Introduction à TypeScript : Pourquoi s'embêter avec des types ?

1. C'est quoi au juste ?

TypeScript n'est pas un nouveau langage qui remplace le JavaScript. C'est un **"Superset"** (sur-ensemble) de JavaScript.

- **La règle d'or :** Tout code JavaScript est du code TypeScript valide.
- **L'identité :** C'est du JS auquel on a ajouté une couche de "Typage Statique".
- **La Transpilation :** Les navigateurs et Node.js ne savent pas lire le .ts. On utilise un compilateur (tsc) qui transforme le TS en JS pur avant l'exécution.

2. Pourquoi c'est indispensable aujourd'hui ? (Les "Pain Points")

A. Le principe du "Fail Fast" (Échouer vite)

En JS classique, une erreur de type (passer un objet au lieu d'une string) ne plante qu'au moment de l'exécution. En TS, l'erreur s'affiche en rouge instantanément.

B. L'Auto-complétion (IntelliSense)

L'éditeur connaît la structure de vos objets. Cela évite de chercher pendant 20 minutes si la propriété s'appelle user_id, userId ou id.

3. Le Coeur de la POO : Les Modificateurs d'Accès

En JavaScript, tout est "ouvert". N'importe qui peut modifier n'importe quoi. TypeScript apporte la **sécurité** avec trois mots-clés essentiels :

Modificateur	**Visibilité**	**Cas d'usage "Ligue Sportive"**
public (par défaut)	Partout (Intérieur, Extérieur, Enfants).	Les méthodes que l'utilisateur appelle : connexion(), ajouterAuPanier().
private	**Uniquement** à l'intérieur de la classe.	Les données sensibles : motDePasse, cleAPI. Personne ne doit y toucher

		directement de l'extérieur.
protected	Dans la classe **ET** ses classes filles (héritage).	Les propriétés partagées : id, dateCreation. Utile si Adherent et Produit héritent d'une base commune.

4. Application concrète dans votre semaine de formation

Cas 1 : La sécurité des données (Projet Ligue Sportive)

Imaginez votre classe Adherent. Si le soldeLocation est en public, n'importe quelle partie du code peut faire adherent.soldeLocation = 1000000.

- **Solution TS :** On le met en private. On crée une méthode public payer(montant: number) qui vérifie si le solde est suffisant avant de modifier la valeur. **C'est l'encapsulation.**

Cas 2 : Le contrat d'interface (API Node.js)

Jeudi, vous allez créer une API. Avec TS, vous définissez une interface IProduit.

- Si votre Front-end (React) essaie d'afficher un produit sans le prix alors que l'interface dit qu'il est obligatoire, l'application ne compilera même pas. Vous évitez les undefined sur l'écran du client.

Cas 3 : L'héritage propre (Refactoring)

Vous aurez des Adherents et des Admins.

- L'Admin est un Adherent avec plus de droits. Avec protected, l'Admin peut accéder à la méthode genererToken() de son parent, mais le reste du code (le Front) ne le peut pas.

5. Comparaison Flash : JS vs TS

Caractéristique	JavaScript (JS)	TypeScript (TS)
Typage	Dynamique (change à tout moment)	Statique (fixé à la déclaration)
Détection d'erreurs	À l'exécution (chez l'utilisateur)	À la compilation (chez le dév)
Maintenance	Difficile sur de gros projets	Très robuste pour le

		refactoring
Vitesse de dév	Rapide au début, lent à la fin	Un peu plus lent au début, très rapide après