Pourquoi utiliser TypeScript plutôt que JavaScript ?

Qu'est ce que le TypeScript ?

TypeScript est un sur-ensemble typé qui a pour but d'améliorer et de sécuriser la production de code JavaScript. Tous les frameworks front n'utilisent pas TypeScript mais cela reste une bonne option pour la sécurité du code.

L'incapacité de JavaScript à intégrer des éléments tels que les types et les contrôles d'erreurs de compilation en fait un mauvais choix pour le code côté serveur dans les sociétés et les grandes bases de données.

Les types dans TypeScript sont facultatifs, et chaque fichier JavaScript est un fichier TypeScript valide. Bien que le compilateur n'aime pas quand il y a des erreurs de types dans les fichiers initiaux, il rend un fichier JavaScript qui fonctionne.

Les types dans TypeScript

Les langages de programmation se divisent en deux catégories : les types statiques et les types dynamiques.
- Dans les langages à typage statique, le type de la variable doit être connu au moment de la compilation. Si une variable est déclarée, le compilateur doit savoir (ou pouvoir en déduire) s'il s'agit d'un nombre, d'une chaîne de caractères ou d'un booléen.
- Dans les langages à typage dynamique, ce n'est pas nécessairement le cas. Le type d'une variable n'est connu qu'au moment de l'exécution du programme.

TypeScript peut prendre en charge le type statique, alors que JavaScript ne le fait pas.

Les systèmes de types statiques permettent de créer ses propres types de données composites. Cela permet aux développeurs d'exprimer leurs intentions de manière plus détaillée.

Les types explicites permettent également au code de s'auto-documenter : ils garantissent que les variables et fonctions correspondent à ce qui est prévu et permettent à l'ordinateur de se souvenir du contexte environnant.

TypeScript est plus fiable :
- Contrairement à JavaScript, le code TypeScript est plus fiable et plus facile à remanier. Cela permet aux développeurs d'éviter les erreurs et de procéder à des réécritures beaucoup plus facilement.

TypeScript est plus explicite :
- Rendre les types explicites permet de faire un focus sur la manière dont le système est exactement construit et dont les différentes parties de celui-ci interagissent les unes avec les autres.

TypeScript et JavaScript sont pratiquement interchangeables :
- Comme JavaScript est un sous-ensemble de TypeScript, il est possible d'utiliser toutes les bibliothèques JavaScript dans son code TypeScript.

Types Any & Unknown :
Le type “Any” peut couvrir tout ce qu’on veut, “Unknown” est son homologue qui assure la sûreté du type.
Pour échapper au système de typage, “any” permet de lui attribuer n’importe quelle variable JavaScript. Il est fréquemment utilisé pour modéliser des variables entrantes (provenant d’API tierces, par exemple) qui n’ont pas encore été vérifiées et dont le type est inconnu.
“Unknown” ressemble beaucoup à “any”, mais il ne permet pas d’effectuer des opérations sur la variable avant qu’elle ne soit explicitement vérifiée par le système de types.

Source : *mobiskill*
Date de l'article : 22 février 2021

Any:
- Tout le confort du typage en "any" se fait au prix de la perte de la sécurité du type. La sécurité du type est l'une des motivations principales pour l'utilisation de TypeScript et vous devriez éviter d'utiliser "any" quand ce n'est pas nécessaire

Source : *Documentation officielle*

- En utilisant "any", vous vous exposez problèmes qui sont difficiles à tracer et à debuguer, surtout une fois le code déployé en production.
Recommandation:
- - Utiliser le bon type
-- Utiliser unknow (lorsque vous ne connaissez pas le type et voulez assurer la sécurité du type.
-- Vous pouvez utilisez le compilateur "noImplicityAny" pour signaler tous les "any" implicite comme une erreur.

Source : *dev.to*