Comparaison entre Elm et JavaScript pour le projet TcTurtle

Tout d'abord Elm est un langage fonctionnel avec une vérification stricte des types. Cela signifie que la compilation détectera de nombreuses erreurs courantes en JavaScript, telles que *"undefined is not a function"*. Cela garantit que les commandes TcTurtle sont bien interprétées et qu'il n'y aura pas d'imprévus lors du rendu SVG. En revanche, en JavaScript, les types sont vérifiés à l'exécution, rendant parfois le débogage plus difficile.

Ensuite, Elm possède une bibliothèque de parsing (*elm/parser*) qui facilite la transformation d'une chaîne de texte en un format de données structuré. Cela rend l'analyse du langage TcTurtle plus robuste.

Un avantage fondamental d'Elm réside dans le fait que son algorithme est, en lui-même, une preuve mathématique de son bon fonctionnement. Une fois qu'un programme Elm compile sans erreur, il est garanti qu'il fonctionnera toujours comme prévu, indépendamment de la machine sur laquelle il s'exécute. En revanche, en JavaScript, l'exécution du code dépend de l'environnement. Cela signifie qu'un script JavaScript peut fonctionner correctement sur une machine et échouer sur une autre, entraînant ainsi des comportements imprévisibles.

De plus, Elm compile en un code optimisé, ce qui le rend rapide et stable. Cependant il a un nombre limité de bibliothèques spécialisées. À l'inverse, JavaScript permet un développement plus rapide grâce à ses nombreux outils prêts à l'emploi, mais cela peut entraîner plus de bogues.

En conclusion Elm garantit une meilleure sécurité, un parsing robuste et un comportement cohérent sur toutes les machines. JavaScript, quant à lui, offre plus de flexibilité et un vaste écosystème de bibliothèques, mais au prix d'une gestion plus complexe des erreurs.