Historical, Philosophical and Linguistic Studies

https://www.harrisonacademicpress.com/index.php/HPL

Vol 1, Issue 1, 2026

A COMPREHENSIVE GUIDE TO IMPROVE ANGULAR APPLICATIONS PERFORMANCE THROUGH SERVER-SIDE RENDERING (SSR)

Banda Saikumar
American Airlines Inc, USA
saikumar.banda@gmail.com

Abstract– ***The research emphasises the way Angular Universal brings in a different approach to improving performance and SEO in Angular applications with the use of Server-Side Rendering (SSR). The page load times are significantly improved, thus improving FCP and TTI with SSR. Improvement is witnessed through better indexing of search engines and more exposure and visibility of the content as for SEO. It solves state management challenges by applying lazy loading, pre-rendering and state transfer to ensure seamless integrations between client and server. The optimum best practices involve caching, pre-rendering, and efficient treatment of states. The trend for the future can be that performances can increase along with SEO and developments in web technology, including integrating HTTP/2 and service workers at a larger scale.***

Keywords: ***State Management, Server-Side Rendering, Best Practices, Angular Universal, SEO***

I. Introduction

Performance is a significant component that influences user engagement and retention in current web development. Most SPAs that include most Angular applications-suffer from poor loading time that are not SEO-friendly. Problems are solved by SSR, doing its job right on the server rather than on the client. This makes the load time of the very first page load much quicker, improving the performance of SEO. Angular Universal is the official solution to Angular, enabling SSR to let the system serve its pages effectively from the server [1]. The effect of applying SSR to an Angular application using Angular Universal is discussed with respect to the performance and optimisation of the search engine.

II. Aims and Objective

The main aim of this study is to assess the influence of Server-Side Rendering (SSR) using Angular Universal on performance and SEO in Angular apps.

- To investigate best practices for using SSR with Angular Universal to improve Angular application speed and user experience.
- To identify and overcome the issues associated with the implementation of SSR in Angular apps, such as state management and server-side rendering complications.
- To assess the influence of Server-Side Rendering (SSR) on performance indicators such as Time to First Paint and Time to Interactive.
- To assess the SEO advantages of adopting SSR in Angular apps, with an emphasis on increased search engine indexing and content exposure.

III. Research Questions

- What are the best practices for integrating SSR with Angular Universal to improve Angular application performance and user experience?
- What challenges occur during the deployment of SSR in Angular apps, particularly those involving state management and server-side rendering complications, and how can they be addressed?
- What influence does Server-Side Rendering (SSR) have on important performance indicators like Time to First Paint and Time to Interactive?
- What SEO benefits can SSR offer for Angular apps, specifically in terms of boosting search engine indexing and content visibility?

IV. Research rationale

The problem in Angular application development involves performance constraints on SPAs. Most SPAs suffer from their slow loading times and bad SEO performance. These are very adverse that hamper user engagement and limit the visibility of the search engine. Slowing down the loading of pages reduces user experience, translating to a high bounce rate. It's hard for search engines to index dynamic content for better discoverability. Challenges are addressed by SSR that renders content on the server, thereby improving both performance and SEO [2]. This research explores the way SSR can be used to optimise Angular applications and provide a solution for these common issues.

V. Literature Review

Best Practices for Implementing SSR with Angular Universal to improve Application Speed and User Experience

Implementing Server-Side Rendering (SSR) with Angular Universal needs following best practices to provide maximum speed and user experience. A number of good practices, such as by lazy loading the modules-meaning, load in the time of required-help decrease an application's loading time. Those can generally be useful techniques that have been considered for enhancing perceived performance. The benefit offered by Angular Universal is due to dividing the application into smaller modules to prioritise rendering important components for better page load [3]. Other best practices for this are to do with server-side caching whereby caching the already rendered HTML content at the server itself saves lots of headaches getting it rendered for the same page many times. This speeds up not only response times but also reduces the number of requests to the server, thereby improving its scalability.

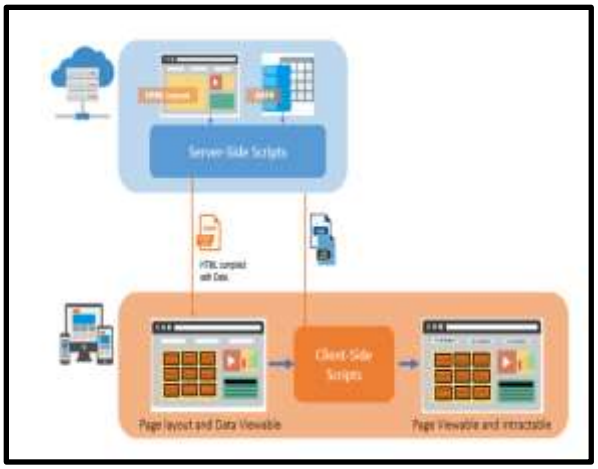


Fig 1: Server-Side Rendering (SSR)

Users can serve cached contents much faster and speed up page loading and another helpful technique in this regard is pre-rendering. Pre-rendering basically lets a developer statically generate HTML at the build time for certain routes [4]. This is very useful for content that rarely changes, while the user gets faster page loads without having to rely on server-side rendering for each request. Another important aspect of SSR is the state transfer between the server and the client. Angular Universal supports the transferring of application state from the server to the client. The client-side app takes off where the server-side rendering can have left off. Proper state management ensures that a seamless transition can be effected sans glitches like superfluous re-renders.

Challenges in Implementing SSR in Angular Applications and Solutions for State Management and Server-Side Rendering

Most of the challenges related to SSR in Angular applications concern state management and the process of rendering itself. One basic challenge is that of state management. The application state can be properly synchronized between the server and the client in functional server-side rendering [5]. Most Angular applications rely on various client-side services or APIs to manage state, ensuring this state transfer is properly done during SSR can prove quite challenging. The client-side app can fail to pick up where the server-side rendering left off, leading to inconsistent behaviour in the time of there is no proper mechanism for state transfer.

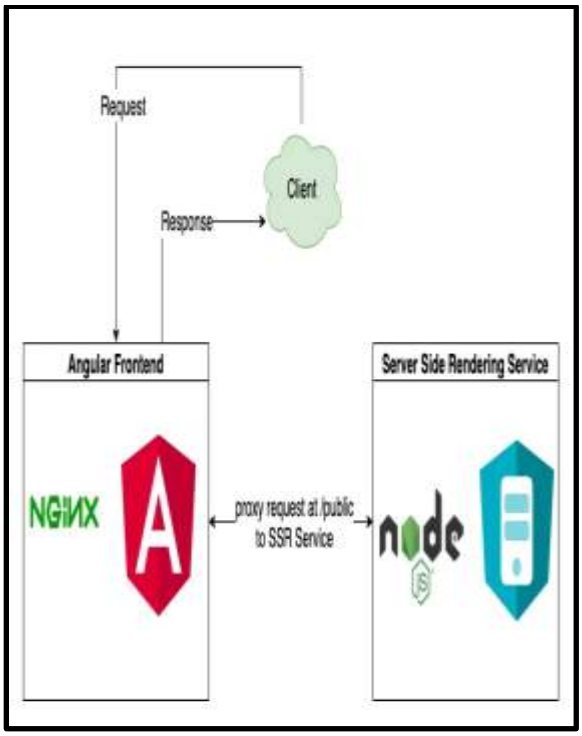


Fig 2: Server-Side Rendering (SSR) in Angular Universal

Complications can occur with server-side rendering and Angular applications are targeted for client-side execution, complications can arise in the time an application needs to be set up for SSR. Components and services relying on browser-specific APIs cannot work on the server. For example, functionalities that have to do with manipulating the window or document cannot be executed on the server. This is left to the developer to make sure works by implementing workarounds or conditional logic. Complications can increase development time and complexity. Other challenges can be mitigated by the application of state transfer techniques made available through Angular Universal [6]. It makes an application's state serializable on the server enabling seamless hydration at the client. One can be allowed to employ Universal-friendly API respect to Browser-dependent code for better handling at the client and server-side.

Influence of Server-Side Rendering on Performance Metrics such as Time to First Paint and Time to Interactive

Most of the important performance metrics, including Time to First Paint (FCP) and Time to Interactive (TTI), are considerably improved by SSR. Time to First Paint (FCP) is explained by the fact that SSR makes sure a fully rendered HTML is returned from the server to the client on the very first request. Performance metrics can be combined with lazy loading for even more optimisation [7]. The user has to wait for JavaScript download and execution before the application becomes visible with client-side rendering. This rapid, or immediate, feedback that the user gets enhances a user's overall perceived experience by the reduction of perceivable load time.

SSR also impacts Time to Interactive besides improving FCP and the HTML content is there almost instantaneously, and the client can start making the page interactive with the SSR. The execution of client-side JavaScript can happen parallel to the user already interacting with the page that has been rendered in the meantime [8]. The application reaches a full interactive state sooner that reflects on the whole performance metrics. It loads those resources that are actually needed, whereas the rest of the components get loaded afterwards.

SEO Benefits of Adopting SSR in Angular Applications for Enhanced Search Engine Indexing and Content Visibility

The use of Server-Side Rendering in Angular applications can help a lot in improving SEO issues greatly, providing better indexing for search engines and making website content more perceivable. Classic SPAs are not friendly for search crawlers because of their reliance on client-side rendering. Search engines cannot index dynamic content generated by JavaScript that reduces web page findability [9]. The challenge is SSR that delivers complete HTML to the client browser. This makes it much easier to crawl and index by whatever search engine's crawlers or spiders.

The initial HTML delivered to the client already contains all the relevant content that search engines are bound to index such a page more accurately due to SSR. This readily available content helps improve search engine rankings. The search engines have something to index-complete content of the web page can start getting ranked much higher in results. Server-Side Rendering (SSR) can improve click-through rates since users are presented with completely rendered results in search that general user experience is likely to increase [10]. It does a much better job in the time it comes to dealing with meta tags and dynamic content that is deemed crucial for better Search Engine Optimisation. Meta tags relating to titles, descriptions and keywords are considered critical in terms of ranking and categorising content.

VI. Methodology

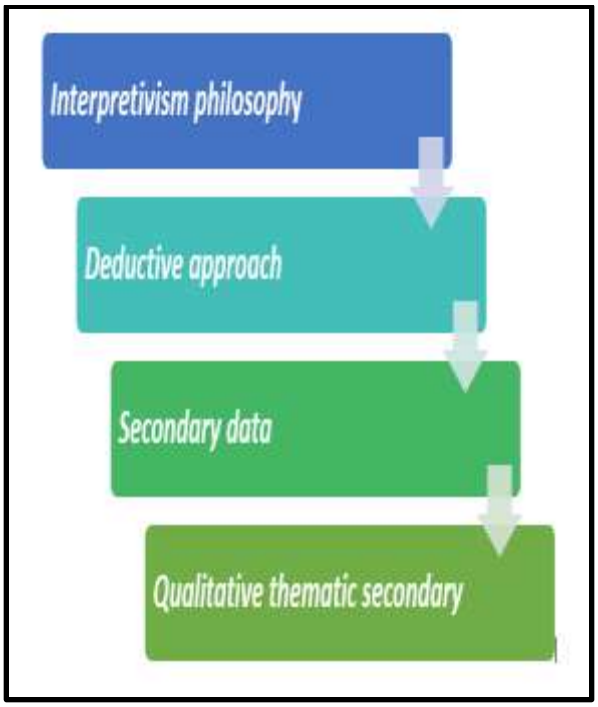


Fig 2: Methodology of the Research

This research follows an interpretivism philosophy, a deductive approach, and utilizes secondary data with qualitative thematic analysis. Each one of them is selected in a manner that enables the research to explore and interpret in detail the impact SSR causes on the performance of Angular applications in relation to SEO. ***Interpretivismphilosophy*** focuses on meaning, human behaviour, and common phenomena. Interpretivism can help explore both the subjective experience that developers hold and interpret more accessible data relating to the implications of SSR [11]. It is actually not to investigate the way much the performances have risen but, instead, to try to ascertain the agents and practices causing this very success in Angular apps. The approach is important in helping the research understand the way existing practices, studies and challenges interpret SSR.

The ***Deductive approach*** tests the pre-existing theory or hypotheses and the research is based on the way SSR improves metrics such as FCP and SEO in Angular applications. The deductive approach adopted in this study is deductive, where the researcher tries to confirm or refute ideas presented in the existing literature [12]. This can be a step-by-step examination of exactly the way SSR can affect those particular areas through secondary data sources. ***Secondary data*** are preferred as they tend to be of the highest value for information related to the implementation and outcome of SSR.

Secondary sources also include published research papers, industry reports, case studies, and technical documentation, where there can be comprehensive analyses without necessarily having to collect the data from the field. These are easily available and give insights into the various project's implementations, performance benchmarks of SSR and its subsequent outcomes on SEO performance. Secondary data can be practical to collect, most especially on such issues as SSR because so much study and reporting have been done. The secondary data can be analysed by using ***Qualitative thematic secondary*** data analysis. An approach can outline patterns and themes in textual data that can enable the researcher to gain necessary knowledge from literature and case studies. ***Thematic analysis*** as a method can allow broad insight into the issues of SSR affecting performance and SEO, not relying on numerical data only [13]. It is expected that the research can be in a position to provide detailed information on challenges, solutions and best practices related to SSR in Angular applications by analysing qualitative data.

VII. Data Analysis

Theme 1: Best Practices are examined to determine successful techniques for combining SSR with Angular Universal to increase application performance and user experience.

It is to revisit the best practices to implement successful techniques in the process of integrating SSR with Angular universal for improved application performance and enhanced user experience. Lazy loading is a technique that an application loads only that can be needed on the initial load [14]. Translating reduces the general quantum of JavaScript results in a much speedy application for any end user using it. Lazy loading also optimises the performance both on the server and client side, minimising useless data transfers and allowing an application to focus resources on critical resources.

Another great strategy that works in performance is the pre-rendering approach. It generates static HTML for a set of routes at build time, so that opening the page it's ready to serve. This pre-rendering can be very effective, especially for static or less dynamic content, as the content hardly changes. It reduces the server's load by not having to render the pages every time there is a request to the server. Another best practice in SSR is state transfer. Angular Universal enables the application state to be transferred from the server to the client, the client-side application can continue from where the server-side rendering stops [15]. Proper management of state can prevent issues related

to re-rendering or loss of data at the client side, resulting in a very smooth user experience.

Theme 2: The challenges are discussed with an emphasis on difficulties such as state management and server-side rendering complexities experienced during SSR implementation in Angular apps.

Challenges are discussed with an emphasis on difficulties such as state management and server-side rendering complexities experienced during SSR implementation in Angular apps. The biggest challenge is the way to handle the state. Angular applications depend on some client-side services and APIs as far as keeping the state of an application is considered [16]. It is necessary that the application state can be rightly synchronized between the server and client while doing SSR. The client-side application cannot be able to continue with the page served by the server without a good mechanism of state transfer. This causes inconsistent behavior or problems in user experience.

The application on the client side can fail to continue from the page the server has rendered without a good mechanism of state transfer. A big challenge is the complexity of the server rendering since by design Angular applications are supposed to run at the client-side [17]. Some browser-specific features and APIs, like window and document, do not work on the server. Hydration can be a bit problematic and the client has to "take over" the page after the server renders it for the first time by hydrating the HTML with JavaScript. First render happens to differ from the server-rendered page, discrepancies and visual glitches can occur. This cannot happen upon proper synchronisation and state management for seamless transitions from SSR into client-side rendering.

Theme 3: Performance impact is evaluated, especially the way SSR affects critical metrics such as Time to First Paint and Time to Interactive.

The SSR performs great regarding Time to First Paint since it sends fully rendered HTML from the server. It measures performance impact and especially the way well SSR does on critical metrics such as Time to First Paint and Time to Interactive. It reduces the time the user has to see something on the screen, hence improving perceived performance. Client-side rendering requires the browser to download and execute JavaScript before content rendering that delays what is shown to the user versus FCP [18]. The content is pre-rendered on the server that means the user can see content sooner with server-side rendering. Another point of impact SSR has on TTI is that it makes the page so much more interactive and a lot faster. The initial HTML content is rendered on the server, the browser can run JavaScript concurrently with the user interacting with the already-rendered content. That means interaction can be way faster compared to client-side rendered applications that have to wait for JavaScript to load and execute before the page is interactive. Improved TTI directly enhances the user experience by reducing wait time. The technology can be able to introduce a few other famous performance optimization techniques out there, including lazy loading and pre-rendering [19]. This is one way of speeding these FCPs and TTIs by introducing a small amount of JavaScript on the starting page. Faster and wider load times-and, making resource loading for optimisations of Angular can create impressive responsive instances and engage many users for extra time with SSR in place.

Theme 4: SEO benefits are explored, with a focus on the way SSR adoption enhances search engine indexing, content exposure, and overall search visibility for Angular apps.

SEO benefits are explored in ways SSR adoption increases the indexing of search engines, exposure of content, and general visibility for Angular applications. SSR enables search engines to crawl and index fully rendered HTML content coming from a server with ease. Most search engines find it hard to index dynamic content that needs the execution of JavaScript in client-side rendering. It has already displayed the content that is more accessible to search engine crawlers in the case of SSR by the time it has loaded the page. The visibility of content is highly enhanced because all Meta tags, titles, and descriptions are immediately crawled and indexed by search engines in SSR [20]. The page becomes more relevant for various search algorithms and takes a higher rank in SERPs. SSR has a direct influence on the process of search engine optimisation by embedding critical metadata into initial server-rendered HTML. Another important area where SSR provides benefits is in search visibility. Web applications are more likely to appear in relevant searches with search engines now able to index complete pages much better [21]. This improved indexing increases the likelihood of better organic rankings that bring more traffic to the website. This also helps with rich snippets since the structured data and metadata can already be there for the search engines to interpret.

VIII. Future Directions

Angular applications keep optimising performance and further improve SEO outcomes in the future. Angular Universal can integrate with more advanced features in server-side rendering as web technologies continue to evolve. Angular Universal can lead to even more fluid and responsive web applications, even for complex dynamic content [22]. Another promising development can happen at the integration point with the emergent standards of the web. SSR frameworks are also finding a way to utilise those features for far better performance as newer technologies like HTTP/2 and service workers get more widespread adoption.

IX. Conclusion

The above data concludes SSR boosts performance and SEO of an application in Angular. The performances include improved page load times for faster and superior interactivity on the SEO side, superior indexing, and superior exposure of your content, and high rankings in search engines. Most of the challenges, including state management and several rendering complexities, are tackled through techniques like lazy loading, pre-rendering, and state transfer. Best practices can optimise SSR integration including caching and state management. Future directions show even more optimisation with developing web technologies.

References

[1] Chandler, T., Shroff, H., Oldenbourg, R. and La Rivière, P., 2020. Spatio-angular fluorescence microscopy III. Constrained angular diffusion, polarized excitation, and high-NA imaging. *Journal of the Optical Society of America. A, Optics, image science, and vision*, *37*(9), p.1465.

[2] Jartarghar, H.A., Salanke, G.R., AR, A.K., Sharvani, G.S. and Dalali, S., 2022. React apps with Server-Side rendering: Next. js. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, *14*(4), pp.25-29.

[3] Araujo, I.F., Park, D.K., Petruccione, F. and da Silva, A.J., 2021. A divide-and-conquer algorithm for quantum state preparation. *Scientific reports*, *11*(1), p.6329.

[4] Ballamudi, V.K.R., Lal, K., Desamsetti, H. and Dekkati, S., 2021. Getting Started Modern Web Development with Next. js: An Indispensable React Framework. *Digitalization & Sustainability Review*, *1*(1), pp.1-11.

[5] Ioana, A. and Korodi, A., 2020. Improving OPC UA publish-subscribe mechanism over UDP with synchronization algorithm and multithreading broker application. *Sensors*, *20*(19), p.5591.

[6] Lim, Y.W., Tan, W.S., Ho, K.L., Mariatulqabtiah, A.R., Abu Kasim, N.H., Abd. Rahman, N., Wong, T.W. and Chee, C.F., 2022. Challenges and complications of poly (lactic-co-glycolic acid)-based long-acting drug product development. *Pharmaceutics*, *14*(3), p.614.

[7] Aslanpour, M.S., Gill, S.S. and Toosi, A.N., 2020. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, *12*, p.100273.

[8] Iskandar, T.F., Lubis, M., Kusumasari, T.F. and Lubis, A.R., 2020, May. Comparison between client-side and server-side rendering in the web development. In *IOP Conference Series: Materials Science and Engineering* (Vol. 801, No. 1, p. 012136). IOP Publishing.

[9] Samarasinghe, N. and Mannan, M., 2021. On cloaking behaviors of malicious websites. *Computers & Security*, *101*, p.102114.

[10] Nordström, C. and Dixelius, A., 2023. Comparisons of Server-side Rendering and Client-side Rendering for Web Pages.

[11] Pervin, N. and Mokhtar, M., 2022. The interpretivist research paradigm: A subjective notion of a social context. *International Journal of Academic Research in Progressive Education and Development*, *11*(2), pp.419-428.

[12] Hall, J.R., Savas-Hall, S. and Shaw, E.H., 2023. A deductive approach to a systematic review of entrepreneurship literature. *Management Review Quarterly*, *73*(3), pp.987-1016.

[13] Teo, K.J.H., Teo, M.Y.K., Pisupati, A., Ong, R.S.R., Goh, C.K., Seah, C.H.X., Toh, Y.R., Burla, N., Koh, N.S.Y., Tay, K.T. and Ong, Y.T., 2022. Assessing professional identity formation (PIF) amongst medical students in Oncology and Palliative Medicine postings: a SEBA guided scoping review. *BMC Palliative Care*, *21*(1), p.200.

[14] Turcotte, A., Gokhale, S. and Tip, F., 2023, September. Increasing the Responsiveness of Web Applications by Introducing Lazy Loading. In *2023*

38th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 459-470). IEEE.

[15] Steiner, A.M., Lissel, F., Fery, A., Lauth, J. and Scheele, M., 2021. Prospects of coupled organic–inorganic nanostructures for charge and energy transfer applications. *Angewandte Chemie International Edition*, *60*(3), pp.1152-1175.

[16] Iskandar, T.F., Lubis, M., Kusumasari, T.F. and Lubis, A.R., 2020, May. Comparison between client-side and server-side rendering in the web development. In *IOP Conference Series: Materials Science and Engineering* (Vol. 801, No. 1, p. 012136). IOP Publishing.

[17] Boutsi, A.M., Ioannidis, C. and Verykokou, S., 2023. Multi-Resolution 3D Rendering for High-Performance Web AR. *Sensors*, *23*(15), p.6885.

[18] Iskandar, T.F., Lubis, M., Kusumasari, T.F. and Lubis, A.R., 2020, May. Comparison between client-side and server-side rendering in the web development. In *IOP Conference Series: Materials Science and Engineering* (Vol. 801, No. 1, p. 012136). IOP Publishing.

[19] Abdolrasol, M.G., Hussain, S.S., Ustun, T.S., Sarker, M.R., Hannan, M.A., Mohamed, R., Ali, J.A., Mekhilef, S. and Milad, A., 2021. Artificial neural networks based optimization techniques: A review. *Electronics*, *10*(21), p.2689.

[20] Kostagiolas, P., Strzelecki, A., Banou, C. and Lavranos, C., 2021. The impact of Google on discovering scholarly information: managing STM publishers' visibility in Google. *Collection and curation*, *40*(1), pp.1-8.

[21] Nagpal, M. and Petersen, J.A., 2021. Keyword selection strategies in search engine optimization: how relevant is relevance?.*Journal of retailing*, *97*(4), pp.746-763.

[22] Motevaselian, M.H. and Aluru, N.R., 2020. Universal reduction in dielectric response of confined fluids. *ACS nano*, *14*(10), pp.12761-12770.