Fondations, Setup et Typage Strict

Objectif : Configurer un environnement de développement professionnel et maîtriser la syntaxe de base de TypeScript.

1. Installation et Setup (15 min)

Avant de coder, nous devons préparer le terrain. TypeScript a besoin d'un compilateur pour transformer votre code en JavaScript compréhensible par Node.js ou les navigateurs.

A. Initialisation du projet

Ouvrez votre terminal dans un nouveau dossier tp1-typescript :

```
# Création du fichier package.json
npm init -y

# Installation de TypeScript et des outils de développement
# -D signifie "devDependencies" (uniquement pour le développement)
npm install -D typescript ts-node @types/node
```

B. Configuration de TypeScript

Générez le fichier de configuration tsconfig.json :

```
npx tsc --init
```

Configuration recommandée : Ouvrez tsconfig.json et vérifiez que ces options sont bien activées (décommentées) :

- "target": "es2020" (pour utiliser les fonctionnalités modernes de JS)
- "module": "commonjs"
- "strict": true (L'option la plus importante : active le typage fort)
- "esModuleInterop": true

C. Scripts de lancement

Dans votre package.json, ajoutez ce script pour simplifier l'exécution :

```
"scripts": {
  "start": "ts-node src/index.ts"
}
```

2. Exercice 1 : Les Primitifs et l'Inférence

Créez un dossier src et un fichier index.ts à l'intérieur.

1. Déclarez trois variables avec typage explicite : nom (string), age (number) et estAdherent (boolean).
2. **Le test :** Essayez de réassigner age avec une chaîne de caractères (ex: "25"). Observez l'erreur soulignée en rouge par VS Code.
3. **L'Inférence :** Créez une variable score = 10. Ne précisez pas le type. Survolez-la avec votre souris. Quel type TypeScript a-t-il deviné ?

3. Exercice 2 : Tableaux et Tuples

1. **Tableau simple :** Créez un tableau sports qui n'accepte que des string.
2. **Tableau d'objets :** Créez un tableau panier qui contient des objets : { produit: string, prix: number }.
3. **Le Tuple :** Créez une variable reponseAPI qui doit contenir exactement deux éléments : un code de statut (ex: 200) et un message (ex: "OK").
    - *Syntaxe : let reponse: [number, string] = [200, "OK"];*

4. Exercice 3 : Les Fonctions

1. **Typage de retour :** Créez une fonction calculerRemise qui prend un prix (number) et retourne le prix avec 20% de remise (number).
2. **Paramètre optionnel :** Créez une fonction saluer qui prend un prenom et un titre optionnel (ex: "M.", "Mme"). Si le titre est absent, affichez juste "Bonjour [prenom]".
3. **Union Types :** Créez une fonction formaterID qui accepte un argument pouvant être soit une string soit un number.

5. Challenge : Modélisation (Préparation Projet Final)

Ici, on commence à préparer la "Ligue Sportive" de vendredi.

Définissez une **Interface** nommée IEquipement avec :

- id (number)
- nom (string)
- categorie (string)
- disponible (boolean)

Créez une fonction afficherInventaire qui prend un tableau de IEquipement et affiche en console uniquement les noms des équipements qui sont disponible: true.

Pour tester votre code : npm start