



React, next.js, and node.js: A practical framework comparison for full-stack development

¹ Priti Sharma, ² Pramila Joshi, ³ Seema Sharma,

¹ Assistant Professor, Department of Computer Science and Application, Birla Institute of Technology Mesra Ranchi, Noida Campus, India

² Assistant Professor, Department of Computer Science and Application, Birla Institute of Technology Mesra Ranchi, Noida Campus, India

³ Assistant Professor, Department of Computer Science and Application, Birla Institute of Technology Mesra Ranchi, Noida Campus, India

Email - ¹priti@bitmesra.ac.in, ²pramila@bitmesra.ac.in, ³seema@bitmesra.ac.in

Abstract: JavaScript eco system has got several tools in its kitty including React.js and Next.js each offering their own unique services for front end development. In addition to React.js and Next.js which offer specific services for front end development JavaScript also has tools such as Node.js providing backbone for many modern web applications. This paper aims to evaluate these three emerging technologies based on their architecture, scalability, performance and user friendliness, to enable the service providers and clients to select the best fit as per their unique requirements.

Key Words: Next.js, React.js, Next.js, Node.js, web development, SSR, performance, scalability.

1. INTRODUCTION:

the web development world is constantly evolving and javascript is playing a major role in all the web development projects. correct selection and application of a java script can impact the success or failure of a particular project. the study of react.js and its successor next.js can easily establish this argument while react.js specifically uses component-based design and virtual dom, next.js improves upon the performance of react.js by utilising tools such as server-side rendering (ssr) and static site generation (ssg). these two technologies are becoming popular among developer due to the ease which they afford in fast loading seo-friendly applications [2][4]. this paper aims to evaluate these three emerging technologies based on their architecture, scalability, performance and user friendliness, to enable the service providers and clients to select the best fit as per their unique requirements. introduction

in the constantly evolving world of web development selection of the right javascript can impact hugely on a project's success. this can easily be established by studying the two projects developed by facebook i.e. react.js [1] and its successor next.js [2] [4]. while react.js uses component-based design and the virtual dom, next.js adds on to react's strengths by making use of powerful tools like server-side rendering (ssr) and static site generation (ssg). ssr and ssg are gaining importance for developers depending upon fast loading, seo-friendly applications [2][4].

This paper aims at a comprehensive analysis of both the frameworks with a focus on their basic features, performance and usage, which in turn would assist the clients to make considered technological choices.

2. LITERATURE REVIEW:

2.1 React.js Overview

First introduced in 2013 React.js revolutionised how user interfaces were designed by introducing component-based architecture [1] [3]. This concept enabled developers to divide interfaces into reusable modules of codes. The Virtual DOM on the other hand optimizes performance by updating those parts of the page that have been modified instead of updating the entire page.



React also supports JSX, which blends HTML-like syntax with JavaScript, making it easier to structure interactive user interfaces. Create React App (CRA) on the other hand enables speedy setting up of projects with sensible defaults [5]. React's vast community and ecosystem makes available numerous resources permitting easier access to the developers to locate libraries and tools most suitable for their requirements [9].

2.2 Next.js Introduction

Vercel created Next.js as an advanced framework of React.js by adding features that enhance web application performance and user experience [2] [4]. It incorporates features such as server-side rendering (SSR) and static site generation (SSG) by default, thereby providing more flexibility to the developers as to how the content is delivered. File-based routing system, not only makes routing easier but also reduces the need for manual configuration. In addition, backend API routes can be embedded directly into the project. Features such as automatic code splitting and incremental static regeneration (ISR) are making it attractive to companies which must constantly deal with high traffic and fast deliveries [2][6].

2.3 Node.js Overview

Node.js is a JavaScript runtime for server-side applications, based on Chrome's V8 JavaScript engine. It enables developers to execute JavaScript on the server, facilitating full-stack development with just one programming language. Its event-driven, non-blocking structure is especially ideal for creating scalable and high-performance applications. Node.js serves a crucial function in contemporary web development processes, driving development servers, build tools, and backend architectures. Node.js allows JavaScript to go beyond the browser, enabling developers to create a variety of projects, including RESTful APIs and real-time applications [10].

Main highlights comprise:

- Event-Driven Architecture: Effectively manages numerous simultaneous connections.
- V8 Engine: Transforms JavaScript straight into machine code for rapid execution.
- Multi-Platform: Operates on Windows, Linux, and macOS.
- Vast Package Ecosystem: More than one million libraries accessible through npm

3. Core Features

3.1 React.js

Front-end development is simplified by use of a wide variety of tools and concepts provided by React. **Virtual DOM**, one key feature of React enables faster user interfaces by updating only those parts of the web page which change, thereby enhancing the responsiveness of applications, as pages which remain unchanged are not affected [1].

The component-based architecture followed by React allows breaking down of UIs into smaller components that can be re used across the app. It facilitates better and easier maintenance and organisation of the codes as individual components manage their own data and logic.

JSX on the other hand allows writing of HTML like codes inside the JavaScript. As both the structure and logic can be seen at one place building user interfaces using JSX becomes more innovative [5] [9].

Any change in the app can be traced using tools for state management. For basic use, state can be managed within components while libraries like Redux or React Context help manage shared data across more complex apps.



3.2 Next.js

Next.js is the next step of React.js introducing a number of features enabling optimal, efficient and speedy development of goal-oriented websites in particular.

SSR is one of the major benefits of Next.js. It generates the HTML of a page on the server and sends it to browser so that the page loads faster and its performance is enhanced on search engines. In addition, the content can be viewed more quickly by the user even before the entire JavaScript is being loaded.

The other major accrued is SSG which creates HTML pages at build time instead of as and when requested by the user. Storage on CDN ensures instant loading of the pages. SSG is ideal for content that remains unchanged like blogs or documentation [2] [6].

Automatic code splitting ensures that it loads only that JavaScript which is required for the page being viewed by the user as a result initial load time is reduced and performance is enhanced particularly on large sites.

API routes on the other hand enables writing back-end logic i.e. form submissions for database queries directly inside a Next.js project. Thus, doing away with the need for a separate server thereby making development simpler and more integrated.

3.3 Fundamental Features of Node.js

Node.js is frequently utilized alongside both React.js and Next.js. Here are several key characteristics:

- Asynchronous I/O: Enhances throughput and responsiveness during high traffic.
- Modular System: Promotes clean coding practices and the reuse of modules and packages.
- Included Libraries: Contains modules for HTTP, file systems, streams, and additional functionalities.
- Package Management: npm (Node Package Manager) is crucial for handling project dependencies [11].

Node.js facilitates complete JavaScript development across the stack by supporting server frameworks like Express.js and Koa.js, which work effortlessly with React and Next.js on the frontend

4. Performance and Scalability

4.1 React.js

Though React builds smooth and responsive user interfaces, the performance of the interfaces however requires additional efforts from the programmers. The app, by default makes use of **client-side rendering**, where in the content is generated on the browser once the page loads. Even if fast interactions get benefitted by this feature, viewing of content on slow networks may get delayed due to this.

For optimal performance developers often rely on concepts like **memorization**—which restricts unnecessary renders—and exploits tools as **React.memo**, **PureComponent**, or **shouldComponentUpdate** to dictate how components should work. React Lazy and React Suspense facilitate lazy loading ensuring parts of the app load only when required [7].

React uses frameworks such as Next.js or custom setups with express.js making it flexible however it also requires manual configuration [1] [3].



4.2 Next.js

Performance optimization is the biggest highlight of Next.js. While SSR allows users access to a fully rendered page straight from the server without having to wait for the JavaScript to load before seeing the content, thereby enhancing user experience.

SSR on the other hand enables developers to prebuild pages at deployment time itself. The load time becomes incredibly fast as static files are delivered through CDNs. Next.js also supports Incremental **Static Regeneration (ISR)**, allowing specific pages to update doing away with the need to rebuild the entire site.

Automatic splitting of codes by default is another important feature. This entails that the Java Script required for the page under consideration is only loaded thereby making the system fast even for large scale applications. Next.js is thus an ideal choice for apps that need to scale and perform well under heavy traffic [2] [4].

4.3 Node.js: Backend Efficiency & Expandability

Node.js is highly effective at managing simultaneous requests and real-time information. Due to its asynchronous architecture, it can handle numerous simultaneous connections without blocking based on threads.

In applications where performance is critical, such as chat platforms, collaborative tools, or live dashboards, Node.js is remarkably effective. Its support for WebSockets and RESTful APIs makes it an ideal option for the backend of single-page applications (SPAs) developed with React or hybrid applications driven by Next.js [12].

5. Development Workflow and Ease of Use

5.1 React.js

5.1.1 Using Create React App (CRA)

CRA with its preconfigured essential tools and best practices enables speedy development of applications as the developers are relieved of the problems of setting up complex build systems. This frees them to focus on the primary task of writing codes and building features rather than concentrating on configuration details [5].

5.1.2 Custom Configuration Flexibility

In addition to allowing developers to focus on writing codes React also offers custom configurations allowing developers to fine tune their projects using tools like web pack and Babel adding specific plugins. The benefit of this feature is that it provides greater control however higher technical skill is needed for optimizing the performance [9].

5.1.3 Strong Ecosystem and Community

React's massive interactive community provides access to numerous libraries, UI frameworks and other resources. Whatever be the requirement in terms of components, debugging tools, performance optimization etc, there is a high probability of these being already available in the community. This enormous support network facilitates easier and faster access to help, learn best practices, and stay updated with the latest trends.

5.2 Next.js

5.2.1 Simple Setup with Zero Configuration

Easy start is the greatest advantage of Next.js. Built-in defaults allow developers to develop new projects speedily without getting involved with complex configurations. This simplicity is perfect for small projects or MVPs where time is of essence [2] [6].



5.2.2 File-Based Routing System

file-based routing system is another important feature of Next.js allowing developer to create files in the pages directory instead of writing route definitions manually. The routes are generated automatically making navigation intuitive. This in turn keeps the project structure organised [2].

5.2.3 Extensibility with Plugins

Plugins and extensions add powerful features to a project without cluttering the core code. Any specific requirement such as image optimization, internationalization, or advanced data handling might have a plugin to make the process simpler. This enables scaling up the app without adding unnecessary complexity [4] [6].

5.2.4 Active Community and Learning Resources

The ever-growing user base and robust community provides plugins, tutorials, and support as well as guides, examples and help forums allowing developers to troubleshoot problems, learn new techniques, and stay abreast with the framework's evolution. This vast ever evolving eco system is a major support for both new and experienced developers.

5.3 Node.js Development Process

5.3.1 Unified Tools for React and Next.js

React and Next.js both depend on Node.js for development:

- Create React App (CRA) employs Node.js to set up and operate the development server.
- Next.js utilizes Node.js for handling server-side rendering, static generation, and routing through its Node-driven server.

5.3.2 Creating APIs using Express.js

Node.js allows for straightforward development of RESTful APIs with frameworks such as Express.js. React frontends can utilize these APIs, or they can be incorporated straight into a Next.js application via its /api routes [13].

6. Use Cases and Industry Adoption

React.js is a natural choice for applications that demand high interactivity and complex user interfaces—like single-page apps, dashboards, and real-time platforms [1][5][9]. It's trusted by companies like Facebook, Netflix, and Dropbox, which speaks to its maturity and adaptability [3][9].

On the other hand, Next.js is often preferred for content-rich or SEO-driven websites such as blogs, marketing pages, and online stores. With its built-in features and support for hybrid rendering, it powers platforms like Nike, Vercel, and GitHub[4][6].

In general, React suits highly interactive front ends, while Next.js shines when fast delivery, scalability, and search visibility matter most.

6.1 Revised Use Cases and Sector Implementation

- Node.js + React: Perfect for real-time applications such as messaging apps, collaborative software, and tailored dashboards.
- Node.js + Next.js: Ideal for content-rich websites, online shopping sites, and hybrid-rendered web apps that gain advantages from SSR and API routes.



Firms such as Netflix, LinkedIn, and Uber utilize Node.js for their backend services, while GitHub, Nike, and Vercel integrate it with Next.js for efficient frontends.

7. Comparative Analysis

The table below outlines the key differences between React.js and Next.js across major development criteria:

Feature	React.js	Next.js	Node.js
Rendering	CSR	SSR	SSG Not Applicable
Routing	Manual	File-based	Using Express or Koa
SEO Assistance	Manual SSR	configuration Integrated or Built-in	Not relevant
Performance	Requires Adjustment or tuning	Enhanced	High for I/O-constrained
Setup Complexity	Medium	Low	Low
Backend Integration	Requires distinct	Integrated API routes	Integrated HTTP server
Use Cases	SPAs, dashboards	SEO sites, e-commerce	Real-time, APIs

8. CONCLUSION :

Deciding between React.js and Next.js ultimately depends on what the project demands. React works best when flexibility and a dynamic, interactive front-end are the top priorities. It's ideal for single-page applications that need real-time responsiveness and customized setups.

On the other hand, if you're aiming for faster page loads, better SEO, and built-in tools to manage both front-end and back-end in one place, Next.js is a strong choice. It's especially useful for content-rich or marketing-focused websites.

By understanding the unique strengths of each framework, developers can make better decisions and build web applications that are not just functional, but fast, scalable, and easy to maintain.

REFERENCES:

Web References:

1. <https://reactjs.org>
2. <https://nextjs.org/>
3. <https://github.com/facebook/react>
4. <https://github.com/vercel/next.js>
5. <https://flaviocopes.com/react/>
6. <https://masteringnextjs.com/>
7. <https://overreacted.io/>
8. <https://testingjavascript.com/>
9. <https://www.udemy.com/course/react-redux/>
10. <https://nodejs.org/en/docs/>
11. <https://expressjs.com/>

Books:

1. Casciaro, M., & Marasoiu, L. (2022). Node.js Design Patterns. O'Reilly Media.
2. Wargo, J. (2022). Full-Stack React Projects: Build Modern Web Applications Using React 18, Node.js, Express, and MongoDB. Packt Publishing.