Qu'est-ce que le Web Assembly ?

Qu’est-ce qui a motivé la création du WebAssembly ?

Aujourd'hui, en moyenne, les humains passent 6 heures et 56 minutes sur Internet chaque jour, ce qui est considérable et leur navigation est une grande partie de ce temps.

Le Web 2.0 a apporté de nombreux changements aux usages d'Internet. On peut maintenant créer de véritables applications dans le navigateur. Les derniers moteurs JavaScript, conjointement à l'arrivée de nouveaux standards web tels que le WebGL, ont permis une transformation radicale de l'utilisation d'Internet. Nous sommes passés d'un monde de consommation à un monde interactif.

Ces nouvelles utilisations d'Internet ne sont pas en adéquation avec l'idée première de JavaScript. Bien que JavaScript ait été amélioré, son caractère en tant que langage de script, son fonctionnement par interprétation et la rétrocompatibilité qu'il doit conserver depuis 1997 font qu'il ne peut plus satisfaire tous les usages du web.

Cependant, comme nous le découvrirons plus en détail plus tard, JavaScript demeure obligatoire pour l'utilisation du web et le WebAssembly n'est adapté qu'à peu de cas d'utilisation du web.

Quelles sont les prédécesseurs et alternatives ?

Avant l'arrivée de WebAssembly et même avant l'amélioration des performances de JavaScript grâce à la compilation JIT (Juste-à-Temps), il y avait déjà des alternatives à JavaScript pour exécuter du code dans le web. Voici une liste des principales solutions qui étaient disponibles :

- Action Script
- Flash

- Applet Java

Ces solutions, qui ont été abandonnées pour plusieurs raisons, étaient notamment handicapées par la nécessité d'installer un logiciel sur le PC de l'utilisateur, ce qui comportait des risques de sécurité et de commodité.

L'avènement des appareils mobiles a encore affaibli cette technologie.

Une autre possibilité est d'utiliser JavaScript avec les nouveaux standards, ce qui n'offre pas les mêmes performances, mais reste une alternative intéressante par rapport au WebAssembly. Celui-ci présente l'avantage de s'appuyer sur le langage et l'écosystème JavaScript, bien rodés.

Qu’est-ce que c’est que le WebAssembly ?

Le WebAssembly (WASM) est une toute nouvelle méthode d'exécution d'instructions non seulement dans le navigateur, mais aussi dans d'autres contextes. Après de nombreuses tentatives externes, le World Wide Web Consortium (W3C) a proposé une solution.

Cette solution devait répondre aux objectifs suivants :

- Efficacité
- Rapidité
- Sécurité
- Deboggabilité
- Part of the Web

Le point "Part of the Web" a pour but d'éviter les problèmes rencontrés par le passé lors de l'installation de Java et de l'exécution de code en dehors du navigateur. De plus, il fait référence à la grande compatibilité du Web et à l'utilisation de certaines API Web telles que `BigInt`.

Le WebAssembly, également appelé WASM, est une ISA (Instruction Set Architecture) virtuelle.

Une ISA est un modèle abstrait des opérations de base qu'il est possible d'effectuer avec un ordinateur.

Généralement, une ISA est un format binaire conçu pour s'exécuter sur une machine spécifique ou une architecture de processeur particulière. Le WebAssembly ne fait pas partie de cette catégorie, car il est destiné à une machine virtuelle et non à une machine physique.

L'ISA du WebAssembly a été conçue pour être compacte, portable et sécurisée. Les ISA classiques ne sont pas nécessairement conçues pour être compactes, mais le WebAssembly est destiné à être

servi par un serveur web.

En raison de son jeu d'instructions, le WebAssembly n'est pas destiné à être codé directement. Il existe un langage assembleur qui permet d'avoir une représentation de chacune des instructions du WebAssembly, appelé WAT. Il est principalement utilisé pour le débogage, l'apprentissage ou pour aller plus loin dans l'optimisation.

Bien qu'il ne soit pas destiné à être codé directement, il peut être utilisé comme cible de compilation pour d'autres langages. **Ceci permet d'ouvrir le web à d'autres langages que le JavaScript.**

Le jeu d'instruction de WebAssembly dans sa version 1.0 contient 172 instructions. C'est assez peu, compte tenu de la duplication en fonction des types. En réalité, il y a environ cinquante instructions différentes. Ce nombre va augmenter avec le temps et les ajouts au standard, mais apprendre le WebAssembly maintenant est une belle opportunité, car en prenant les ajouts petit à petit, vous aurez une meilleure compréhension.

L’état de l’art aujourd’hui

Au moment de l'écriture, le WebAssembly en version 1.0 est pratiquement entièrement pris en charge par les navigateurs modernes. Vous pouvez suivre l'avancement du standard et des disponibilités sur la roadmap du projet “https://webassembly.org/roadmap/”.

Le WebAssembly promet beaucoup :

- Exécuter du code plus rapidement
- ISO-Fonctionnel
- Multi-langage

Dans la réalité, les choses ne sont pas si belles. En grande partie parce que la technologie est jeune, mais on ne sait pas comment elle va évoluer.

Exécuter du code plus rapidement

En effet, le WASM permet d'exécuter du code plus rapidement, cependant on ne peut pas faire toutes les opérations possibles avec. Par exemple, on ne peut pas manipuler le DOM.

Quand on sait qu'aujourd'hui, un processeur passe la plupart de son temps en mode IDLE à attendre des instructions, avoir un langage plus rapide ne changera pas les performances dans tous les cas. D'autres standards du Web vont traiter ces problèmes de performance, comme le shadow DOM et les "Web Components". Mais les cas où l'utilisation du WebAssembly apportera un gain de performance considérable sont peu fréquents. On peut citer les jeux vidéo, la

cryptographie, le calcul brut de données, l'intelligence artificielle et encore d'autres. Peut-être que le gain de performance apporté par le WebAssembly va amener de nouveaux usages.

ISO-fonctionnel

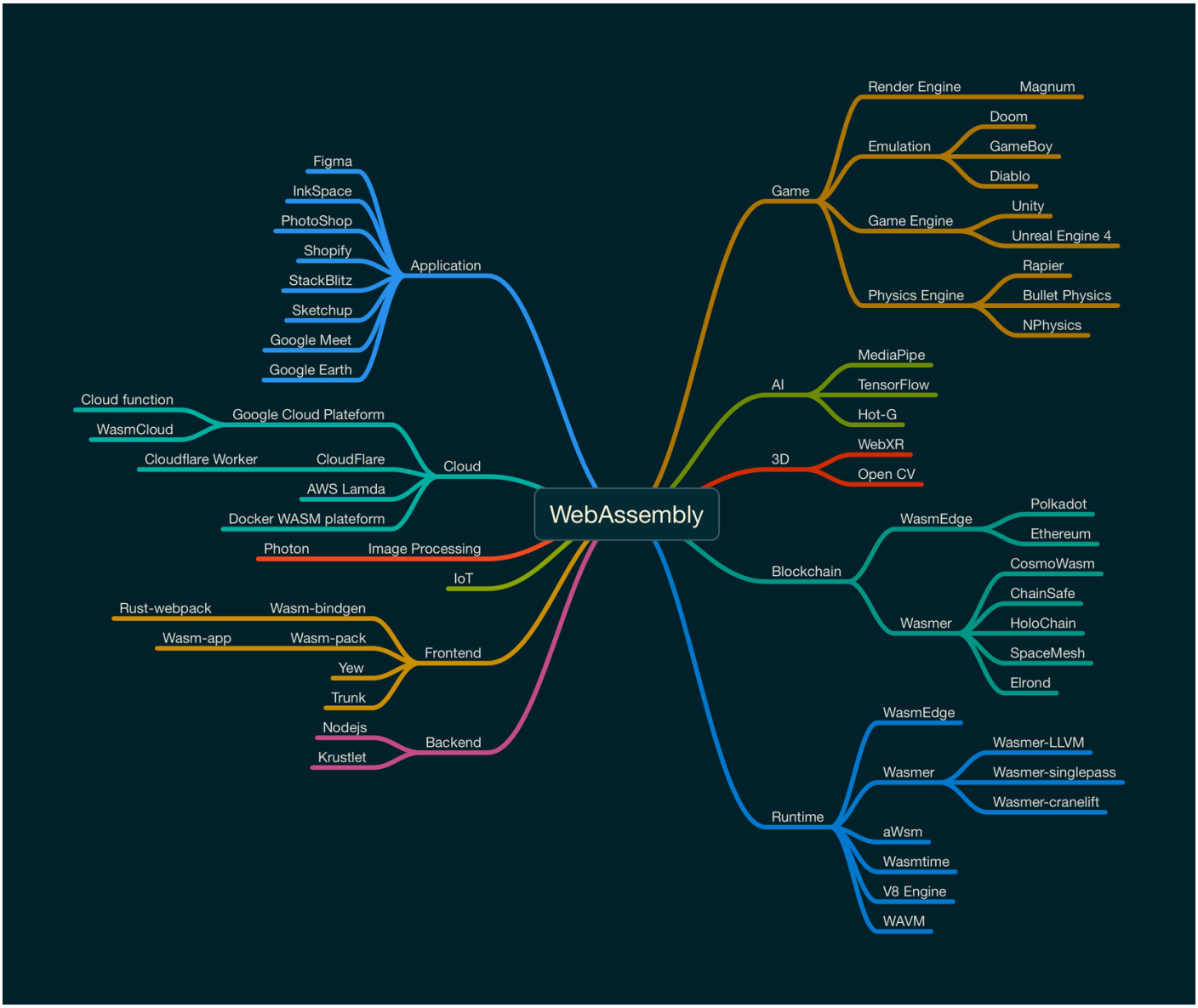
Cette promesse est presque atteinte. Au moment de l'écriture, Safari n'implémente pas une fonctionnalité du standard 1.0, mais cela n'empêche pas le fonctionnement du WebAssembly sur ce navigateur. Là où le support des navigateurs peut être un peu faible, c'est sur les débogueurs qui ne sont pas encore totalement au point et qui peuvent rendre la tâche un peu compliquée.

Multi-langage

Malgré le fait que le WASM soit destiné à la compilation, le support actuel des langages n'est pas parfait et aujourd'hui, beaucoup de langages restent indisponibles ou incomplets.

Les librairies

Les libraires peuvent tirer parti des fonctionnalités du WebAssembly. On peut citer QR2Text, qui permet de générer des `QrCode` et qui est plus performant que ses homologues en JavaScript (le `bundle` est plus petit). Cette librairie vient à l'origine d'un outil CLI fait en RUST et aucun changement dans le code n'a été nécessaire. D'autres frameworks, comme Flutter Web, utilisent le WebAssembly pour intégrer SKIA, un moteur de rendu qui permet d'afficher dans le web via le canvas une application Flutter.



Ressources

https://datareportal.com/reports/digital-2022-global-overview-report

https://www.youtube.com/watch?v=kCnYRhkfWHY

https://nostarch.com/art-webassembly

Revision #2
Created 2023-05-05 12:38:10 UTC by Noé Larrieu-Lacoste
Updated 2023-05-05 12:43:02 UTC by Noé Larrieu-Lacoste