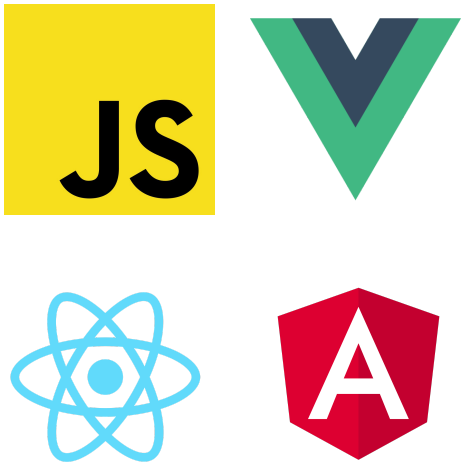

Pros and cons of Front-end JavaScript Frameworks

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

Dennis Skoko
privat@dennisskoko.se

Abstract

Context

JavaScript, the scripting language for web browsers is so advanced that web server can send the all of the resources needed for the web application to the user. When the user then navigates or interacts with the application, the data is then fetched from the server and injected into the application without reloading the page or the application. This is called SPA short for single-page applications. This will reduce the load on web servers with high traffic by avoiding transferring the same resources to the users.

But developing single-page applications is not always easy. So developers created several frameworks that would let other developers create web applications with ease. But when solving problems, new ones are introduced.

Objectives

Every time some technology is introduced, developers needs to find out what are the strengths and weaknesses of that technology. To find out, the problems that has been solved and are unsolved will be investigated and listed. The features, benefits and drawback of the technology will be filtered to only answer how it benefits both the users and developers.

Keywords

JavaScript, Front-end JavaScript frameworks, Single-page applications

Table of Contents

Introduction

JavaScript is a scripting language that is executed at the browser on web applications. It is used to make the web application interactive and more. Over the years JavaScript has become more advanced. One of the features is AJAX, this lets browsers fetch information from the server and dynamically add it into the web page. [14]

This allowed the developers to create web servers that sends the whole application to the user and from there the application can fetch data from the servers and inject the data to the page. This then resulted in the application would never refresh or re render anything whenever the user clicks on a new page within the application. Making the application into a single-page application (SPA). [14]

Another way to make web application is using the multi-page application (MPA). Whenever a user enters the application, only the resources for that page is sent and rendered. When a user navigates to a new page, the browser make a new request to the server. Then the server sends the resources for that page and gets rendered again. Some of the data between pages are often shared and is sent in every request as well. This results in the same data is sent over and over again to the user. [14]

But each way has its strength and weaknesses. By learning the strengths and weaknesses of front-end JavaScript framework to create a single page application is key to know when it’s best to use it. [14]

Research Questions

How do the users benefit from front-end JavaScript frameworks?

The most important group that should benefit are the users. They are the priority and if they don’t benefit then why would the developers spends hours and money into this.

How do the developers benefit from front-end JavaScript frameworks?

How does the developers get affected? It is more complicated than other technologies or is it easier? These things decide how much money the companies would need to spend and this is needed to decide if the cost is worth the benefits.

What problems can occur when using front-end JavaScript frameworks?

Sometimes problems can’t be solved and this can sometimes hurt either the users or the developers. Knowing the weaknesses from this technology can also help the companies decide if they should spend time and money into this.

Method

Most of the sources are on the internet and was only taken if it contains any information that answers a research question. Some sources that has been taken from the creators of frameworks. Some has been taken from big and trusted companies like Mozilla. The rest are articles from unknown people. The information from these unknown people is only taken if multiple people shares the same information. The information will be analyzed and discussed how it answers the research questions.

The answers to the research questions is different for each framework out there. That is why three most popular frameworks (most GitHub starts) has been selected. The features that will be brought up will at least contains in these three frameworks. The three frameworks are: React, Vue.js and Angular. [1]

Also note that some features may not always be included within the framework but has some community modules that adds the feature that is missing. An example is the router for react.

Analysis and Discussion

User Experience

One normal use case is when a user goes to the home page of a website and then decides to click on the about page button. What happens then is that the browser makes a new request to the web server and fetches the same resources (like images, stylesheets) and the HTML. Most of the resources are cached but the browser still needs to make requests to check if the cached resource is outdated. All of these requests takes bandwidth and especially time if the network is slow. [2, 13]

But when creating a single page application, the whole application is in one page. That means the web server sends the page and the resources to the browser, the browser renders the page and when the user click on about page button. It will show the about page without reloading the page. This is because JavaScript is used to remove the content from the home page and then add the content from the about page. If the page requires data from the server like staff members, the applications sends an AJAX request to the web servers API and gets only the data and nothing else. The application then parses the data and injects it on the page. In this way, the loading time is reduced for users with slow internet. [2, 13]

One drawback from rendering the application at the client side is that performance is moved from the server to the client. Especially when the application gets bigger. Most users has powerful desktop computers or laptops that has no problem handling these applications. But the case is not the same for mobile phones. Mobile phones are not as strong as computers

so they will take more time when rendering the applications. This issue however will be solved when using server-side rendering that will be brought up later. [2, 13]

Developing Process

Hot Reload

Front-end JavaScript frameworks has been around for some time now. Over the years, developers has find new ways to improve the developing process. This could be important, especially for projects with limited time. [4, 5, 6]

The developers of the frameworks has a feature included with the name *hot reload*. This lets developers see their change on the website without changing the actual state. This might not sound interesting at first but it is revolutionary for developers. [4, 5, 6]

Let's say for example, we are developing a counter. When someone visits the site then a number is shown. Then every second it increments the number by one, starting from zero. Then if we change the increment value by five instead and save our change to the file. Then without needing to do anything, the counter in the browser starts to increment by five instead of one without resetting the number value to zero again. [4, 5, 6]

This might be a small feature for the example but for big projects, it can save a lot of time for the developers. [4, 5, 6]

Components

Using components are not only restricted to front-end JavaScript frameworks but implementing components without it is not easy. [7, 8]

One thing that all developers share is that they want to create reusable code. Using the component structure will allow developers to create reusable code with ease. [7, 8]

A component is a bundle of HTML, a stylesheet (CSS) and logic (JavaScript). The stylesheet has its own scope within the page that doesn’t collide with the other stylesheets if the component is injected into multiple places of the page. It also comes with its own logic that can also be highly customizable. This will let developers create highly rich components into one module that can easily be injected into the application for other developers to use. [7, 8]

Less coupled with the API

When developing a single-page application, the interface is separate from the API. The server must provide the raw data (eg. staff members, posts or users) so the web application that is sent to the user can access these data so it can be displayed. The web application is just an interface. [3, 13]

Because of that, it allows developers to easily create a single API on the server and then be able to create multiple interfaces that accesses that data from that API. This will help

developers a lot when creating an application that is going to work on the web, mobile phones and desktops. [3, 13]
Nowadays, JavaScript is available everywhere and if an interface is created for the web, the same code can be used to deploy on desktops and on mobile phones. [3, 13]

Production Deployment

When deploying the traditional server-side rendered application, it can sometimes be cumbersome. Some projects in one language doesn't work on all servers like for example Node.js on Apache. [2, 13]

But when creating a single page application, all of the files are bundled into fewer files that can be easily uploaded on any static content server like Apache, Nginx, Amazon S3 or Firebase Hosting. But this depends on whether you will need to build an RESTful API for the application. [2, 13]

Search Engine Optimization (SEO)

Distinguish Pages

One problem was since the web application was in one page, web crawlers could not separate the pages from each other. This was because when moving from the home page to another didn't actually change the URL since the page had been modified with JavaScript. [9]

These problems occurred when Google released AngularJS in 2009 and so they tried to solve it by making a proposal to make AJAX pages crawlable. But it didn't always work and then it got deprecated in 2015. [10]

Since Twitter relied on the AJAX technology, they came up with a idea for this. They came up with separating the different pages with the 'hash bang'. This meant that a URL with two different pages could look like this: [9]

http://example.com/#/
http://example.com/#/about

This solved the problem and even got recommended by Google but it still wasn't the solution for the problem since it didn't follow the W3C Standards. So developers came up with another idea and it was the HTML5 History API that lets JavaScript change the URL without reloading the page. This meant that the URL of two different pages could look like this: [9]

http://example.com/
http://example.com/about

Just like a normal web application will look like. But a small problem came with this idea. It required some configuration for the server so when a user refreshes, instead of showing a

404 Not Found error, it would fallback to the web application instead. This idea followed the W3C standards and it became the solution. [9]

Content Analyzing

Another problem with creating single-page applications is having your web application analyzed by web crawlers. Since the web page is rendered on the browser, the page you get without executing the JavaScript is empty. Usually the HTML from a single page application looks like this. [9]

```
<!doctype>
<html>
  <head>
    <title>Application</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <div id="app"></div>
    <script src="main.js"></script>
  </body>
</html>
```

Since there is no content then nothing will show up on the search results. So Google decided that their web crawler should execute the JavaScript and then analyze it. This will solve this problem but not entirely. If the web application fetches content for the page asynchronously then the web crawler will not wait for the content. [11]

Server-Side Rendering (SSR)

Since the web crawlers does not wait for the web applications to fetch content asynchronously then another approach is needed. So when the web crawlers can't render the pages properly and those pages needs to be rendered on the server. After it has rendered the page, it will then send the result to the user. [12]

After the result has been sent, the result will act like a single page application again. So when the user navigates through the application, it will not reload. This way when web crawlers visits the different pages, they will receive the rendered page. This solves the problem and it is the only solution for now, but it has its drawbacks. [12]

JavaScript is the only script language for web browsers, this means the frameworks are written with JavaScript. In order to make SSR work, servers needs to execute JavaScript. [12]

This is no problem with creating servers with Node.js but if something else is used like PHP or Java, then the only way to make it work is to run a JavaScript runtime inside the language

like PHP V8 or Java JavaScript Runtime. Problem with those is that they come with bad performance and is not easy to implement. [12]

Result / Conclusion

How do the users benefit from front-end JavaScript frameworks?

The users benefit because the response time between navigating links is much faster and reduces bandwidth by sending the application to the browser and the fetch the raw data from the servers only when required.

How do the developers benefit from front-end JavaScript frameworks?

Developers benefits as well since it's easier to build reusable highly customizable components that they can use to build interfaces quick. Since developers see their change fast without changing the state within the application during development also lets them build applications faster.

Since the application gets separated from the server, this allows developers to create different applications in various devices like desktop or mobile phones. So if a project that needs to target other devices, using a front-end JavaScript framework would save some money.

What problems can occur when using front-end JavaScript frameworks?

Even if front-end JavaScript frameworks are really good, it still has its drawbacks. If the project demands a programming language other than JavaScript on the server then SEO will be hard or even impossible depending on the project. If the project requires a blog where SEO is important then it is impossible right now to make it work if Node.js is not used on the server. For some application like Google Drive or Dropbox do not require SEO and if the project is similar to those then using a front-end JavaScript framework is best suited for that project.

But it is not impossible to be able to get the best of two worlds. Some companies have their website running as a *hybrid web application*. Combining both single-page and multi-page to create an application that uses single-page technology where SEO is not important, then the rest is rendered on the server-side just like a multi-page application. This is not always ideal because some of the application will be coupled to the server while the rest is loose from the server, making the application in two different places. This will not be easy to maintain and will not be easy for the developers to develop or debug if the application will need to grow in the future.

Future Work

One could dive deeper into how using a front-end JavaScript framework benefits the developers and how it lets them build applications faster with the features it provides.

Right now the SSR solution is not perfect and maybe the developers will find another solution that is better. The authors of the frameworks and the community will always update and add new features. Single-page applications has grown a lot since it came out and will continue to grow.

References

[1] Front-end JavaScript frameworks
https://github.com/showcases/front-end-javascript-frameworks
Accessed: 3/10 2017

[2] Why a Single Page Application, What are the Benefits? What is a SPA?
https://blog.angular-university.io/why-a-single-page-application-what-are-the-benefits-what-is-a-spa/
Accessed: 3/10 2017

[3] Single-page application vs. multiple-page application
https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
Accessed: 3/10 2017

[4] Angular 2 Hot Loader
http://blog.mgechev.com/2015/10/26/angular2-hot-loader-hot-loading-tooling/
Accessed: 3/10 2017

[5] What About Separation of Concerns
https://vuejs.org/v2/guide/single-file-components.html#What-About-Separation-of-Concerns
Accessed: 3/10 2017

[6] Hot Reloading in React
https://medium.com/@dan_abramov/hot-reloading-in-react-1140438583bf
Accessed: 3/10 2017

[7] Modular CSS with React
https://medium.com/@pioul/modular-css-with-react-61638ae9ea3e
Accessed: 3/10 2017

[8] Introduction - webcomponents.org
https://www.webcomponents.org/introduction
Accessed: 3/10 2017

[9] Single Page Application (SPA) and the SEO Problem
https://adkgroup.com/insights/single-page-applications-spa-and-seo-problem
Accessed: 3/10 2017

[10] Deprecating our AJAX crawling scheme
https://webmasters.googleblog.com/2015/10/deprecating-our-ajax-crawling-scheme.html
Accessed: 3/10 2017

[11] GET, POST, and safely surfacing more of the web
https://webmasters.googleblog.com/2011/11/get-post-and-safely-surfacing-more-of.html
Accessed: 3/10 2017

[12] Introduction - ssr.vuejs.org
https://ssr.vuejs.org/en/
Accessed: 3/10 2017

[13] SPA vs MPA, What Are the Pros and Cons?
https://insanelab.com/blog/web-development/spa-vs-mpa-single-multi-page-application-pros-cons/
Accessed: 19/10 2017

[14] A re-introduction to JavaScript
https://developer.mozilla.org/sv-SE/docs/Web/JavaScript/A_re-introduction_to_JavaScript
Accessed: 7/11 2017