

Exemple :

Documentation du code pour le component "X":

1- **Introduction**
2- **Structure du code :**
c-a-d les fichiers ou les dossier que vous créer, modifier, supprimer,..

Exemple :

« index.js » : le point d'entrée de l'application, qui crée une instance de l'application et la rend dans le DOM.

« App.js : » le composant principal de l'application, qui contient la logique et la structure générale de l'application.

« components/ : » un répertoire contenant les composants réutilisables de l'application, tels que des boutons, des formulaires, des listes, etc.

« pages/ : » un répertoire contenant les pages spécifiques de l'application, telles que la page d'accueil, la page de connexion, la page de création de tâches, etc.

« utils/ : » un répertoire contenant les fonctions utilitaires de l'application, telles que les fonctions de validation, de formatage de dates, etc.

3- Fonctionnalités de l'application « Component » OU « Components »:

Exemple :

L'application permet à l'utilisateur de :

- **créer, modifier et supprimer des tâches**
- **…**

4- Utilisation de l'API : (Facultatif c'est vous utiliser des API)

Exemple :

L'application utilise une API REST pour stocker les données des tâches. Les appels à l'API sont effectués à l'aide du module Axios. Les endpoints de l'API sont définis dans le fichier api.js.

5- Installation et utilisation :

Les choses que vous installer dans l'environnement de travail

6- Conclusion :

Remarque :

C’est vous avez d’autre champ pour avoir une bonne Doc n’hésiter pas de l’ajouter.