УДК 004.09

БИНАРНЫЙ ФОРМАТ ИНСТРУКЦИЙ WEBASSEMBLY

Автор: Нечитайло Д.С., студент гр. ИТб-201, IV курс,

Научный руководитель: Николаев Петр Игоревич, доцент Кафедры информационных и автоматизированных производственных систем

Кузбасский государственный технический университет
имени Т.Ф. Горбачева

г. Кемерово

1. Введение

WebAssembly (WASM) представляет собой новую технологию, привносящую большие изменения в веб-разработку. Этот доклад представляет обзор WebAssembly, его историю, преимущества, применение и перспективы развития.

2. Что такое WebAssembly?

WebAssembly — это бинарный формат команд, предназначенный для запуска в веб-браузерах. Это не язык программирования, а скорее целевая платформа для различных языков программирования, таких как C++, Rust и других, которые можно скомпилировать в WASM. Таким образом, WASM позволяет веб-приложениям выполнять вычисления на более низком уровне, чем может обеспечить JavaScript.

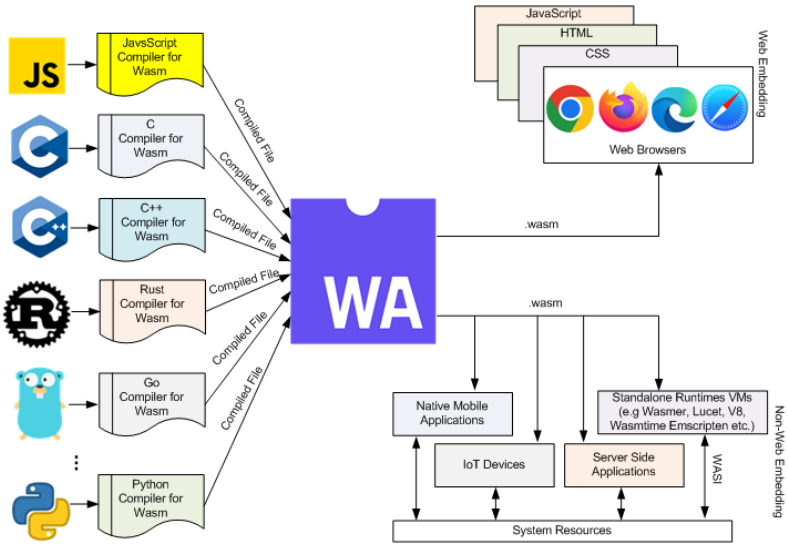

Рисунок 1 - Архитектура потока данных WebAssembly.

3. История

Идея WebAssembly появилась в 2015 году, когда главные разработчики браузеров начали работать над идеей. В июне 2018 года WebAssembly стал полноценной составляющей множества браузеров, За короткий срок WASM приобрел значительную популярность благодаря своей производительности и универсальности.

4. Преимущества

Производительность: WebAssembly предоставляет значительно более высокую производительность по сравнению с JavaScript, особенно для вычислительно интенсивных операций.

Многоплатформенность: WASM является многоплатформенной технологией, что означает, что код, скомпилированный в WASM, может реализовываться в различных средах, включая браузеры и сервера.

Безопасность: Благодаря строгому контролю доступа к памяти, WASM обеспечивает повышенный уровень безопасности для веб-приложений.

Интеграция с JavaScript: Он без труда интегрируется с существующими веб-приложениями, созданными на языке JavaScript. Это позволяет поэтапно внедрять WebAssembly в разработки, не переписывая целиком код.

5. Применение

Игровая индустрия: WebAssembly открывает новые возможности для создания высокопроизводительных веб-игр с использованием языков программирования, таких как C++ и Rust.

Научные вычисления: WASM может быть эффективно использован для выполнения сложных научных вычислений в веб-браузере, что делает его ценным инструментом для ученых и исследователей.

Обработка изображений и видео: Благодаря своей высокой производительности, WASM может быть применен для реализации веб-приложений для обработки изображений и видео на стороне клиента.

6. Перспективы развития

WebAssembly находится на стадии активного развития, и в будущем можно ожидать еще большего расширения его возможностей. Это может включать в себя дальнейшее улучшение производительности, расширение

поддержки языков программирования и внедрение новых возможностей для интеграции с другими веб-технологиями.

7. Пример работы с WebAssembly

Создадим простой пример работы с WebAssembly на основе JavaScript. Для этого мы сначала скомпилируем простую функцию на C в WebAssembly, а затем загрузим и вызовем эту функцию из JavaScript.

1. Для начала создадим простой файл на языке C, который будет содержать функцию сложения двух чисел:

```c
// add.c
int add(int a, int b) {
    return a + b;
}
```

Рисунок 2 - функция сложения двух чисел на языке C.

2. Компиляция в WebAssembly:

С помощью Emscripten мы скомпилируем этот файл в WebAssembly.

```bash
emcc add.c -o add.wasm -s WASM=1
```

Рисунок 3 – компиляция файла в WebAssembly.

3. JavaScript для загрузки и вызова WebAssembly:

Теперь давайте напишем JavaScript-код, который загрузит наш WebAssembly-модуль и вызовет функцию сложения:

```
// Функция для загрузки WebAssembly
async function loadWebAssembly(filename) {
    const response = await fetch(filename);
    const bytes = await response.arrayBuffer();
    const { instance } = await WebAssembly.instantiate(bytes);
    return instance.exports;
}

// Загрузка и использование WebAssembly
(async () => {
    const wasm = await loadWebAssembly('add.wasm');
    const result = wasm.add(5, 3);
    console.log('Результат сложения: ', result);
})();
```

Рисунок 4 - JavaScript-код.

4. HTML-страница:

Создадим простую HTML-страницу, которая подключит наш JavaScript-файл:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Пример работы с WebAssembly</title>
    <script src="main.js" defer></script>
</head>
<body>
    <h1>Пример работы с WebAssembly</h1>
</body>
</html>
```

Рисунок 5 - HTML-страница.

Теперь, когда вы загрузите эту HTML-страницу в браузер, вы должны увидеть в консоли результат сложения, который будет равен 8 (результат сложения 5 и 3).

Это простой пример работы с WebAssembly на основе языка JavaScript.

8. Демонстрация Возможностей WebAssembly

Яркий пример реализации возможностей технологии WebAssembly является масштабный проект – Планета Земля от компании Google

О проекте Google Планета Земля – реализована при помощи технологии WebAssembly, визуализирует сложные 3D модели с бесшовными уровнями, дает возможность просматривать модель планеты с плавной скоростью, что было бы невозможно реализовать с обычным JavaScript.

Уже сейчас WebAssembly активно применяется в таких сферах как:

- Игры, игровые движки
- Эмуляторы, виртуальные машины
- Графические 3D редакторы
- Web клиенты для финансовых площадок
- Базы данных

9. Заключение

WebAssembly представляет собой значимый прорыв в веб-разработке, предоставляя возможность выполнять вычисления на более низком уровне, чем это было возможно ранее с использованием только JavaScript. Его производительность, многоплатформенность и безопасность делают WASM важным инструментом для создания современных веб-приложений.

Список литературы:

1. ya.zerocoder [Электронный ресурс] URL: https://ya.zerocoder.ru/pgt-webassembly-wasm-revoljuciya-v-veb-programmirovanii/
2. researchgate [Электронный ресурс] URL: https://www.researchgate.net/figure/WebAssembly-data-flow-architecture_fig1_373229823