Zeste de savoir

Jouons avec WebAssembly Studio !

20 janvier 2019

Table des matières

1. Introduction

Dans le dernier billet , j'avais abordé la fonction de wasm-bindgen dans la chaîne d'outils mise à disposition, tout en oubliant de présenter un service qui risque de s'avérer très pratique lorsqu'il sera opérationnel : j'ai nommé WebAssembly Studio !

Nous avons ici un Web IDE entièrement dédié à wasm et permet à l'utilisateur de gérer plutôt simplement l'architecture de son projet. En passant par la configuration de gulp, l'intégration de nouveaux modules node ou encore la création libre de nouveaux fichiers et répertoires, tout y est. Il ne reste plus qu'à s'y mettre.

> *i* Il n'est pas nécessaire de connaître les différents langages qui seront présentés plus bas pour lire ce billet.

2. Champ d'action

A l'heure où j'écris ces lignes, l'environnement supporte 4 langages :

1. C ;
2. TS ;
3. Rust ;
4. Et, bien évidemment, wasm lui-même.

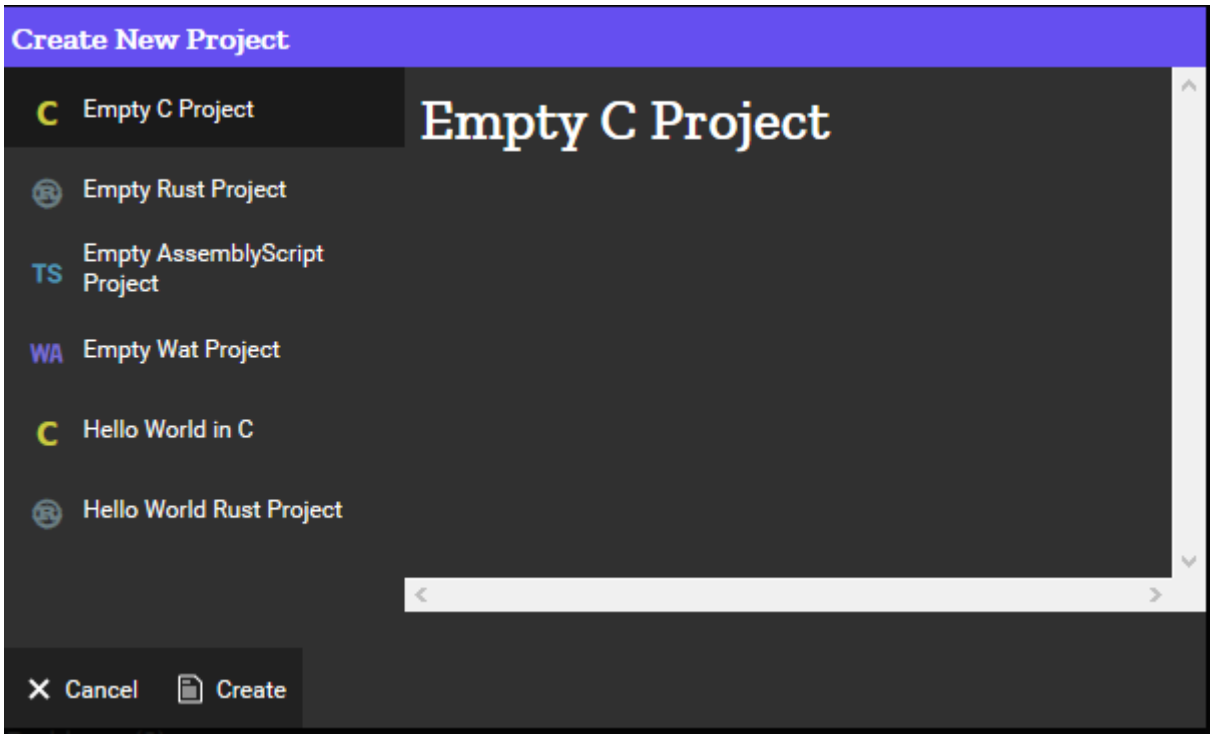

FIGURE 2. – Menu WebAssembly Studio

L'UI reste relativement intuitive et peu chargée, ne contenant que l'essentiel pour télécharger, exporter, compiler et exécuter un projet.

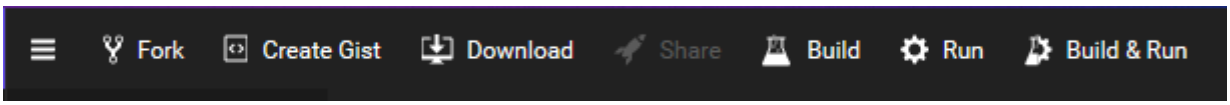

FIGURE 2. – Sidebar de l'IDE

Une fois votre projet compilé, vous pourrez avoir accès à de nouvelles options dans le menu contextuel[1] visant à optimiser, transpiler, désassembler ou même visualiser votre code sous ses différentes formes grâce au backend Binaryen/Emscripten.

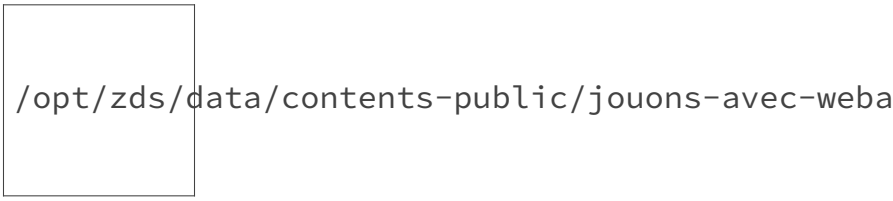

FIGURE 2. – Options du menu contextuel

Au cas où vous vous poseriez la question, le `build.ts` que vous voyez n'a rien de spécial. Il contient simplement les instructions de gulp destinées aux fichiers produits en sortie. Vous pouvez le consulter vous-même en allant jeter un œil !

1. Que vous pouvez déclencher par un clic droit sur un élément de l'arborescence se trouvant à gauche de votre écran.

3. État actuel de l'IDE

Bien que ce très bon outil en devenir soit à notre disposition, il est encore très imparfait et ne nous permettra l'exécution que de très modestes tests (qui ne traitent pas de données complexes, en réalité). Ajoutons à cela qu'il souffre de bugs plutôt lourds puisque tout ce qui requiert les services de Binaryen n'aboutira à rien pour le moment (le backend ne répond pas).

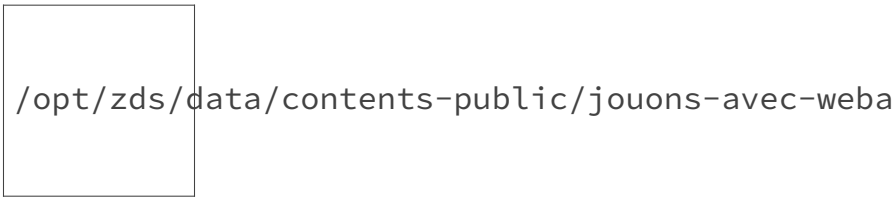

FIGURE 3. – Services défectueux de Binaryen

Il faut toutefois garder à l'esprit que WebAssembly Studio se trouve encore dans son cycle de bêta et que cette situation ne sera certainement pas définitive. Affaire à suivre !

4. Conclusion

5. Voir aussi

— Le site de WebAssembly Studio
— Wasm-bindgen, qu'est-ce que c'est ?

Liste des abréviations

TS TypeScript. 1

UI User Interface. 2