POLYTECH DIJON

TRAVAUX DIRIGÉS

Conception Web Avancée

Introduction à REACT

OBJECTIFS

- Prise en main de ViteJS
- Création de composants React
- Utilisation de TypeScript

I. VITEJS

I.A. Nouveau projet

- Démarrez Visual Studio Code
- Ouvrez un dossier pour le projet du TD
- Ouvrez un terminal dans VS Code
- Exécutez la commande ci-dessous et suivez les instructions pour créer un projet utilisant React avec TypeScript :

```
npm create vite
```

- Suivez les dernières instructions fournies par ViteJS pour démarrer le serveur de développement.
- Affichez la page Web d'exemple dans votre navigateur.

I.B. Remise à 0

- Dans le dossier **src**, supprimez tous les fichiers en dehors de **main.tsx**
- Modifiez le fichier **main.tsx** pour qu'il contienne le code suivant :

```
import React from 'react'
import ReactDOM from 'react-dom/client'

ReactDOM.createRoot(document.getElementById('root')!).render(
    <React.StrictMode>
       //Content of the page
    </React.StrictMode>,
)
```

- Enregistrez les fichiers ouverts et vérifiez que votre navigateur affiche une page blanche.

Parfait, nous allons pouvoir commencer !

II. REACT

II.A. The bouton

II.A.1 Composant

- Dans le dossier src de votre projet, créez un dossier **`the-button`**.
- Ajoutez à ce dossier un fichier **`the-button.tsx`**.
- A partir de la syntaxe suivante, créer un composant React qui affiche un simple bouton intitulé **`OK`**

```
export function ComponentName()
{
    return (
        // JSX code of the component
    )
}
```

- Modifiez le fichier **`main.tsx`** pour ajour une balise **`<TheButton />`** dans le contenu de la page.
- N'oubliez pas de mettre à jour les imports si nécessaire.
- Vérifiez dans votre navigateur que votre bouton s'affiche bien.

II.A.2. Style

- Ajoutez un fichier **`the-button.css`** dans le dossier **`the-button`**.
- Appliquez le CSS nécessaire pour obtenir le résultat suivant :

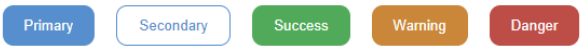
II.A.3. Propriétés

Ce bouton est sympa, mais il affiche toujours le texte **`OK`** et apparaît toujours en bleu. Ajoutons un peu de personnalisation.

- Dans le fichier **`the-button.tsx`**, créez un type **`TheButtonProperties`** qui représentera un objet ayant un attribut **`label`** de type **`string`**.
- Modifiez le composant **`TheButton`** pour que l'attribut **`label`** définisse le texte à afficher.
- Modifiez le fichier **`main.tsx`** pour intégrer cette modification.
- Testez le bon fonctionnement.

II.A.4. Déclinaisons

- Modifiez votre composant afin qu'il puisse être personnalisé dans les déclinaisons suivantes :



> **Astuce**
> En TypeScript, vous pouvez faire des énumérations :
>
> ```
> enum MyEnum {
>     LABEL_1 = "label_1",
>     LABEL_2 = "label_2",
> }
> ```

- Essayez de créer plusieurs boutons dans des déclinaisons différentes.

II.A.5. Evénements

Personnaliser le style du bouton, c'est bien. Pouvoir définir ce qu'il se passe lorsque l'on clique dessus, serait mieux.

- Modifiez votre composant afin que ce dernier ait une propriété onClick.

> **Astuce**
> En TypeScript, il est possible d'indiquer un type fonction :
>
> ```
> type MyType = {
>     myFunction : () => void ;
> }
> ```
>
> Ici, **`myFunction`** contiendra la référence sur une fonction qui ne prend pas de paramètre et qui ne retourne rien (**`void`**).

> **Astuce**
> React fournit un type pour les événements de la souris : **`MouseEvent`**.
>
> *Attention à bien importer le type* **`MouseEvent`** *de* **`React`**

- Testez le bon fonctionnement en affichant un message dans la console lorsque l'on clique sur un bouton.

II.B The Switch

- Réalisez un composant de type switch qui permette de mémoriser une valeur booléenne.

  Voici à quoi devra ressembler votre composant (off – on) :

- Testez le bon fonctionnement
- Ajoutez les éléments nécessaires pour :
  - Définir l'état initial du Switch depuis le composant parent
  - Récupérer l'état Switch lorsque celui-ci change

II.C Combo

- Créez un formulaire permettant d'afficher ou non chaque type de bouton :