Available online at : http://www.advancedscientificjournal.com
http://www.krishmapublication.com
IJMASRI, Vol. 3, issue 1, pp. 910 - 915, Jan. -2023
https://doi.org/10.53633/ijmasri

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY ADVANCED SCIENTIFIC RESEARCH AND INNOVATION (IJMASRI)

ISSN: 2582-9130 **IBI IMPACT FACTOR 1.5** **DOI: 10.53633/IJMASRI**

RESEARCH ARTICLE

SINGLE PAGE APPLICATION AND SERVER SIDE RENDERING A SPA

Kartik Gupta
Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, India

Abstract

Traditional applications used to be pretty straight forward, multiple clients connected over the internet made HTTP requests to servers which in turn returned static content back to the client. Each URL used to correspond to a path on the server which eventually returned that asset. But with the growth of internet technologies & libraries, frameworks and protocols, a website is not just a static asset hosted on a server that serves the file over HTTP. Websites are more of web applications now, which make thousands of requests per second to not just one but multiple servers. Websites have millions of assets, which need to be loaded in a matter of milliseconds, and with the increasing server load due to traffic a traditional multi page application would ideally bottleneck if not optimised. This is where modern single page applications come into play, SPA is composed of individual page that can be updated independently on each user's action, so that the entire page does not need to be reloaded like classical web application. This, in turn, helps to increase the levels of interactivity, responsiveness and user satisfaction. But it's not that simple, routing that used to be simple task in a traditional application is now a hassle, because we just have one page to deal with. The assets that used to be loaded over a span of user's activities on your application need to be loaded at once, otherwise the application won't function. These are all valid reasons, and that's the reason libraries like ReactJS exist. That is to make life easier using a single page application.

Keywords: SSR, React JS, Next JS, HTML, Single Page Application

Introduction

SPA (Single-page application) which is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs such as XML Http Request and Fetch when different content is to be shown. This therefore allows users to use websites without loading whole new pages from the server, which can result in performance gains and a more dynamic experience, with some trade off disadvantages such as SEO, more effort required to

maintain state, implement navigation, and do meaningful performance monitoring.

Literature Survey

ReactJS and frontend development:

Use of react is primary for the front-end technologies and for the creation Web applications. The front end frameworks React and Angular were compared and React was definitely a better choice. React was discussed in details and its advantages and disadvantages were listed too. The criteria for selection of specific framework were clearly identified and react should be chosen in most of the cases.

ReactJS and frontend development:

Use of react is primary for the front end technologies and for the creation Web applications.The front end frameworks React and Angular were compared and React was definitely a better choice. React was discussed in details and its advantages and disadvantages were listed too. The criteria for selection of specific framework were clearly identified and react should be chosen in most of the cases.

Performance enhancements in ReactJS:

React is one of the popular web frameworks that has gained importance over other frameworks such as Angular, Vue, etc. This is because of its implementation of Virtual DOM; whose primary objective is to enhance the overall performance of the application. However, there are certain things that one has to keep in mind before designing the applications. Failing to anticipate the problems that may occur component hierarchy will lead to performance degradation. Some of the commonly faced problems are component re-rendering, application lag due to background computations being run, lag due to processing large data sets in a single stretch, etc. This paper will describe some of the practical ways of overcoming such problems within the application, thus enhancing the performance of the ReactJS App in a production environment. The paper will also describe a time-efficient search algorithm that can be used for searching objects in a large data set.

ReactJS and Node:

Node.js, is a JavaScript runtime built on Chrome's V8 JavaScript engine. It allows developers to run JavaScript on the server side, enabling them to build server-side applications with JavaScript. Node.js provides an event-driven, non-blocking I/O model, making it well-suited for real-time, data-intensive applications that run across distributed devices. It is often used for building back-end services, such as APIs, and command-line tools.

Problem Statement

Single-page application) which is a web app implementation that loads only a single web document, and then updates the body content of that single document via JavaScript APIs such as XML Http Request and Fetch when different content is to be shown.

Therefore allows users to use websites without loading whole new pages from the server, which can result in performance gains and a more dynamic experience, with some trade off disadvantages such as SEO, more effort required to maintain state, implement navigation, and do meaningful performance monitoring.

Advantages of a React SPA:

Speed:

We load a single page application just once, there’s no reload that happens. So ideally on subsequent loads and navigation a SPA is much faster than your traditional multi page application.

Virtual DOM:

The virtual DOM (VDOM) is a programming concept where an ideal, or “virtual”, representation of a UI is kept in memory and synced with the “real” DOM by a library such as React DOM. This process is called reconciliation. Virtual DOM allows devs to

update even the smallest components of the app without influencing its entirety. Virtual DOM is nothing else than a virtual copy of the real DOM. Additionally, it requires less time to update than a regular DOM which is a huge advantage. This approach enables the declarative API of React: You tell React what state you want the UI to be in, and it makes sure the DOM matches that state. This abstracts out the attribute manipulation, event handling, and manual DOM updating that you would otherwise have to use to build your app.

Reusable Components:

In a single page application, we're just dealing with a single page, and most SPA implementing libraries like react give us the ability to create reusable UI elements that can be reused, added and removed to the application.

One-way data flow:

Usually, frameworks feature a two-way data flow and it means that whenever you make a change, an entire application also changes or requires a change. When it comes to React though, it features a one-way data flow that ensures code stability. Thanks to it, any change made in a particular section doesn't influence the structure of the entire app. That essentially means that if you alter child components it doesn't affect the parent data in any way.

Proposed Design System

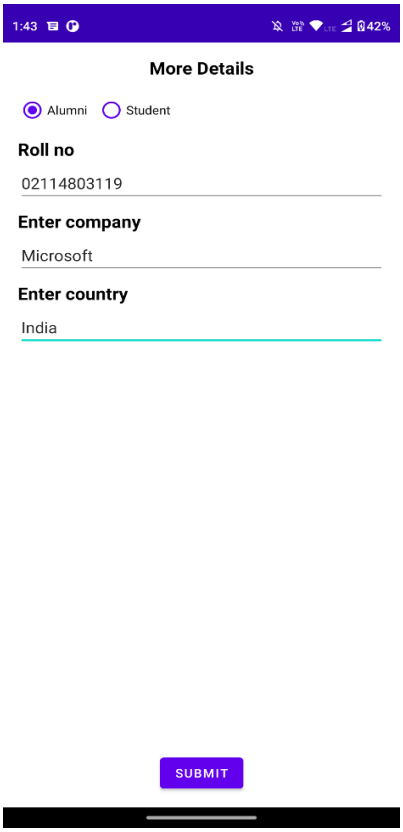
Server-side Rendering

If an application is as fast as an SPA, as good in SEO as a multi-page application and paints fast. That might be the solution to web applications and their struggles.

This is the problem server-side rendering solves. Server-side rendering means using a server to generate HTML from JavaScript modules in response to a URL request. That's in contrast to client-side rendering, which uses the browser to create HTML using the DOM.

Server-side rendering with JavaScript works similarly to other server-side languages such as PHP or .NET, but with Node.js as the runtime environment. When the server receives a request, it parses the JavaScript modules and data required to generate a response, and returns a rendered HTML page to the browser. Single-page applications use client-side rendering. All URL requests are redirected to the same bare-bones HTML document, like the example that follows.

Client-side rendering fills in the rest. Views for specific URLs are managed by a JavaScript routing mechanism. Each URL request triggers a DOM update instead of a network request. As a result, sites that use client-side rendering can feel "snappier" and more responsive to user actions. However, client-side rendering has two significant drawbacks. Site visitors have to wait for the JavaScript bundle to load and for the browser to build the DOM before any content is visible. They may see a blank page or loading image while JavaScript loads.



You are bare bones HTML document lacks the keyword, description, and social media metadata (e.g. Open Graph) necessary for search engine optimization and social media sharing.

Server-side rendering addresses both concerns by creating HTML at run time, when the server receives a browser request). Search engines can index your URLs. Visitors can share them on Facebook or Twitter. In theory server-side rendering is a great way to enhance a SPA, and eliminate its drawbacks but in application it is relatively complex to implement. Especially in a closed bubble that Create React App is, because of its no override babel/webpack policy.

But there are frameworks like Next.JS by vercel, Gatsby JS etc, that implement SPA out of the box, and make it really easy to server side render a SPA and eliminate its drawbacks.

Advantages of server-side rending

Server-side rendering (SSR) in React can provide several benefits, including:

Faster initial load times: When a user makes a request to a server-rendered React application, the server sends back a fully rendered HTML page, which the browser can display immediately. This can result in faster load times, especially for users with slower internet connections.

Improved SEO: Search engines have a difficult time crawling and indexing JavaScript-heavy applications. By rendering the initial HTML on the server, server-side rendering can make it easier for search engines to understand and index the content of a React application.

Better performance on low-end devices: JavaScript-heavy applications can be slow to load and run on low-end devices with limited resources. Server-side rendering can help mitigate this issue by reducing the amount of JavaScript that needs to be loaded and executed on the client side.

Better user experience: A faster initial load time and better performance on low-end devices can result in a better overall user experience.

Better security: SSR can help mitigate some types of cross-site scripting (XSS) attacks by rendering user-generated content on the server before it is sent to the browser.

It is important to note that server-side rendering can add complexity to the development process and may not be suitable for all React applications. Developers should evaluate the trade-offs and decide if SSR is the best approach for their specific use case.

Next JS and SDR

Next.js is a framework built on top of React that makes it easy to add server-side rendering (SSR) to a React application. Some of the key ways that Next.js aids SSR include:

Automatic code splitting: Next.js automatically splits your code into small chunks, so that the browser only needs to load the code that is needed for the current page. This can help to improve the initial load time of your application.

Server side rendering out of the box: Next.js provides a built-in server that can handle rendering your React components on the server. This means that you don't need to set up your own server or configure server-side rendering manually.

Dynomic imports: Next.js allows you to use dynamic imports, which can be used to lazy-load parts of your application. This can help to improve the load time of your application by only loading the code that is needed for a specific page.

Hot code reloading: Next.js supports hot code reloading, which allows you to see changes to your code without having to refresh the page. This can make development faster and more efficient.

Routing: Next.js has a built-in routing system that can handle client-side and server-side routing out of the box.

Custom server: Next.js allows you to create a custom server if you want to handle server-side rendering and other server-side tasks in your own way, instead of using the built-in server.

Overall, Next.js makes it easy to add server-side rendering to a React application, providing a set of features and functionalities that are tailored to improve the performance and development experience of SSR.

Disadvantages of SSR

Server-side rendering (SSR) in React can provide several benefits but it also has some disadvantages, including:

Incresed complexity

Server-side rendering can add complexity to the development process, especially if you are not familiar with server-side programming. This can make it more difficult to debug and maintain your application.

Incresed server load

Server-side rendering requires the server to do more work, which can increase the load on the server and make it more difficult to scale your application.

More difficult to implement

Setting up server-side rendering can be more difficult than just using client-side rendering. This can be especially true for developers who are not familiar with server-side programming or the specific server-side rendering library or framework being used.

More difficult to test

Testing server-side rendered code can be more difficult than testing client-side code. This is because the code needs to be run in a Node.js environment and may require a more complex test setup.

Longer development time

Implementing server-side rendering can take longer than just using client-side rendering, especially if you are not familiar with the process.

Longer build time

Building an SSR application can take longer than building a client-side only application, because the server needs to pre-render the pages.

It is important to note that server-side rendering may not be suitable for all React applications, and developers should evaluate the trade-offs and decide if SSR is the best approach for their specific use case.

Conclusion and future scope

React, Next.JS now days is one of the most popularly used JavaScript libraries and it is being used by almost everyone. It is also one of the most starred repositories with a huge contributor base. Modern React API, coupled with JSX is one of the most efficient and agile way to create an application. But SPA's even after being server side rendered might still have their drawbacks like Frequent server requests, An overall slow page rendering, Full page reloads, Non-rich site interactions.

So, at the end using traditional multi page application or a Single page application or a Server side rendered Single Page application is a critical architectural and performance question, which needs to be taken early in the stages of application development, with each having their pros and cons.

Reference

1. Performance comparison (2019) – angular vs react vs vue.js, https://blog.logrocket.com/angular-vsreact-vs-

vue-a-performance-comparison/Angular Vs React-A
2. Comparison(2017)http://work.haufegroup.io/Angu lar-VS-React/ Complete comparison guide (2020)
3. https://medium.com/front-end-weekly/react-vsangular-vs-vue-js-a-complete-comparison-guided16faa185d61
4. https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d

*****