WebAssembly 를 활용한 웹 어플리케이션 성능개선에 관한 연구

이진범, 지도교수 박민호
숭실대학교

jinbomang@gmail.com, mhp@ssu.ac.kr

Using WebAssembly Web Application Performance Improvement Research

Lee Jin Beom, Park Min Ho
Soongsil Univ

요 약

본 논문은 웹 어플리케이션의 발전으로 프론트엔드 개발 시 사용하는 Javascript 언어의 성능적 한계와 기능적인 제한을 C/C++, Rust 등과 같은 컴파일 언어를 웹 브라우저에서 실행할 수 있는 새로운 유형의 코드인 WebAssembly 를 테스트하여 기존 웹 어플리케이션 구조를 개선하고 JavaScript 대비 성능을 개선할 수 있는 방법을 제시한다.

Ⅰ. 서 론

웹 서비스의 발전으로 최근 웹 서비스는 단순한 웹페이지 이외에도 기존의 네이티브 어플리케이션을 대체하는 다양한 종류의 웹 기반의 어플리케이션들이 등장했고 현재 계속 증가하고 있는 추세이다.

이와 같은 이유로 웹 서비스에서 요구하는 기능들은 다양해지고 복잡해졌으며 온라인이 아닌 오프라인에서도 동작이 필요로 하고 있다.

하지만 웹 어플리케이션 프론트엔드 개발 시 사용하는 인터프리터 언어인 JavaScript 는 성능적 한계와 기능적인 제한으로 네이티브 어플리케이션을 성능적인 부분에서 대체하지 못하거나 기능적인 부분을 구현하지 못해 일부 기능은 서버에 데이터를 넘겨 서버에서 처리 후 다시 클라이언트로 처리된 데이터를 넘기는 방식의 구현으로 불필요하게 네트워크 트래픽을 사용하거나 Round-Trip Time(RTT) 이 발생하게 된다.

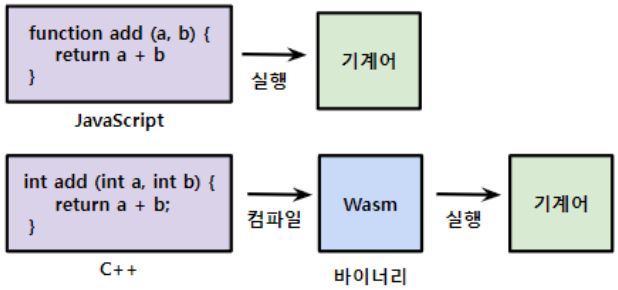
이를 개선할 수 있는 방법으로 2017 년 등장하여 개발 및 연구가 진행중인 WebAssembly 활용해보려고 한다.

WebAssembly 는 최신 웹 브라우저에서 실행할 수 있는 새로운 유형의 코드로 네이티브에 가까운 성능으로 동작하며 컴팩트한 바이너리 포맷을 제공하는 저수준 어셈블리 언어로, C/C++, Rust 등과 같은 언어의 컴파일 타겟으로써 그런 언어로 작성된 프로그램을 웹에서 사용할 수 있게 한다. [1][2]

이러한 WebAssembly 를 테스트하여 기존 웹 어플리케이션 구조를 개선하고 JavaScript 대비 성능을 개선할 수 방법을 제시한다.

Ⅱ. WebAssembly 를 활용한 구조개선

<그림 1>과 같이 JavaScript 는 인터프리터 언어로 작성된 명령어를 실행 시 기계어로 바꾸어 실행합니다. 반면 WebAssembly 는 컴파일러로 명령어를 바이너리 형태인 Wasm 파일로 변형하여 자바스크립트에 비해 실행 시 기계어로 변환이 빠르고 용량이 가볍습니다.



<그림 1> JavaScript, WebAssembly 동작비교

또한 데스크톱이나 서버에서 사용하던 기존 코드들의 재사용 가능하고 JavaScript 에서 지원하지 않는 기능들의 코드들을 구현함으로써 기존에 서버를 활용해야 구현할 수 있었던 기능들을 모두

클라이언트에서 구현함으로써 불필요한 네트워크 트래픽 사용과 RTT 발생을 줄일 수 있다.

Ⅲ. WebAssembly 성능 테스트

본 논문은 데이터를 정렬하는 코드와 이미지에 Grayscale 필터를 적용하는 코드를 JavaScript 와 WebAssembly 로 작성하여 테스트 진행하여 성능비교를 했다.
WebAssembly 코드는 C++ 로 작성 했으며 컴파일을 위해 Esscripten [3] 를 사용했다.

No.	*JavaScript*	*WebAssembly*
1	3,268ms	789ms
2	3,253ms	789ms
3	3,217ms	789ms
4	3,209ms	820ms
5	3,343ms	782ms

<표 1> 정렬 성능비교

정렬 성능비교를 위해 난수데이터를 천만개 생성하여 Array 에 넣었고 해당 Array 를 내림차순으로 정렬하는 동안의 시간을 총 다섯번씩 측정하였다. <표 1> 과 같이 JavaScript 의 경우 평균 3,258ms 시간이 소요됐고 WebAssembly 의 경우 평균 794ms 의 시간이 소요되어 WebAssembly 가 JavaScript 에 비해 4 배정도 빠른 성능을 보여줬다.

No.	*JavaScript*	*WebAssembly*
1	8372ms	3214ms
2	8271ms	3108ms
3	8323ms	3132ms
4	8134ms	3432ms
5	8432ms	2984ms

<표 2> Grayscale 이미지필터 성능비교

이미지 필터 코드는 400 X 400 컬러 이미지를 생성하여 해당 이미지에 Grayscale 필터를 적용하는 시간을 총 다섯번씩 측정하였다. <표 2> 와 같이 JavaScript 의 경우 평균 8,305ms 시간이 소요됐고 WebAssembly 의 경우 평균 3,174ms 시간이 소요되어 WebAsasembly 가 JavaScript 에 비해 대략 2.6 배정도 빠른 성능을 보여줬다.

테스트결과 테스트를 진행한 정렬과 Grayscale 이미지 필터의 성능모두 WebAssembly 가 JavaScript 보다 더 좋은 성능을 보여줬다.

Ⅳ. 결론

WebAssembly 는 웹 어플리케이션에서 사용하기 위해서는 Glue 코드로 JavaScript 사용하고 있기 때문에 완전히 JavaScript 를 대체할 수는 없다. 하지만 이번에 테스트해본 결과 일부 코드를 WebAssembly 로 대체하면 충분히 JavaScript 만을 사용했을 때 보다 성능을 개선할 수 있었고 JavaScript 로 구현할 수 없는 코드를 WebAssembly 로 구현함으로서 이전에 불필요하게 서버를 사용하는 구조를 개선하여 네트워크 트레픽이나 RTT 시간을 줄일 수 있고 오프라인에서도 동작하는 웹 어플리케이션을 개발할 수 있다.

참 고 문 헌

[1] MDN web docs Web Assembly
(https://developer.mozilla.org/ko/docs/WebAssembly)
[2] Web Assembly Future features
(https://webassembly.org/roadmap)
[3] Esscripten
(https://emscripten.org)