Programmation Fonctionnelle : des concepts aux applications web (NFP119)

Données immutables

Télécharger les données suivantes et créér un fichier TypeScript contenant la déclaration d'une constante `users` initialisée avec ces données :

https://jsonplaceholder.typicode.com/users

Définir les types `Geo`, `Address`, `Company` et `User` (dont toutes les propriétés seront `readonly`) qui permettent de typer la déclaration de la constante ainsi `users: readonly User[]`.

Utiliser ensuite les méthodes des tableaux pour implanter les diverses requêtes et transformations qui suivent (on testera bien sûr ces fonctions en utilisant la constante `users` donnée).

1. Ecrire une fonction qui renvoie le tableau de toutes les compagnies :

   ```
   function allCompanies(users: readonly User[]): readonly Company[]
   ```

2. Ecrire une fonction qui renvoie le tableau de tous les noms des compagnies :

   ```
   function allCompanyNames(users: readonly User[]): readonly string[]
   ```

3. Ecrire une fonction qui renvoie le tableau de toutes les adresses :

   ```
   function allAddresses(users: readonly User[]): readonly Address[]
   ```

4. Ecrire une fonction qui renvoie le tableau de toutes les villes :

   ```
   function allCities(users: readonly User[]): readonly string[]
   ```

5. Ecrire une fonction qui détermine si la ville d'adresse commence par `"S"` :

   ```
   function cityStartsWithS(address: Address): boolean
   ```

   On pourra utiliser la méthode `startWith` du type string.

6. Ecrire une fonction qui renvoie le tableau des adresses dont la ville commence par `"S"` :

   ```
   function addressesWithS(addresses: readonly Address[]): readonly Address[]
   ```

7. Ecrire une fonction qui détermine si le site web d'un utilisateur termine par `".org"` :

   ```
   function userWebsiteWithOrg(user: User): boolean
   ```

   On pourra utiliser la méthode `endsWith` du type string.

8. Ecrire une fonction qui renvoie le tableau des utilisateurs dont le site web termine par `".org"` :

   ```
   function usersWithOrg(users: readonly User[]): readonly User[]
   ```

9. Ecrire une fonction qui remplace les coordonnées géographiques des adresses par `(0, 0)` :

   ```
   function resetGeo(addresses: readonly Address[]): readonly Address[]
   ```

10. Ecrire une fonction qui remplace toutes les adresses mail des utilisateurs par `"someone@example.com"` :

    ```
    function resetEmail(users: readonly User[]): readonly User[]
    ```

11. Ecrire une fonction qui convertit le nom de la ville d'une adresse en lettres majuscules :

    ```
    function addressCityToUpperCase(address: Address): Address
    ```

    On pourra utiliser la méthode `toUpperCase` du type string.

12. Ecrire une fonction qui convertit les noms de ville des adresses en lettres majuscules :

    ```
    function addressesCityUC(addresses: readonly Address[]): readonly Address[]
    ```

13. Ecrire une fonction qui renvoie les utilisateurs dans l'ordre inverse :

    ```
    function usersReversed(users: readonly User[]): readonly User[]
    ```

    On pourra utiliser la méthode `reverse` des tableaux.

14. Ecrire une fonction qui compare deux `string` selon ">" (qui correspond à l'ordre alphabétique pour le type `string`) :

    ```
    function cmpString(s1: string, s2: string): (1 | 0 | -1)
    ```

    Cette fonction doit renvoyer la valeur `1` si `s1 > s2`, la valeur `-1` si `s2 > s1`, et la valeur `0` dans les autres cas (`s1` et `s2` sont égales).

15. En déduire une fonction qui compare deux adresses en comparant uniquement les villes (selon l'ordre alphabétique) :

    ```
    function cmpCities(address1: Address, address2: Address): (1 | 0 | -1)
    ```

16. Ecrire une fonction qui trie les adresses selon l'ordre alphabétique des villes :

    ```
    function addressesSorted(addresses: readonly Address[]): readonly Address[]
    ```

    On pourra utiliser la méthode `sort` des tableaux.

17. Ecrire une fonction qui compare deux `number` selon ">" (qui correspond à l'ordre naturel pour `number`) :

    ```
    function cmpNumber(n1: number, n2: number): (1 | 0 | -1)
    ```

    C'est exactement la même fonction que `cmpString`, seuls les types sont différents.

18. En déduire une fonction qui compare deux utilisateurs en comparant uniquement les codes postaux de leurs adresses :

    ```
    function cmpZipcode(user1: User, user2: User): (1 | 0 | -1)
    ```

19. Ecrire une fonction qui trie les utilisateurs selon l'ordre croissant du code postal :

    ```
    function usersSorted(users: readonly User[]): readonly User[]
    ```

20. Ecrire une fonction qui renvoie la latitude maximale des adresses :

    ```
    function maxLatitude(addresses: readonly Address[]): number
    ```

    On pourra utiliser la fonction `parseFloat` pour convertir une `string` en `float`, la fonction `Math.max`, et la méthode `reduce` des tableaux.