Vue.js와 Svelte의 성능 비교 및 분석

엄 세 진

Compare and Analyze of Vue.js and Svelte's Performance

Se-jin Eom

요 약

Node.js의 SPA(Single Page Application) Framework 인 React.js의 등장으로 SPA 개발에 대한 수요가 급증하였다. 뒤이어 나온 Vue.js와 Svelte.js는 낮은 러닝커브로 빠른 구축과 유지보수가 간편한 장점이 있어 두 프레임워크를 첫 렌더링과 업데이트를 중점으로 관찰할 수 있는 웹 페이지 두 가지 유형으로 제작하여 실험하였다. 그 결과 Vue.js는 비교적 규모가 작은 웹 페이지에서는 첫 렌더링이 빠르고, 규모가 커질수록 Svelte.js가 업데이트와 첫 렌더링에서 우수한 성능을 보였다.

Key Words : Web, Front-End, SPA, Vue, Svelte

ABSTRACT

With the advent of React.js, which is the SPA (Single Page Application) Framework of Node.js, the demand for SPA development has surged. Vue.js and Svelte.js, which followed, have the advantage of fast construction and easy maintenance with a low learning curve, so the two frameworks were tested by creating two web pages that can be observed focusing on the first rendering and update. As a result, Vue.js showed faster first rendering on a relatively small web page, and Svelte.js showed excellent performance in updates and first rendering as the size increased.

Ⅰ. 서 론

1. 최신 웹 트렌드

최근 웹 개발의 형태를 보면 SPA(Single Page Application) 형태로 개발 하는 곳이 적지 않다. SPA가 상용화되기 이전에는 MPA(Multi Page Application) 방식으로 웹 개발을 진행하였다. 보통의 웹사이트들은 모든 화면에서 공통으로 보이는 컴포넌트가 존재하는 경우가 많다. 예를 들어 Header, Footer, Navigation 등이 있을 것이다. 하지만 MPA 방식에서는 이러한 공통된 요소의 컴포넌트가 페이지 갱신 또는 페이지 변경의 경우에도 서버에서 항상 모든 컴포넌트가 포함된 html을 다시 그려서 클라이언트로 보내주게 된다. 이렇게 되면 불필요한 자원의 낭비가 생기게 된다. 그래서 등장하게 된 것이 SPA 방식의 웹 개발이다.

SPA를 구현할 수 있는 다양한 프레임워크가 존재하는데, Github의 star 수를 보면 facebook/react.js, vue/vuejs, google/angular.js가 매우 높은 숫자를 기록하고 있는 것을 볼 수 있다. 특히나 React.js는 stackoverflow의 질문 수만 봐도 그 인기를 실감할 수 있다. (2021년 6월4일 16시 기준 310,871개) [1]

2. 연구 배경

Svelte.js는 다른 프레임 워크들보다 다소 늦게 출시하여 이미 다른 프레임워크들이 점유하고 있어 점유율이 많이 낮은 편이다.

Svelte.js는 Vanilla JS와 거의 유사하게 작성이 가능하여 러닝커브가 매우 낮다. 문법과 구조가 간단함에 따라 생산성 역시 높아진다.

프론트엔드 점유율 상위 3개 중 비교적 러닝커브가 낮고, 구조가 간단한 Vue.js와 Svelte.js를 비교하여 비교적 빠른 생산성과 유지보수가 용이한 프레임워크가 적합한 스타트업 등의 업종에서 프레임워크를 적용할 때 어떠한 프레임워크를 선택하는 것이 좋은 지 비교 및 분석을 해 본다.

Ⅱ. 이론적 배경

1. SPA(Single Page Application)

SPA(Single Page Application)은 1개의 페이지만 있는 웹 애플리케이션이다. 즉, 하나의 페이지에서 내부의 내용만 바뀌는 것이다. 서버에서 최초로 한 번만 페이지 전체를 로딩 한 후에, 클라이언트의 동작에 따라 내부의 내용이 바뀌도록 구현이 되어있다. 그리고 WAS(Web Application Server)에 추가적인 데이터나 변경되어야 할 데이터만 추가로 요청하여 페이지에 갱신 시켜준다. 대표적으로 SPA 구현을 지원하는 Node.js 프레임워크는 React.js, Vue.js, Angular.js가 있다.

2. Vue.js [2]

Vue.js는 2014년 2월에 첫 발표로 등장한 자바스크립트 프레임워크이다. Vue.js는 반응형 웹을 위해 Observer pattern으로 구현 되어있는데, 이 덕분에 computed, watched 속성에서 개발자가 따로 통제하지 않아도 데이터를 반응형으로 사용할 수 있게 된다. Vue의 가장 큰 특징 중 하나는 Virtual DOM을 이용한다는 점이다. Virtual DOM이란, 실제 DOM에 적용되기 전 DOM과 같은 구조의 Virtual DOM Tree를 구축한다. 페이지 내부의 내용이 업데이트 되면, Virtual DOM에서 업데이트 된 부분을 교체해주고 바뀌기 전 Virtual DOM과 비교하여 달라진 부분만 실제 DOM으로 반영하여 페이지 전체를 업데이트 하지 않아도 되는 효율적인 방법을 사용하고 있다.

2. Svelte.js [3]

Svelte는 2016년 11월 26일에 첫 출시 된 오픈소스 프론트엔드 프레임워크 또는 컴파일러이다. Vue와 유사한 문법으로 코드를 작성하게 되지만, Svelte는 Vue와는 다르게 Virtual DOM이 없고, Runtime에 로드할 프레임워크가 없음을 의미한다. 즉, 빌드 시 응용프로그램을 이상적인 javascript로 변환한다. Svelte.js에서 이야기하는 3가지 특징은 다음과 같다. Write less code, No virtual DOM, Truly reactive 세 가지가 그 특징들이다.

첫 번째 특징인 write less code는 svelte의 문법이나 표현 방식에서 나타난다. 순수 javascript를 사용하는 것과 거의 유사한 방식으로 작성이 가능하다. 즉, 불필요한 코드 반복을 줄여서 간단한 코드를 만든 것이다. 코드가 간단하고 문법이 쉬우면 자연스레 개발자의 생산성과 유지보수 효율 역시 올라가게 된다.

두 번째 특징인 No Virtual DOM은 Vue에서 발생할 수 있는 overhead를 방지해준다. 기본적으로 Virtual DOM을 사용하게 되면 이전의 Virtual DOM과 업데이트된 Virtual DOM 등 메모리를 더 차지 할 수밖에 없어 overhead가 발생한다. 하지만 Virtual DOM을 사용하지 않는다면 그만큼 메모리영역을 덜 차지한다는 말이 된다.

세 번째 특징인 Truly reactive는 svelte의 문법에서 나타난다. 다른 프레임워크인 Vue.js나 React.js는 변수에 변화가 일어났을 때 발생시킬 동작에 대해서 미리 프레임워크의 api를 통해 감싸서 작성을 해야 한다. 하지만 svelte에서는 이러한 반응형 동작을 $: 문법을 통해 간단히 작성할 수 있다. 즉 순수 자바스크립트 코드에서 거의 벗어나지 않아 자연스러운 코드 작성이 가능하다.

Ⅲ. 성능 비교를 위한 실험

1. 실험 Context

웹 페이지의 사용자 입장에서 가장 크게 체감되는 요소인 첫 로딩과 페이지 갱신 시 DOM Tree가 얼마나 빨리 업데이트 되는지에 대한 요소를 측정 할 것이다. 서버에 요청했을 때 응답이 와서 처리하는 시간은 실험 시 네트워크 환경이나 프레임워크 외적인 요소가 많은 영향을 미치므로 배제하도록 한다. 첫 로딩과 업데이트 소요시간의 측정을 위해 동일한 기능을 하는 웹 페이지를 Vue.js와 Svelte.js로 두 가지의 유형으로 제작하였다.

첫 번째 유형의 웹 페이지[4]는 첫 렌더링 성능을 위주로 테스트 할 수 있도록 하는 웹 페이지를 제작하였다. 웹 페이지에는 Header, Footer, Nav, Body 요소가 있고, Body 내부에 100만개의 div Element를 생성한다. Vue.js에서는 v-for를, svelte에서는 for each 구문을 이용해 Object 100만개가 들어있는 array를 적용하였다. DOM Tree 업데이트의 경우 버튼을 클릭하면 array 내부의 데이터를 모두 변경하여 100만개의 element 모두 업데이트가 발생하도록 한다. 결과를 측정할 때, 첫 로딩은 Chrome의 performance tab의 새로 고침 후 로딩 성능 기록하기를 활용하여 측정하였다. 업데이트의 경우 Vue.js는 nextTick hook을, Svelte.js는 tick hook을 활용 하였다. 이 때 각각의 hook은 웹 페이지가 업데이트 되고 실제 DOM에 적용된 직후 실행하는 역할을 한다. 이를 이용해 버튼 클릭 시 바인딩 된 함수에서는 함수 실행 직후 timestamp를 기록한 후 각각의 hook에서 timestamp를 한번 더 기록한 뒤에 그 둘의 차를 이용해 업데이트에 소요된 시간을 측정하였다.

두 번째 유형의 웹페이지[5]는 실시간 업데이트를 관찰해 볼 수 있는 서버 모니터링 웹페이지를 제작하였다. 우선, Node.js의 서버 framework인 express를 이용해 서버를 만들고, socket 통신을 사용하여 1초 간격으로 현재 RAM과 CPU의 각 쓰레드 사용량 정보를 전송해준다. 웹페이지인 클라이언트 사이드에서는 서버에서 socket 통신을 통해 시스템 정보를 받을 때 마다 현재 상태를 보여주는 그래프를 업데이트 시켜주는 동작을 하게 하였다. 이 때 차트는 Chart.js 라이브러리를 사용했다. 결과를 측정할 때, first-paint와 DOM Content Load Timing을 측정하고 지속적으로 update가 이루어지므로 js heap 사용량 역시 로딩한 뒤 1초 간격으로 5번의 snapshot을 캡처한 뒤 평균 값을 도출하였다.

각각 웹 페이지는 Vue.js는 vue-cli에서 webpack 템플릿을 이용하여 제작하여 webpack을 이용해 build한다. Svelte는 아직 지원하는 cli가 없어 svelte 공식 github에서 clone 해온 프로젝트를 수정하여 제작하여 rollup을 이용해 build하게 된다. Build 과정을 거치면 .vue파일과 .svelte 파일들이 모두 js파일 및 css파일로 변환되어 결과가 나오게 된다. 이를 nginx-1.18.0을 이용해 로컬 서버로 실행 하였다.

2. 실험 환경

표 1. 실험 환경

CPU	AMD Ryzen5 PRO 4650G
RAM	16G
browser	Chrome 91.0.4472.77
OS	Window 10 64bit

실험을 진행한 환경은 표 1과 같다.

3. 실험 결과

1) 첫 번째 웹 페이지는 두 가지 측면에서 결과를 측정하였는데, 먼저 DOM Content Load와 First Paint timing은 다음과 같은 결과가 나왔다. 각 측정값의 단위는 ms이다.

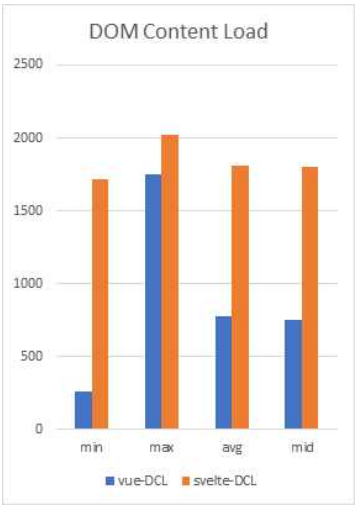

그림 1 리스트 페이지 DCL 비교 그래프

그림 1 그래프를 살펴보면 최대 DCL timing은 Vue, Svelte 두 프레임워크가 비슷한 양상을 보이고 있다. 최소, 평균, 중간값을 OM Content Load가 이루어지는 것을 볼 수 있다.

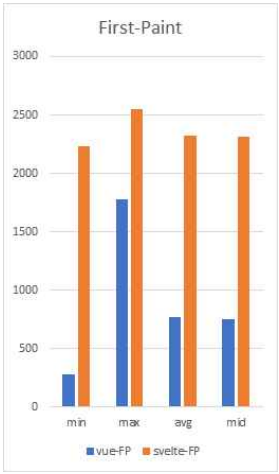

그림 2 리스트 페이지 FP 비교 그래프

그림 2 그래프를 살펴보면 FP 역시 DCL과 비슷한 양상을 보이고 있다. FP는 First Paint Timing의 값으로, 화면에 첫 픽셀이 그려지는 시간을 의미한다. 그림1과 그림2의 그래프 결과를 통해 첫 렌더링에는 Vue.js가 월등히 빠르다는 것을 알 수 있다.

DOM update에 소요되는 시간을 측정했을 때 결과는 그림 3에 나타난 그래프와 같다.

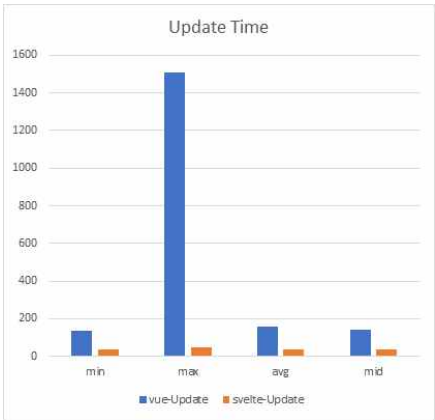

그림 3 리스트 페이지 update 시간 비교 그래프

그림 3에서 보다시피, svelte의 update시간은 그래프에서 보이지 않을 정도로 매우 적은 시간이 소요됨을 알 수 있다. 즉, 첫 번째 웹 페이지를 통해서는 Vue.js는 첫 렌더링

timing이 빠르고, Svelte.js는 DOM의 update 하는데 소요되는 시간이 매우 짧음을 알 수 있었다.

2) 두 번째 웹페이지에서는 실시간 서버 상태 모니터링을 구현하여, 첫 번째 웹페이지에서 측정했던 DCL, FP에 js heap를 추가적으로 측정하였다. 각 단위는 ms이다.

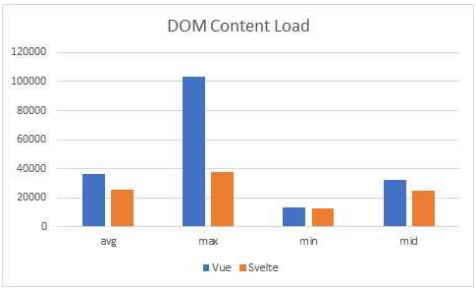

그림 4 모니터링 페이지 DCL 비교 그래프

모니터링을 위한 웹페이지에서는 첫 번째 웹페이지와는 상반되게, Vue.js에서의 DOM Content Load Timing이 더 늦게 나타나는 양상을 보여주었다.

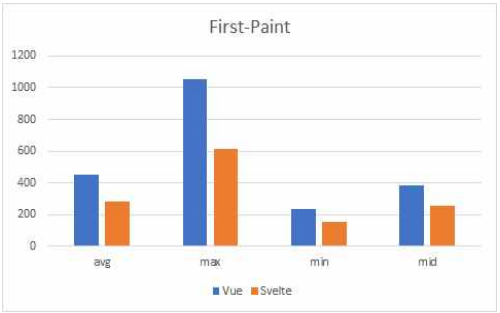

그림 5 모니터링 페이지 FP 비교 그래프

First-Paint의 Timing 역시 DOM Content Load Timing과 같이 Vue.js의 Timing이 더 늦는 것을 볼 수 있다.

웹 페이지가 렌더링 된 후 1초 간격으로 5회간 js heap의 수치를 측정하여 평균을 낸 결과는 그림 6의 그래프와 같다. 각 요소의 단위는 MB이다.

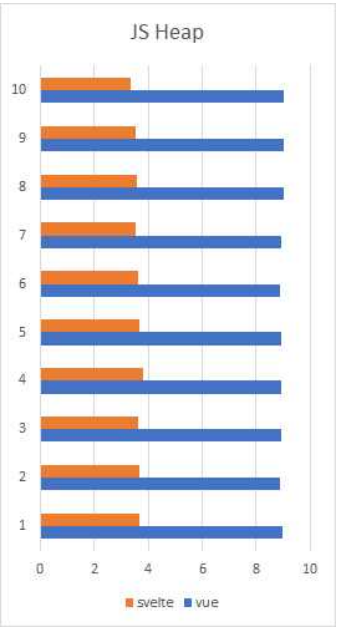

그림 6 모니터링 페이지 js heap 비교 그래프

그림 6에서의 그래프를 보면 svelte가 vue에 비해서 절반 이하의 js heap을 사용하고 있는 것을 알 수 있다. 이를 통해 전반적으로 Svelte.js가 Vue.js보다 상대적으로 가볍다는 것을 알 수 있다.

Ⅳ. 결론

1. 실험 결과 분석

1) Vue.js와 Svelte.js를 단순 반복요소로 구성된 웹 페이지와 실시간으로 DOM이 업데이트가 이루어지는 모니터링 웹 페이지 두 가지 유형으로 제작하여 첫 렌더링 타이밍과 js heap 사용량을 기준으로 성능을 평가하였을 때, 두 프레임 워크간의 차이는 확연하게 나타났다.

2) Vue.js는 단순 반복요소로 구성 된 웹 페이지에서 첫 렌더링에서 Svelte에 비해 더 빠른 렌더링 성능을 보여주었다. 반면 모니터링 웹페이지에서는 Vue.js가 모든 측면에서 상대적으로 저조한 성능을 보여주었다. 이를 통해 Vue.js의 observer pattern로 인해 사용되는 객체의 비용이 높아질수록 js heap의 사용량이 높아지는 것을 알

수 있다. 또한, 런타임 이전에 반응형이 결정되어 Vue로 구현된 웹페이지가 사용자에게 보이는 시간인 DOM Content-load timing과 First-Paint timing이 런타임 때 반응형이 결정되는 Svelte에 비해 더 빠를 것 같지만, 웹페이지의 구현 복잡도나 사용되는 객체들의 비용에 따라서 더 저조한 성능을 보일 수도 있다. 업데이트에 소요되는 시간 역시 observer pattern의 적용으로 각 객체에 더 많은 비용이 필요하고, 가장 큰 요인인 Virtual DOM의 재계산 시간으로 인해 더 오랜 시간이 소요됨을 알 수 있다.

3) Svelte.js는 Vue.js와는 다르게 런타임 시 반응형을 결정한다는 특징을 가지고 있다. Virtual DOM을 사용하지 않는 Svelte의 가장 큰 특징이 모니터링 페이지의 js heap 측정 결과에서 두드러지게 나타났다. 이로 인해 Vue.js에 비해 오버헤드가 적게 발생한다는 사실을 알 수 있다. js heap 뿐만 아니라 실제 업데이트에 소요된 시간 역시 Svelte.js가 두 웹페이지 모두 우수한 결과를 나타내고 있다.

2. 결론

1) Vue.js와 Svelte.js는 모두 구조와 문법이 간단해 러닝커브가 낮은 Node.js의 Single Page Application 프레임 워크이다. 이 덕분에 빠른 개발과 유지보수의 용이성이 중요한 스타트업 기업에서는 웹 애플리케이션 개발을 시작하기에 앞서 고려할 만한 프레임 워크들이다. 위의 실험에서 나온 결과로 보면 업데이트가 자주 일어나지 않는 정적인 웹페이지에서는 Vue.js가, DOM 요소의 갱신이 자주 발생하는 웹페이지에서는 Svelte.js가 조금 더 우위에 있다고 볼 수 있다.

2) 두 프레임워크에 각각 적합한 용도를 앞서 기술한 내용에 기반을 두어 추측해보면, Vue.js는 한 번 렌더링 되면 페이지 갱신이 잘 이루어지지 않는 쇼핑몰, 블로그 등 정적인 페이지가 적합할 것이다. 반면 Svelte.js는 잦은 갱신이 이루어지는 웹페이지들, 예를 들면 채팅을 위한 사이트나 실시간 모니터링 사이트 등에 적합할 것으로 보인다.

3) 하지만 두 프레임 워크가 각각 다른 장점을 가지고 있지만, 개발에 앞서서 해당 프레임워크의 개발커뮤니티가 얼마나 활성화 되었는지 여부나, 기존 개발자들이 능숙한 프레임워크가 있는 등의 다양한 변수도 고려해야 단순 성능을 넘어서서 개발에 더 적합한 프레임워크를 선정할 수 있을 것이다.

Reference

[1] Highest Voted 'reactjs' Questions - Stack Overflow(2021), Retrieved June., 04., 2021, from https://stackoverflow.com/questions/tagged/reactjs

[2] Introduction -Vue.js, Retrieved May.,05., 2021, from https://vuejs.org/v2/guide/

[3] Svelte • Cybernetically enhanced web apps, Retrieved March., 05, from https://svelte.dev/

[4] Sejin Eom, sejin1031/vue-svelte-comparison at list(2021), Retrieved June., 04., 2021, https://github.com/sejin1031/vue-svelte-comparison/tree/list

[5] Sejin Eom, sejin1031/vue-svelte-comparison at monitoring(2021), Retrieved June., 04., 2021, https://github.com/sejin1031/vue-svelte-comparison/tree/monitoring

엄 세 진 (Sejin-Eom)

2016년 3 월~현재: 한양대학교 소프트웨어학부 컴퓨터전공 학사 과정