Projet Formation React

1. Découpage de l’application en composant
2. Créer un dossier images dans public
   * Ajouter logo192.png et logo512.png seront accessible directement par le navigateur, et ne sont pas soumis par le moteur de rendu React
3. Mettre en commentaire le contenu qui est à l’intérieur du <div className="App"> => App.js
4. Modifier index.css

* Créer un dossier assets/images dans src
* Ajouter pattern.webp
* Remarques : Ces ressources seront soumises au moteur de rendu React tandis que les dossiers qui sont dans publics ne sont pas soumis au moteur de rendu react, car ils seront accessibles à partir du navigateur

1. Créer le composant « **Presentation.js** » dans le dossier pages
2. Créer le composant Routage.js
   * Dans ce composant je vais indiquer comment installer la librairie externe React Router DOM
3. Dans App.js, créer la route pour le composant Presentation « / » et la route pour le composant Routage « /routing »
4. Créer composant Navigation.js dans le dossier components pour pouvoir naviguer entre les routes :
   * Créer le lien <NavLink to= />Presentation</ NavLink>
   * Equivalent à <a class href=/>Presentation</a>
5. Dans Navigation.js on va créer un lien avec NavLink pour la route /routing
6. Ajouter le composant Navigation.js dans le composant App.js
7. Modifier App.css (se trouve dans le dossier style)
   * Ajouter la class .App
   * Ajouter la classe .monspan défini dans le composant Présentation
8. Créer le fichier CSS Navigation.css dans le dossier
9. Créer Logo.js dans le dossier components et le fichier Logo.css
10. Créer un composant static.js dans le dossier pages
    * Dans App.js créer la route
    * Dans Navigation.js créer le lien
11. Dans App.css ajouter les classes .grid et .card (qui vient du composant Static)
12. Création du composant State.js dans le dossier pages
    * Dans App.js crée la route /hooks
    * Dans Navigation.js crée le lien correspondant à cette route
      + Un exemple sans utiliser le hook d’etat
        - Tester les méthodes boom, event, param, incrementCount
        - Un autre exemple en ajoutant le hook d’état (useState)
          1. Déclarer une variable d’état qu’on nommera count

Ajouter la méthode increment et le tester

Ajouter la méthode handleSubmit et le tester

* + Dans App.css ajouter le sélecteur css correspondant à la balise button

1. Création du composant Props.js dans le dossier pages
   * Dans App.js on va créer la route /props
   * Dans Navigation.js on crée le lien pour cette route
   * Créer le composant Product.js dans le dossier components
     + Il contient
       - Une fonction Product
       - Une fonction ProductObj
       - Une fonction ProductTyped

* Dans le composant Props.js
  + Ajouter la méthode Props
  + Dans le JSX tester la méthode Product et ProductObj
  + Utilisation du typage
    - npm i prop-types
    - Dans le composant Product ajouter « ProductTyped.propTypes »
    - Dans JSX tester la méthode ProductTyped
  + Ajouter le hook d’état (useState)
    - Déclare une « variable d’état » qu’on nommera isLoading
    - Tester dans le JSX le useState
      * Créer un composant Bouton nommé **Buttons.js** dans le dossier components. Ce composant Bouton sera appelé par le composant Props.js en lui passant des paramètres (props)
  + Dans le composant Props.js créer la méthode testBtn
  + Créer **Boutton.css** dans le dossier style et l’importer dans Buttons.js
  + Ajouter la fonction **BackBtn** dans le composant buttons.js

1. Créer un composant **Connexion.js** dans le dossier pages
   * Dans App.js créer la route /login et le lien dans Navigation.js
   * Ajouter le hook d’état (useState) 🡺 [isLogged updateLog] = useState(false)
     + Déclarer une variable d’état qu’on nommera isLogged
   * Créer une fonction login en lui passant la méthode updateLog qui met à jour isLogged
   * Tester dans JSX
2. Créer un composant **Effect.js** dans le dossier pages
   * Dans App.js créer une route /effect et le lien dans Navigation.js
   * Créer deux hooks (variable d’état « counter » et « isLogged »)
   * Créer la méthode « increment » qui va incrementer la variable « counter » grâce à sa paire de méthode setCounter