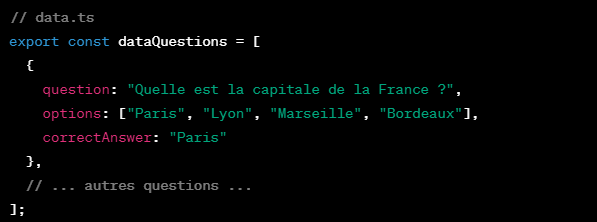
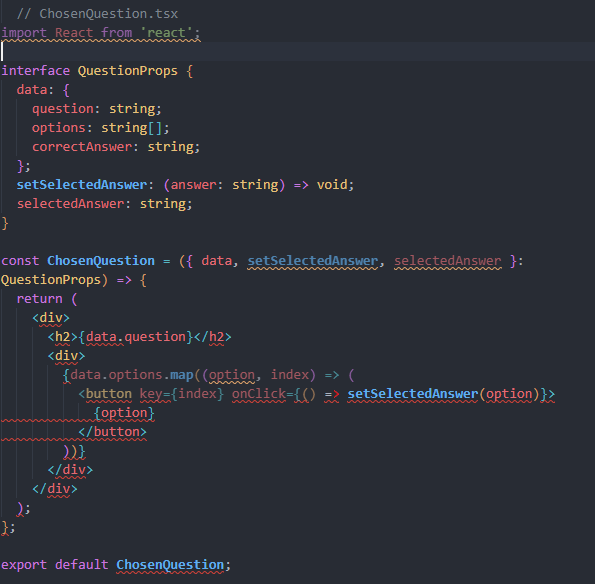
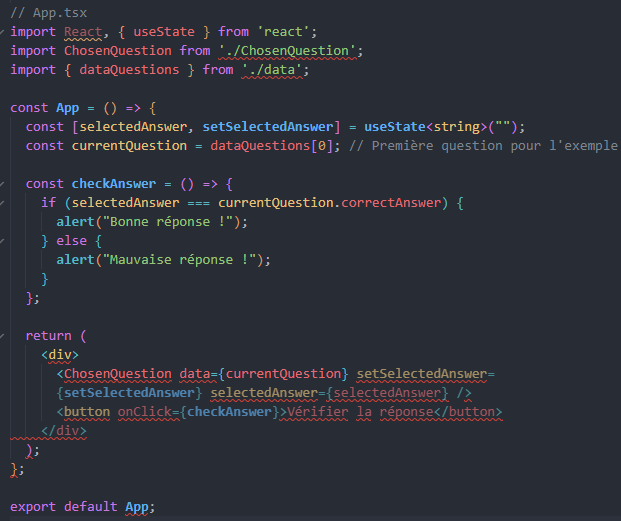
1. **Déclaration de selectedAnswer dans App.jsx** :
   * Vous utilisez **useState** pour créer un état local **selectedAnswer** et une fonction pour le mettre à jour **setSelectedAnswer**.
   * **useState<string>("")** indique que **selectedAnswer** est de type **string** et son état initial est une chaîne de caractères vide.
   * L'utilisation de **useState** est appropriée car **selectedAnswer** changera en fonction des actions de l'utilisateur (sélection d'une réponse).
2. **Passage de setSelectedAnswer à Question.tsx** :
   * Vous passez **setSelectedAnswer** au composant **Question** comme une prop. Cela permet au composant **Question** de mettre à jour l'état **selectedAnswer** dans **App.jsx**.
3. **Mise à Jour de selectedAnswer dans Question.tsx** :
   * À l'intérieur de **Question.tsx**, pour chaque option de réponse, vous avez un **div** avec un événement **onClick** attaché.
   * **onClick={() => setSelectedAnswer(option)}** est la partie cruciale. Quand l'utilisateur clique sur une option, cette ligne de code est exécutée.
   * Cela appelle **setSelectedAnswer** avec la valeur de l'option sélectionnée (**option**), ce qui met à jour l'état **selectedAnswer** dans **App.jsx**.
4. **Répercussion de la Sélection** :
   * Le composant **Question** utilise la valeur de **selectedAnswer** pour afficher visuellement quelle option a été sélectionnée (par exemple, en changeant la couleur du carré).
   * La mise à jour de **selectedAnswer** déclenche un re-render du composant **App** et de ses enfants, reflétant la nouvelle sélection.

En résumé, **selectedAnswer** dans **App.jsx** garde trace de la réponse sélectionnée par l'utilisateur, et **setSelectedAnswer** est utilisé pour mettre à jour cet état lorsque l'utilisateur clique sur une option dans le composant **Question**. C'est un excellent exemple de la façon dont React permet de gérer l'état et les interactions utilisateur dans une application web.

Explication exemple simplifié :  
  






**Explications des Grandes Lignes du Code**

Déclaration des Fonctions

* Dans **App.tsx**, vous utilisez **useState** pour créer un état local **selectedAnswer** et une fonction pour le mettre à jour **setSelectedAnswer**. On aurait pu mettre le nom que l’on veut
* **checkAnswer** dans **App.tsx** est une fonction qui compare la réponse sélectionnée (**selectedAnswer**) avec la réponse correcte de la question actuelle.

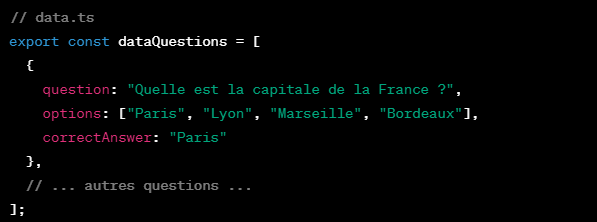
Onclick pour Sélectionner une Réponse

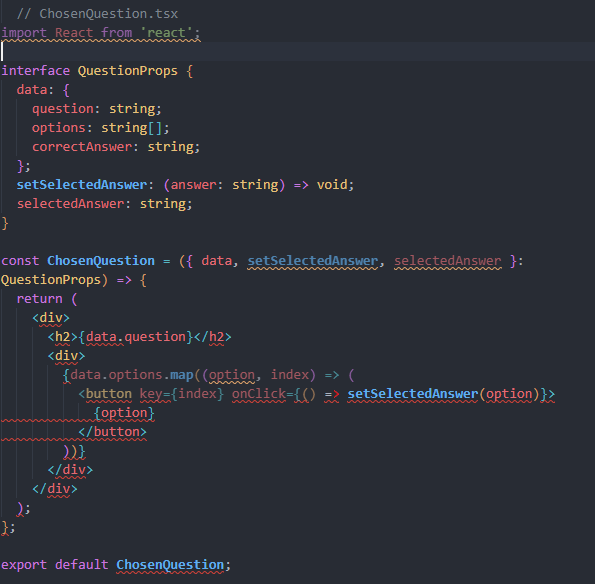
* Dans **ChosenQuestion.tsx**, chaque bouton d'option a un gestionnaire d'événements **onClick** qui appelle **setSelectedAnswer(option)** lorsqu'il est cliqué. Cela met à jour **selectedAnswer** dans **App** avec la réponse choisie.

Bouton pour Vérifier la Réponse

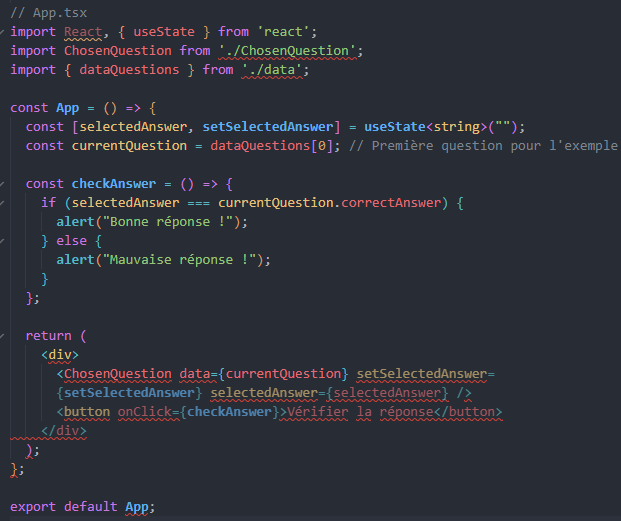
* Le bouton dans **App.tsx** (**<button onClick={checkAnswer}>**) appelle la fonction **checkAnswer** lorsqu'il est cliqué pour vérifier si la réponse sélectionnée est correcte.

Explications de **.map**, **data.questions**, etc.

* **.map()** est utilisé pour itérer sur chaque **option** dans **data.options** et les transformer en éléments JSX (boutons). Cela est nécessaire pour afficher une liste d'éléments basée sur les données d'un tableau.
* **data.question** est une chaîne de caractères simple, donc elle peut être insérée directement dans le JSX avec des accolades **{}**.
* **data.options** est un tableau, donc **.map()** est utilisé pour afficher chaque option de manière dynamique.  
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
    
  
* Dans **data.ts**, vous définissez un tableau de questions. Chaque question est un objet avec une **question**, des **options**, et une **correctAnswer**.



Dans **ChosenQuestion.tsx**, le **<h2>{data.question}</h2>** affiche la question actuelle. Par exemple, pour la première question du tableau **dataQuestions**, cela affichera "Quelle est la capitale de la France ?".



Dans **App.tsx**, **currentQuestion** est défini comme la première question du tableau **dataQuestions**. Le composant **ChosenQuestion** est rendu avec cette question, et l'utilisateur peut sélectionner une réponse et vérifier si elle est correcte.

Fonctionnement des Boutons

* Les boutons dans **ChosenQuestion.tsx** permettent à l'utilisateur de choisir une réponse. Lorsqu'un bouton est cliqué, **setSelectedAnswer** est appelé avec la réponse choisie.
* Le bouton dans **App.tsx** appelle **checkAnswer** pour vérifier si la réponse sélectionnée (**selectedAnswer**) est la même que la **correctAnswer** de la question actuelle.

Utilisation de **.map()**

* **.map()** est utilisé pour itérer sur chaque option dans data.options et les transformer en éléments JSX (boutons). Cela est nécessaire pour afficher une liste d'éléments basée sur les données d'un tableau.
* **data.question** est une chaîne de caractères simple, donc elle peut être insérée directement dans le JSX avec des accolades {}.
* **data.options** est un tableau, donc **.map()** est utilisé pour afficher chaque option de manière dynamique.