УДК 004.42

Григоренко А.В.,
студент института цифровой экономики
и информационных технологий
Российский экономический университет им. Г.В. Плеханова
Россия, г. Москва

НЕКОТОРЫЕ ВОПРОСЫ ИСПОЛЬЗОВАНИЯ ТЕХНОЛОГИИ WEBASSEMBLY И WEB USB В СОВРЕМЕННЫХ БРАУЗЕРАХ

Аннотация: *В статье рассматривается технология Web Assembly, принцип ее работы, области применения и возможности, которые открываются с ее интеграцией в современные браузеры. Плюсы и минусы использования данной технологии.*

Ключевые слова: *WebAssembly, браузеры, кроссплатформенность, безопасность, байткод, нейронные сети.*

Annotation: *The article discusses the technology Web Assembly, the principle of its work, the scope and opportunities that open with its integration into modern browsers. Pros and cons of using this technology..*

Key words: *WebAssembly, browsers, cross-platform, security, bytecode, neural networks.*

Результатом научно-технического прогресса последних десятилетий стало повседневное использование компьютерной техники в различных сферах человеческой деятельности, которая оказала и продолжает оказывать влияние на все сферы жизни общества. Вслед за автором [2, 5-7] будем считать, что информационные технологии изменили и продолжают изменять наш мир. То, что было ранее трудно вообразить – сейчас является рядовыми вещами. К примеру, обмен данными – невероятно интересный и сложный процесс, для развития и совершенствования которого потребовалось немало времени [8].

Появление интернета и использование его как единой глобальной сети невероятно упростило жизнь людям. Однако для передачи данных в интернете используются разнообразные и часто сложные технологии.

С появлением интернета веб-технологии находятся в постоянном развитии. Сначала была изобретена гипертекстовая разметка и статичный контент сайта, позже он сменился динамическим, появились новые инструменты для генерации html и css, специальные фреймворки, предназначенные для разделения веб-сайта на модули, рендеринг страниц на стороне сервера. Позже стали популярными SPA [10], создали разнообразные инструменты сборки-бандлирования javascript кода и стилей. Мир фронтэнда очень расширился, но одна из основных проблем веба осталась – невозможность написать на javascript в браузере полноценные приложения, которые могли бы работать как нативные.

Однако появилась технология, которая позволяет запускать готовые приложения прямо в браузере - WebAssembly. Под термином WebAssembly будем подразумевать бинарный формат инструкций для стековой виртуальной машины [11]. WebAssembly представляет собой переносимое абстрактное синтаксическое дерево, то есть конечное помеченное ориентированное дерево, в котором внутренние вершины сопоставлены (помечены) с операторами языка программирования, а листья — с соответствующими операндами [1], обеспечивающее как более быстрый парсинг, так и более быстрое выполнение кода, чем просто сам JavaScript. За счет этого данная технология приносит новые возможности и большой прирост производительности. Сами разработчики не пишут инструкции на webassembly коде, они компилируют низкоуровневые языки, такие как C, C++, Rust, а также все интерпретируемые языки.

По сути, WebAssembly позволяет запускать приложения, написанные на нескольких языках, в вебе на скорости, близкой к скорости обычных веб-сайтов. Нужно сразу обратить внимание на то, что код, скомпилированный при помощи WebAssembly, не может работать быстрее, чем javascript. С одной стороны это минус, но с другой стороны современные движки браузеров делают очень качественные оптимизации, которые позволяют достичь около «C++–ных»

скоростей. Замеры скоростей javascript и C++ с применением пяти фильтров на изображении представлены на Рисунке 1.

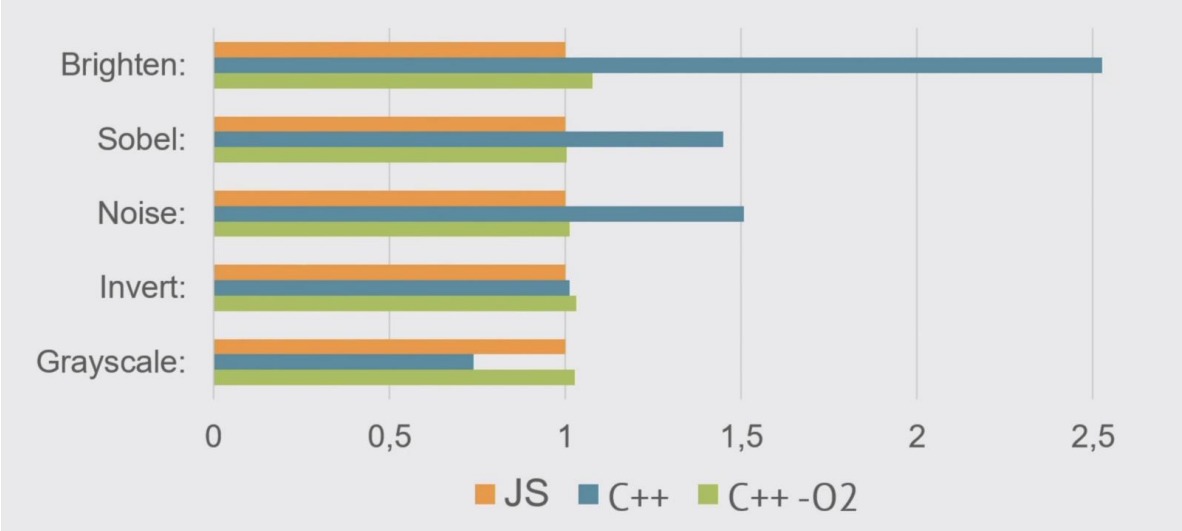

Рисунок 1. Сравнение производительности C++ с Javascript в WebAssembly

Еще одним плюсом является то, что можно вообще не знать, как писать код для WebAssembly, так как ее модули можно импортировать в веб-приложение, предоставив ей функции для использования, написанные на javascript.

Вслед за автором [9] выделим следующие основные цели WebAssembly:

- Быть быстрым, эффективным и портативным - код WebAssembly может выполняться с близкой скоростью на разных платформах, используя общие аппаратные возможности.
- Быть удобочитаемым и отлаживаемым - WebAssembly — это низкоуровневый язык ассемблера, но он имеет удобный для пользователя текстовый формат (спецификация для которого все еще завершается), что позволяет писать, просматривать и отлаживать код вручную.
- Сохранять безопасность. WebAssembly запускается в безопасной изолированной среде. Как и на обычных веб-страницах данная технология будет иметь те же политики безопасности.
- WebAssembly сконструирован таким образом, что он отлично работает с другими веб-технологиями и поддерживает обратную совместимость.

Отвечая на возникающий вопрос «Как WebAssembly вписывается в веб-браузер?», можно сказать, что браузер можно рассматривать, как:

1) Виртуальная машина (VM), которая запускает код веб-приложения, например. код JavaScript, который активирует веб-приложения, написанный разработчиками.

2) Набор веб-API, который веб-приложение может вызывать для управления функциональностью веб-браузера и позволяет использовать различные инструменты (DOM, CSSOM, WebGL, IndexedDB, Web Audio API и т. Д.).

Исторически сложилось, что VM могла загружать только JavaScript. И это устраивало разработчиков, так как JavaScript достаточно мощный, чтобы решить большинство проблем, которые люди сегодня испытывают в Интернете. Однако мы столкнулись с проблемами производительности при попытке использовать JavaScript для более интенсивных приложений, таких как 3D-игры, Virtual и Augmented Reality, компьютерное зрение, редактирование изображений и видео и ряд других доменов, требующих собственной производительности.

API JavaScript предоставляет разработчикам возможность создавать модули, таблицы и экземпляры. Отдав экземпляр WebAssembly, код JavaScript может синхронно вызывать его экспорт, который отображается как обычные функции JavaScript. Произвольные функции JavaScript также можно синхронно вызывать кодом WebAssembly, передавая в них функции JavaScript как импорт экземпляра WebAssembly. Поскольку JavaScript имеет полный контроль над тем, как загружается, компилируется и запускается код WebAssembly, разработчики JavaScript могут даже думать о WebAssembly как о функции JavaScript для эффективного создания высокопроизводительных функций.

Для компиляции используется и создания WASM кода используется сборщик wasm Emscripten.

Инструмент Emscripten способен взять практически любой исходный код C / C ++ и скомпилировать его в модуль .wasm, а также необходимый код «склеивания» JavaScript для загрузки и запуска модуля и HTML-документ для отображения результатов кода (Рисунок 2).

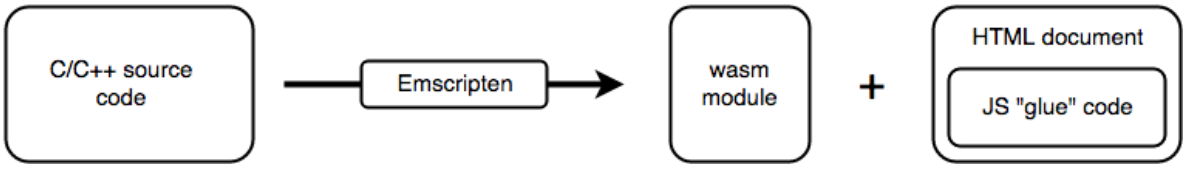

Рисунок 2. Принцип работы с Emscripten

Процесс рыботы Emscripten можно описать так:

1) Emscripten сначала передает C / C ++ в clang + LLVM - зрелую программную цепочку компилятора с открытым исходным кодом C / C ++, например, поставляемую как часть XCode на OSX.

2) Emscripten преобразует скомпилированный результат clang + LLVM в двоичный файл .wasm.

3) Сам по себе WebAssembly не может напрямую обращаться к DOM; он может вызывать только JavaScript, передавая примитивные типы данных с целыми числами и с плавающей точкой. Таким образом, для доступа к любому веб-API WebAssembly необходимо вызвать JavaScript, который затем совершает вызов Web API.

Таким образом WebAssembly обеспечивает браузер следующими возможностями [12]:

- Редактирование изображения / видео.
- Игры, которые должны иметь быстрый старт.
- AAA-игры с тяжелыми ресурсами.
- Игровые порталы.
- Музыкальные приложения (потоковая передача, кеширование).
- Распознавание изображений.
- Увеличение видео в реальном времени (например, установка шляп на головы людей).

- VR и дополненная реальность.
- Научная визуализация и моделирование.
- Интерактивное образовательное программное обеспечение и новостные статьи.
- Моделирование / эмуляция платформы (ARC, DOSBox, QEMU, MAME, ...).
- Устные переводчики и виртуальные машины.
- POSIX, позволяя портировать существующие приложения POSIX.
- Инструмент разработчика (редакторы, компиляторы, отладчики, ...).
- Удаленный рабочий стол.
- VPN.
- Шифрование.
- Локальный веб-сервер.
- Общие пользователи NPAPI, в рамках модели безопасности Интернета и API.
- Жирный клиент для корпоративных приложений (например, баз данных).

Одной из интересных возможностей, которые открываются при использовании данной технологии – управление внешним устройством прямо из браузера. Ранее уже предпринимались попытки работы с внешними устройствами на языке javascript. Одним из примеров может быть платформа Espruino. Вообще говоря, Espruino — это несколько вариантов микроконтроллерных устройств, в которых прошит встроенный интерпретатор JavaScript. Espruino Pico — самое миниатюрное из них. Оригинальный интерпретатор JavaScript, используемый в Espruino, предназначен для быстрой разработки на устройствах с ограниченными процессорными ресурсами. Есть его версии для целого перечня платформ, начиная с ESP8266 и до Raspberry Pi [3]. С появлением WebAssembly архитектуру, позволяющую взаимодействовать

с USB, можно переместить в песочницу браузера, и тем самым создавать драйверы прямо в браузере.

WebAssembly также возможно использовать для сложных вычислений. К примеру, рассмотрим самый популярный фреймворк для построения графиков в браузере – D3.js. Опираясь на автора [4], можно сделать вывод, что данная библиотека, оперируя огромными массивами данных, использует в своей основе работу с SVG и canvas. SVG – основанный на XML векторный формат изображения для двумерной графики с поддержкой интерактивности и анимации. Работа с SVG происходит в браузере с использованием обычного javascript, и задействование ресурсов дополнительных машинных средств невозможно, соответственно скорость работы с SVG не может превысить скорости выполнения javascript без оптимизаций. Canvas – очень эффективный инструмент, который позволяет делать вычисления прямо на GPU. Альтернативой ему может быть WebAssembly, которая также способна выполнять вычисления как на CPU, так и на GPU. Более того, API работы с WebGL2 стал намного удобнее и понятнее.

WebAssembly открывает новый мир в области веб-приложений. Становятся доступными ранее невозможные типы программ и виды взаимодействия с пользователем. Сообщество из ведущих компаний, такие как Google, Mozilla, Apple, Microsoft вместе создают и развивают данную технологию. Все большую популярность приобретают нейронные сети, дополненная реальность и другие интересные технологии, которые вскоре мы сможем использовать прямо на клиентской стороне – в нашем браузере.

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ

1. Википедия. Абстрактное синтаксическое дерево. [Электр. ресурс]. URL: https://ru.wikipedia.org/wiki/Абстрактное_синтаксическое_дерево (Дата обращения 26.11.2018).

2. Гаспариан М.С., Уринцов А.И. – ПЕРСПЕКТИВЫ РАЗВИТИЯ ЭЛЕКТРОННОГО ОБМЕНА ДАННЫМИ И СТАНДАРТ XML – В сборнике:

Математические и инструментальные средства в современных экономических системах. Сборник научных трудов. Московский государственный университет экономики, статистики и информатики – 2003 – 17-20 с.

3. Лыкошин А.С. Espruino Pico. Учимся программировать USB-микроконтроллер на Javascript и делаем из него токен авторизации – журнал Хакер – 2017.

4. Лыкошин А.С. Визуализируем данные на Javascript с помощью D3.js – журнал Хакер – 2015.

5. Уринцов А.И. НЕКОТОРЫЕ ВОПРОСЫ ВЛИЯНИЯ НАУЧНО-ТЕХНИЧЕСКОГО ПРОГРЕССА НА РАЗВИТИЕ СЕТЕВОЙ ЭКОНОМИКИ И ИНФОРМАЦИОННОГО ОБЩЕСТВА – сборник: Опережающее управление социально-экономическим развитием регионов: устойчивое развитие экономики & электронное управление экономическим развитием материалы 4-ой Международной научно-практической конференции. – 2013 – 318-326 с.

6. Уринцов А.И., Федорищева С.В. – О НЕКОТОРЫХ ТЕНДЕНЦИЯХ В ОБЛАСТИ ЭЛЕКТРОННОГО ОБМЕНА ДАННЫМИ – Электронный бизнес Тезисы докладов семинара – 2001 – 62-65 с.

7. Уринцов А.И., Староверова О.В. – НЕКОТОРЫЕ ТЕНДЕНЦИИ ИНФОРМАТИЗАЦИИ ОБЩЕСТВА – ООО "Издательство "Юнити-Дана" – 2016 – 125-128 с.

8. Уринцов А.И. – ЭЛЕКТРОННЫЙ ОБМЕН ДАННЫМИ. УЧЕБНОЕ ПОСОБИЕ – Евразийский открытый институт – 2011.

9. MDN Документация. Mozilla: Что такое WebAssembly? [Электр. ресурс]. URL: https://developer.mozilla.org/en-US/docs/WebAssembly/Concepts (Дата обращения 26.11.2018).

10. Michael Mikowski. Single Page Web Applications: JavaScript end-to-end 1st Edition – Manning Publications Co. – 2014 –5 с.

11. Mike Rourke. Learn WebAssembly – Packt Publishing. – 2018 – 7 с.

12. WebAssembly official site. WebAssembly Community Group: WebAssembly documentation. [Электр. ресурс]. URL: https://webassembly.org/docs/use-cases/ (Дата обращения 26.11.2018)