웹사이트 로딩 시간 단축을 위한 프론트엔드 프레임워크 비교 분석

김영임*, 유헌창*
*고려대학교 컴퓨터정보통신대학원
e-mail : {springday, yuhc}@korea.ac.kr

Analysis of frontend framework for shortening web site loading time

YoungIm Kim*, HeonChang Yu*
*Dept. of Computer & Information Technology, Korea University

요 약

현재 다양한 성격을 가진, 무수히 많은 웹사이트들이 존재하고 있다. 웹사이트에 접속 시 또는 이벤트 실행 시 걸리는 로딩 시간은 사이트를 이용하는 사용자로 하여금 서비스의 질 문제를 야기시킬 수 있으며 나아가서는 비즈니스에도 큰 영향을 미칠 수 있다. 이러한 이유 등으로 브라우저들의 성능은 점점 좋아지고 있으며, 복잡해지는 프론트엔드 로직에 따라 다양한 환경에서의 빠른 개발 및 웹 페이지 성능 개선을 위한 각종 Javascript 프론트엔드 프레임워크들도 등장하고 있다. 비동기 통신 방법인 Ajax 기술을 사용하는 프레임워크를 통해 렌더링 처리 방안을 확인해보고, 속도 성능에 대한 비교 분석을 한다.

1. 서론

2018 년 기준 전 세계 웹 사이트의 수는 12 억 4 천만 개에 해당하며 약 41 억이 넘는 인구가 인터넷을 이용하고 있다. 웹 트래픽의 대부분은 모바일을 통해 유입되고 있으며 가장 빠른 성장을 보이는 인터넷 분야는 커뮤니케이션 및 소셜 미디어이다[1]. 소셜 미디어 분야의 성장과 더불어, 최근의 웹은 단순히 정적인 페이지를 보여주는 것뿐 아니라 하나의 커다란 어플리케이션으로서 동작하고 있다. 인터페이스를 동적으로 표현하기 위해서는 여러 상태들을 관리하고 수정해야 하는데, 페이지가 많아지고 데이터가 방대해질수록 이러한 요소들을 관리하는 것은 쉽지 않다.

이렇게 수없이 바뀌는 DOM 과 상태 값 등의 관리를 최소화하고 사용자 인터페이스 구현에 더 집중할 수 있도록 프론트엔드 라이브러리/프레임워크들이 생겨나고 있다. 이것들은 프론트엔드 개발자들이 코드를 체계적으로 관리하고 유지보수를 쉽게 할 수 있도록 도와준다는 장점 외에도, 특정 화면에 있어서 더 나은 사용자 경험을 제공해주고 각 프레임워크의 특성에 따라서는 렌더링 측면에서 빠른 성능을 보여주기도 한다.

페이지 로딩에 3 초 이상이 소요될 경우 사용자가 해당 페이지를 이탈할 확률이 높아진다는 연구 결과가 있듯이, 인터넷 속도는 전 세계적으로 빨라지고 있고 그에 대응하여 페이지의 최초 로딩 속도는 웹사이트 유입/이탈에 매우 중요한 요인 중 하나가 되고 있다.

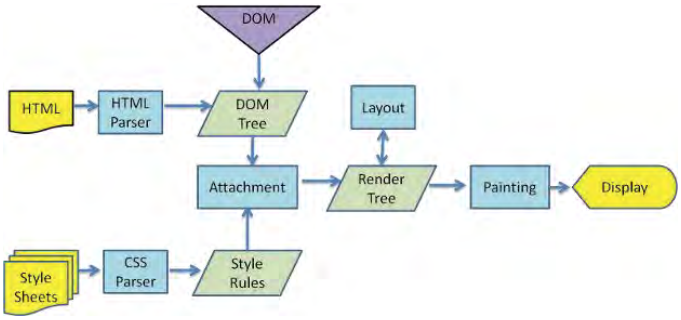
현재 사용 중인 프레임워크들 중 실무에서 가장 많이 사용되는 몇 가지를 골라 비동기로 동작하는 페이지를 구현하여 로딩 시간 단축에 얼마나 영향을 주는지 확인하고 페이지 콘텐츠의 종류와 양에 따라 어떠한 프레임워크를 사용하는 것이 로딩 시간 단축에 도움을 주는지에 대해 제안하고자 한다.

2. 관련 연구

2.1. DOM

DOM(the Document Object Model, DOM)이란 HTML 및 XML 문서를 처리하는 API 로, 구조화된 nodes 와 property, method 를 가지는 객체로서 문서를 표현한다. 각종 프로그램 언어는 DOM 구조에 접근함으로써 문서 구조, 스타일, 내용 등을 변경할 수 있다.

브라우저의 렌더링은 (그림 1)과 같은 방식을 거친다.



(그림 1) Chrome 브라우저의 렌더링[2]

(그림 1)의 상단에 있는 DOM 에 어떠한 변화가 나타나면 동기화(Synchronous) 작업을 통해 객체를 재생

성하여 Render Tree 를 다시 만들고 이는 곧 Layout, Painting 에도 영향을 준다. 변화가 없는 정적인 영역마저 재계산이 되어 브라우저의 연산 과정이 많아지게 됨으로써 결국 전체 페이지 로딩 속도는 느려지게 되고, 이러한 과정은 비효율적일수 밖에 없다[3].

2.2. Virtual DOM

페이지에 DOM 을 많이 조작함으로써 나타나는 비효율을 줄이기 위해 나타난 개념이 Virtual DOM 이라는 즉, DOM 의 복사본이다. 이는 가상에서 DOM 의 복사본을 만들어 놓고 변화가 있을 시 해당 부분을 비교한 이후 수정된 내용만 실제 DOM 에 반영하는 방식으로, 모든 DOM 을 다시 계산하지 않으므로 렌더링 속도가 훨씬 빠르다. 일반적으로 Virtual DOM 은 직접적으로 구현하기 어렵기 때문에 React 나 Vue 등의 프론트엔드 라이브러리/프레임워크를 통해 페이지의 퍼포먼스를 높이는 데 도움을 받는다[4].

2.3. Ajax

Ajax 는 Asynchronous Javascript and XML 의 약자로 웹 어플리케이션의 동작 시 비동기적인 통신 방법을 이용하는 개발 기법을 일컫는다. 웹 브라우저의 제어를 받지 않고 XML, XSLT, XMLHttpRequest(미리 정의된 HTML 이나 텍스트, JSON 등 사용) 객체 등을 이용하여 화면의 갱신없이 동적으로 데이터를 출력할 수 있으며 서버와의 데이터 교환 및 조작이 가능하다.
Ajax 를 지원하지 않는 브라우저가 있다는 점, 화면의 갱신 없는 데이터를 송수신하기 때문에 보안상에 치명적인 문제가 있을 수 있다는 점 등의 몇 가지 문제점이 존재함에도 불구하고 Ajax 를 사용하는 가장 큰 이유는 서버의 부담을 줄일 수 있고, 전체 초기화를 통한 데이터 전송이 아닌 핵심 데이터만을 전송하기 때문에 빠른 속도를 보장하기 때문이다[5].

2.4 Javascript

Javascript 는 프론트엔드 웹 개발에서 가장 널리 사용되는 프로그래밍 언어 중 하나이다. 일반적으로 클라이언트 측에서 HTML 및 CSS 와 함께 구현되어 상호보완적이고 독창적인 웹 페이지를 만들수 있도록 도와준다. Javascript 의 발전과 더불어 웹 애플리케이션의 개발을 보다 쉽고 효율적으로 하기 위해 Javascript Framework(이하 JSF)가 출현하게 된다. JSF 는 복잡하고 다양한 작업을 추상화하거나 일반화하고, 브라우저 간 지원과 호환성을 보장하며 이를 통하여 소프트웨어를 빠르게 개발할 수 있도록 한다. 또한 대규모의 웹 응용 프로그램을 개발할 경우 조직 내 코드의 재사용을 할 수 있어 개발자에게는 하나 이상의 JSF 는 필수라 할 수 있다[6].

2.5. 라이브러리/프레임워크의 종류

(1) Angular

2003 년부터 논의되기 시작한 SPA(Single Page Application: 서버로부터 새로운 페이지를 불러오지 않고 하나의 페이지를 동적으로 재생성함으로써 클라이언트와 소통하는 웹 애플리케이션)의 발전으로 잘 설계된 웹 페이지와 애플리케이션을 구현할 수 있는 프레임워크를 제공하고자 구글에서 Typescript 기반의 Angular 를 제안하였다. 슈퍼 MVW(Model-View-Whatever) Framework 라고도 불리는 Angular 는 HTML 을 확장시켜 다이나믹한 애플리케이션을 지원하며, 유지보수와 재사용성을 높임으로써 코드의 양을 줄여준다.

(2) React

이후 2013 년 페이스북에서 개발한 React 는 사용자 인터페이스 라이브러리로, 사용 가능한 UI 를 생성할 수 있도록 도와준다. Virtual DOM 이라는 개념을 사용하여 상태의 변화에 따라 선택적으로 화면을 렌더링하는, 최소한의 DOM 처리로 속도 개선 등 애플리케이션의 성능 향상에 도움을 주었다. 현재 Airbnb, Netflix, Twitter, Evernote 등 내로라하는 서비스들이 사용하고 있다.

(3) Vue

Vue 역시 사용자 인터페이스를 개발하기 위한 Progressive 프레임워크로, Angular, React 등 다른 프레임워크에 비하여 매우 가벼우며 복잡도가 낮다. 또한 별도의 변환 작업 없이 브라우저에서 바로 동작이 가능하다. React 와 같이 Virtual DOM 을 활용하였으며 DOM 조작에 가능한 적은 오버헤드만을 가하여 렌더링에 있어 다른 프레임워크에 비해 높은 성능 향상을 보인다.

3. 실험 환경 및 방법

3.1. 실험 환경

클라이언트인 브라우저와 웹 서버 사이에 적용되는 비동기 전송 방식인 Ajax 의 통신을 통한 응답시간 속도를 비교하기 위해 VanillaJS(순수 Javascript)와 대표적인 JSF 인 Angular, React, Vue 를 이용하였다.

<표 1> 성능 분석을 위한 환경설정

OS	Windows 10
CPU	Inter(R) Core(TM) i5-4690 CPU @ 3.50GHz
RAM	16GB
WAS	Node.js (node-java)
BROWSER	Chrome(M61+)
OPEN SOURCE	JS Framework Benchmark
FRAMEWORKS	Angular v6.1.0, React v16.4.1, Vue v2.5.16
TEST TOOL	Chrome DevTool (Network)

렌더링 속도 성능 평가를 위해 <표 1>과 같은 테스트 개발 환경을 구축하였다. 사용되는 OS 환경은

Window10 이며, 테스트 브라우저로는 Chrome(M61+)을 이용하였다. 각 프레임워크들의 성능 비교를 위해서 krausest 가 제작한 오픈소스인 "JS Framework Benchmark"[7]의 일부를 수정하여 연산 시간을 측정하였다. 페이스북이나 트위터와 같은 소셜 미디어와 비슷한 환경을 구성하기 위해 TEXT 와 이미지 데이터를 추가, 변경하였다.

3.2. 프레임워크별 실험 방법

페이지 로딩 속도 측정을 위해 TEXT 및 이미지 정보가 들어있는 JSON 을 만들고, 해당 데이터를 Ajax 로 가져와 랜덤으로 테이블을 생성하여 렌더링 지속 시간을 포함한 여러 시간을 측정한다. <표 2>의 시간 측정 내용을 각 1,000 개, 10,000 개씩 불러와 100 회씩 반복하여 실행한 후 Median 값(단위: ms)을 사용하였다.

<표 2> 시간 측정 내용

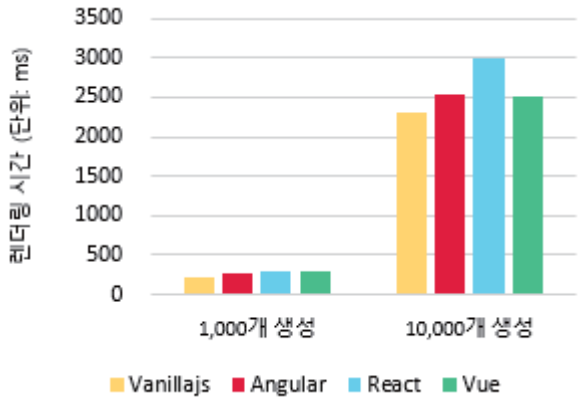
Create rows	페이지가 로드된 이후 일정량의 데이터를 생성하기 위한 지속시간
Edit/Update rows	데이터를 모두, 혹은 부분적으로 다른 콘텐츠로 변경했을 때 걸리는 시간. (5 회 반복)
Remove rows	데이터를 제거하는데 걸리는 시간
Memory	각 1,000 개의 데이터에 대하여, Ready: 페이지 로드 후 메모리 사용량 Run: 행을 추가한 후 메모리 사용량 Replace: 5 번씩 데이터를 대체한 후 메모리 사용량 Repeated clear: 5 번씩 데이터를 반복 삭제한 뒤의 메모리사용량
Script	Startup: Javascript 코드를 로드 및 파싱한 뒤 페이지를 렌더링하는데 걸리는 시간 Script bootup: 스크립트를 구문 분석, 컴파일, 평가하는데 필요한 시간

4. 성능 평가 및 분석

이 장에서는 데이터를 생성, 수정, 삭제하였을 때 걸리는 시간 및 해당 작업을 수행하였을 경우의 메모리 할당량, Script 렌더링 및 컴파일 시간을 분석한 결과를 기술한다.

(1) 데이터 생성

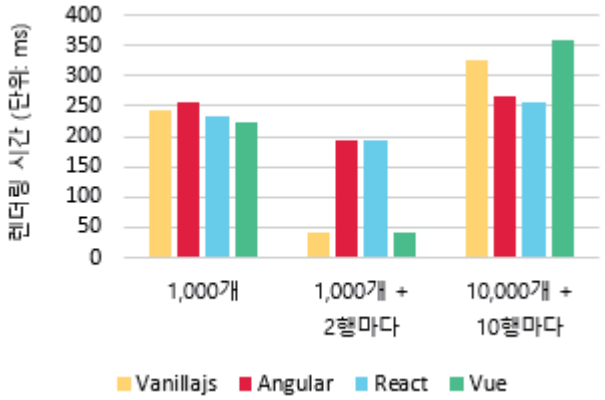
(그림 2)는 각각 1,000 개와 10,000 개의 데이터를 생성했을 경우의 로딩 속도를 보여주고 있다. 1,000 개의 데이터 생성시 대부분 비슷했지만 가장 빠른 것은 275.25ms 로 Angular 였다. 10,000 개의 데이터를 불러온 결과로는 Angular 와 Vue 가 각각 25.29s, 25.13s 로 거의 비슷했지만 Vue 가 0.16s 더 빨랐다. 반면, React 는 Vue 에 비해 1.2 배 느린 렌더링 속도를 보였다.



(그림 2) 데이터 생성 테스트

(2) 데이터 수정 및 업데이트

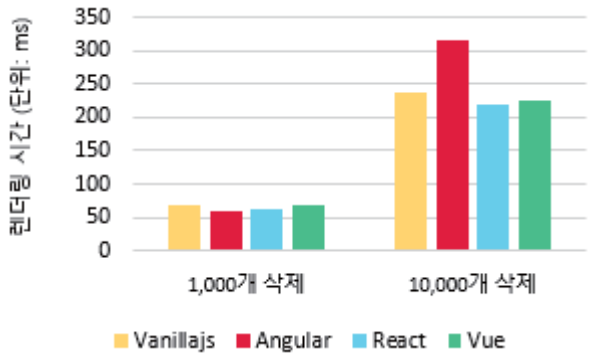
(그림 3)에서 보는 것과 같이 기존 1,000 개의 데이터를 수정하는 작업에서는 Vue 가 224.3ms 로 좋은 성능을 보였다. 특히, 1,000 개의 행에서 2 행마다 500 개 업데이트를 수행했을 때 가장 느린 Angular 에 비해 4.6 배 빠른 속도를 보였다. Vue 와 React 는 반복적으로 업데이트가 일어나는 콘텐츠에 Virtual DOM 을 적용함으로써 렌더링 시간을 절약하여 좋은 성능을 보여준다. 하지만 1,000 개 이상의 데이터를 수정한 경우, Vue 의 속도는 500 번 수정한 경우에 비해 평균 62.1%가 느려지는 것을 확인할 수 있었다. 반면, Angular 나 React 의 경우에는 데이터의 양에 관계없이 안정적인 속도로 출력되었다.



(그림 3) 데이터 수정 및 업데이트 테스트

(3) 데이터 삭제

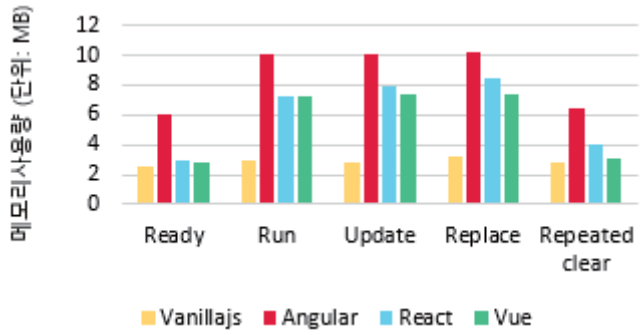
(그림 4)는 데이터를 삭제하는데 걸리는 시간을 측정한 그래프이다. 1,000 개의 데이터 삭제하였을 경우 모두 비슷한 속도를 보이는 가운데 Angular 가 58.83ms 로 가장 빨랐지만, 10,000 개의 데이터를 삭제했을 경우 315.14ms 로 다른 JSF(평균 222.04ms)에 비해 현저히 느려지는 것을 확인하였다.



(그림 4) 데이터 삭제 테스트

(4) 메모리 할당량

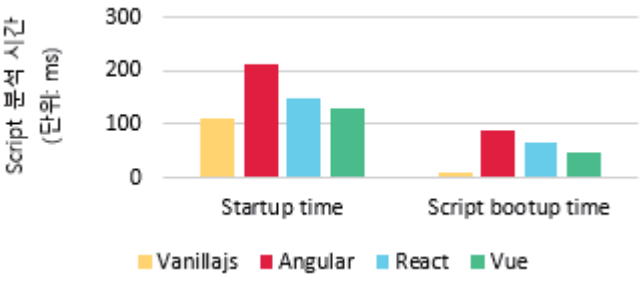
(그림 5)는 1,000 개의 데이터를 생성 및 업데이트 등의 작업 수행 시 각 JSF 의 메모리 할당량을 나타낸 그래프이다. 이 중 최적의 성능을 보이는 것은 평균 5.5MB 의 메모리 사용량을 가진 Vue 였다. Angular 가 평균 8.5MB 로 가장 많이 할당되었고, React 의 경우 6.0MB 의 수치를 보였다.



(그림 5) 메모리 할당량 (단위: MB)

(5) Script 렌더링/ 분석 시간

각 JSF 별 Javascript 엔진의 특성과 용량이 다르기 때문에 많은 양의 Javascript 를 가질 수록 더 무거운 오버헤드와 긴 로딩 시간이 측정된다. (그림 6)은 1,000 개의 데이터를 가져왔을 때 Script 를 분석하는데 걸리는 시간을 측정, 비교한 그래프이다. Angular 가 Version 6 으로 업그레이드되면서 Javascript 의 사이즈가 많이 작아졌지만 가장 빠른 Vue 에 비해 평균 1.5 배 많은 분석 및 컴파일 시간이 걸렸다.



(그림 6) Script 렌더링 및 분석/컴파일 시간 테스트

5. 결론 및 향후 연구

본 논문에서는 초기 데이터가 많지 않고 지속적인 업데이트 및 추가 생성을 하는 소셜 미디어와 같은 웹 사이트에서 좀 더 빠른 화면을 보기 위해 Chrome 브라우저로 비교 평가한 결과 Vue 를 사용하는 것이 가장 적합하다는 분석이 나온다. Vue 는 작업 수행 시 메모리를 가장 적게 사용하고 용량이 작아 대용량의 콘텐츠를 가지지 않는, 빠르게 움직이는 사이트에서 좋은 성능을 낸다.

Angular 도 좋은 방안이 될 수 있지만 자체적으로 사이즈가 크며 메모리 할당량이 많아 대규모 애플리케이션을 제작하는데 적합하다는 결론이 나온다. 또한, React 의 경우 데이터 양에 관계없이 안정적인 렌더링 속도를 보였기 때문에 작업 갱신이 많이 일어나는 큰 프로젝트에 적합하다.

향후 연구로는 TEXT + 이미지뿐 아니라 더 다양한 콘텐츠 (동영상, Layout, Paint 등 DOM 의 재계산에 영향을 주는 CSS 로 구현한 애니메이션 효과 등)에 따라 JSF 를 구현해 보았을 때 속도 측면에서 어떤 영향을 미치는지에 대한 평가와 다양한 브라우저(Firefox, IE Edge 등) 및 네트워크 환경에서의 렌더링 속도에 대한 분석을 진행할 것이다.

참고문헌

[1] Internet Trends 2018. Stats & Facts in the U.S. and Worldwide, https://www.vpnmentor.com/blog/vital-internet-trends

[2] How Browsers Work: Behind the scenes of modern web browsers, https://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

[3] MDN web Docs, Introduction to the DOM, https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

[4] What is Virtual Dom? https://medium.com/@tony_freed/what-is-virtual-dom-c0ec6d6a925c

[5] Ajax, https://ko.wikipedia.org/wiki/Ajax, 2018.09.

[6] Carl Lawrence Mariano, “Benchmarking Javascript Frameworks” https://arrow.dit.ie/cgi/viewcontent.cgi?article=1100&context=scschcomdis, 2017.01.

[7] krausest, js-framework-benchmark, https://github.com/krausest/js-framework-benchmark