Proceedings of IYSC, (2022), vol. 11

Journal homepage: http://journals.sdu.edu.kz/index.php/iysw

CALCULATING THE EFFECTIVENESS OF SPEEDING UP PAGE LOADING OF THE REACT.JS WEBSITE BY IMPLEMENTING THE NEXT.JS FRAMEWORK

T. Kudabayev

Suleyman Demirel University, *Kaskelen, Kazakhstan*

Proceedings of IYSC, (2022), vol. 11

Journal homepage: http://journals.sdu.edu.kz/index.php/iysw

Abstract

The main purpose of this article is to investigate the effectiveness of the Next.js tool for speeding up website loading in browsers. For qualitative research, information about how the client's browser works will be shown. How the React.js framework works, as well as how the Next.js framework works, their advantages and disadvantages, as well as the distinguishing features from each other. For practical proof of the effectiveness of implementing the Next.js framework on the site, site hosting will be used, in which a subdomain for two sites will be created: the first using only the React.js framework, and the second using the Next.js framework. To improve the comparison of indicators, both of these sites will use the same functionality. The end results will be compared against several metrics: the size of the final stable build of the program that can be tested and offered to the user, as well as platform independent scores for measuring site accessibility.

Keywords: React.js library, Next.js framework, Speeding up the React.js loading, performance evaluation rating, first contentful paint, time to interact parameter

Introduction

Website developers are constantly improving development tools and libraries in order to make their websites more functional. Various frameworks and auxiliary libraries are popular among developers, and the most popular among them is the React.js library. According to the "Stack Overflow Developer Survey 2022", the React.js library is used by 40.14% of 67.593 respondents [1].

The React.js library allows you to assemble a complex project from small and isolated pieces of code called components [2]. Websites like Facebook, Netflix, Instagram, and many others use React.js as a development tool [3]. After creating an application in React.js, we get a SPA application. One of the main advantages of the SPA application is that clients will be able to get instant results from working with the website (going to other pages, etc.), because the SPA applications are downloaded completely on the client side. Also, due to the use of service workers, the SPA application can work offline. On the other hand, one of the main disadvantages of this type of application is the size of the final JavaScript files that website clients have to download.

One of the main attributes of the website to the user is the time of its loading. According to the data from the "WebsiteBuilderExpert" service, 73% of mobile website users experience

website loading time issues [4]. Also based on them, one in four users leaves the site if it doesn't load in four seconds. In 2006, Amazon conducted an A/B test that found that every 100 milliseconds the website load increased, resulting in a 1% loss in sales [5]. That is why website loading is one of the most important attributes for a business.

In order to get around this shortcoming of the React.js application, the Next.js framework was invented [6]. It is a React-based framework that allows you to build web applications with improved performance and improved user experience with additional pre-rendering features such as full SSR (Server-Side Rendering) and SPG (Static Page Generation). As a result, the server will give the client the filled HTML, the filled state, and also all the necessary JS, CSS, and other resources. The client will receive HTML resources, synchronise the state, and work with the application as with a regular SPA.

There are a large number of articles on the Internet regarding the use of the React.js library and the Next.js framework. In some articles, the experience of using the React.js library is given, and in others, the Next.js framework. This article will focus on comparing the loading speeds of websites with the same functionality built into both of these tools. To structure this article, it will be divided into several logically separated sections in a numbered order from 1 to 5, including this introduction section. In Section 2, we will have methods for this topic. Section 3 presents the results and discussion of this framework. Section 4 provides conclusions. Finally, section 5 provides references, and the last section provides an abstract.

Methods

In order to start working on projects, you need to install a special npm package manager. It can be installed by downloading a special installer from the official website [7]. After successful installation, open a terminal and type "npm -v" in Figure 1 to check if the package manager works.

```
temirlankudabayev@MacBook-Pro-Temirlan articles % npm -v
8.9.0
```

Figure 1: NPM Version check

After checking the functionality of the package manager, you need to install the React project. To install, you need to enter the command "npx create-react-app article-react-app" in the terminal in Figure 2 [8].

Figure 2: Installing the article-create-app

Once installed, a basic React app will be created that can be launched using the "npm run start" command in Figure 3 [8].

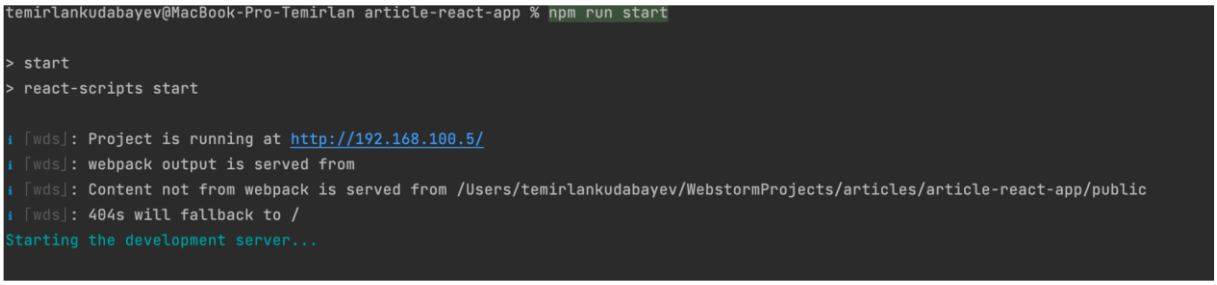

Figure 3: Run the article-create-app server

To work with styles, you need a node-sass library to work with a CSS framework called Sass [9]. Also, to speed up the creation of pages, the bootstrap library was added, which has ready-made stylized classes and components [10]. The finished application will have four pages: home, about us, clients, team, and contacts.

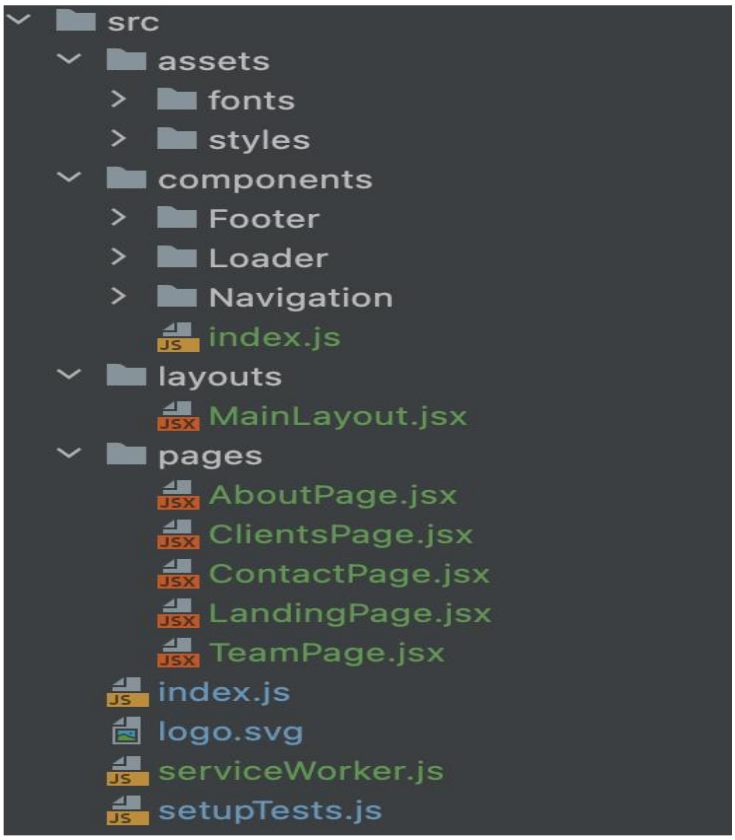

Figure 4: Architecture of the article-create-app

In order to make better structured applications, the file system will be divided by their logical bundles, as shown in Figure 4:

- The Assets folder, where the styles of the application are stored, as well as the fonts used
- The Components folder is where individual React components used in layouts or pages are placed.
- The Layouts folder, which stores a common layout for the entire application.
- The Pages folder, where application pages are stored.
- The index.js file is the React library's main application.
- The file logo.svg is the main logo of the site.
- The serviceWorker.js file is a service worker of the React application [11].
- The setupTests.js file is the file that is responsible for the test system in our application, but in this case, we will not be responsible for writing the tests.

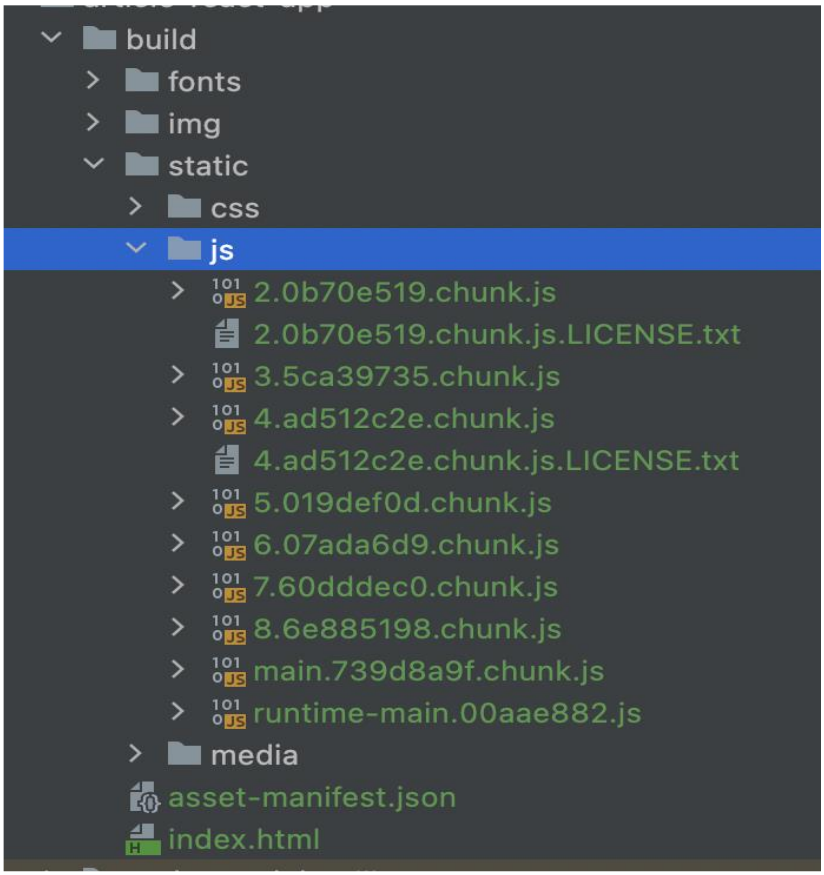

Figure 5: Optimised production build of the article-create-app

After completing work on the react project, using the "npm run build" command [8], react will make a build (a grouped folder with ready-made and compiled files) in Figure 5, which can be uploaded to the server. Working with a Next.js project is a bit different than working with a React.js project. There is also a section in the official documentation of the next.js framework on how to properly migrate a react website to next.js [12]. First you need to copy the article-react-js project, which stores a completely finished, previously made react project. Next, you need to install the next package and add the "next.config.js" configuration as shown in figure 6.

```
temirlankudabayev@MacBook-Pro-Temirlan article-nextjs-app % npm install next

up to date, audited 49 packages in 2s

4 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Figure 6: Installing the Next framework to the article-nextjs-app

Next.js has a different folder allocation system than React. In React, we created separate page files and then imported them into a common router as shown in figure 7 [13].

```
<Switch>
  <Route exact path="/about" component={AboutPage} />
  <Route exact path="/clients" component={ClientsPage} />
  <Route exact path="/team" component={TeamPage} />
  <Route exact path="/contact" component={ContactPage} />
  <Route path="/" component={LandingPage} />
  <Route path="/*" component={() => <Redirect to="/" />} />
</Switch>
```

Figure 7: Routing system in article-react-app

In the next.js project, the framework checks for the presence of files in the "pages" folder and builds routing systems relative to them, as shown in figure 8 [14].

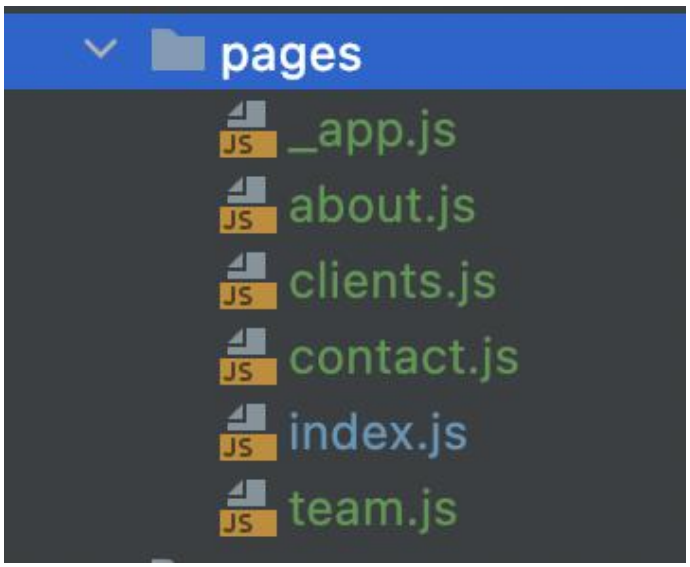

Figure 8: Routing system in article-nextjs-app

After the migration to the Next.js framework is completed, we launch the project using the "npm run start" command, and the project is assembled [6]. Next, go to "localhost:3000" and see the same project as on React.js, but now it is produced using the Next.js framework. To create the optimised build of article-nextjs-app, first of all, you need to enter the npm run build command. After creating the ".next" folder, you need to enter the npm run export command [15]. The result of these operations will be the creation of a separate folder called "out", where the build for the frontend part of the application will be stored as shown in Figure 9.

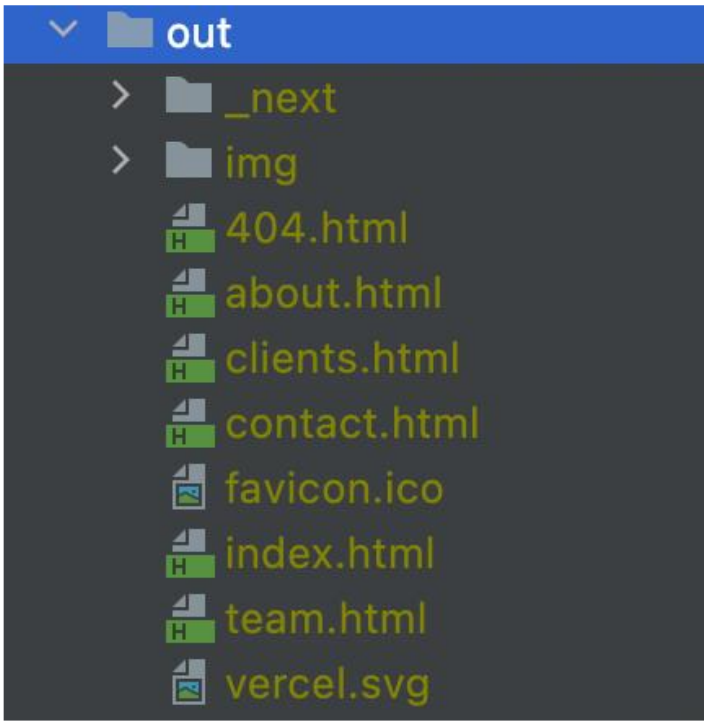

Figure 9: Optimised production build of the article-nextjs-app

After creating a build of two projects, we can visually compare them. The optimised build of article-react-app (Figure 5) has a single index.html file that loads the main javascript file and auxiliary chunks, while the article-nextjs-app build generates all rendering for the user.

To compare the results of the work of two projects, you need to upload them to a hosting with two separate domains connected. These domains must be different for the convenience of the experiment. Hosting must be configured correctly since the services that test the operation of sites emulate the operation of the browser. To improve the quality of testing, it is necessary to properly configure the server so that the test results are minimally dependent on the server. In order to properly configure the server, you need to understand how the browser works [16]:

- The user enters the address of the desired site to be bypassed.
- The browser looks up the server by IP address in DNS.
- The browser attempts to establish a connection to the server using a special TCP/IP protocol.
- The browser sends an HTTP request to the server to display the page.
- The server, using Apache or nginx, processes the request and sends a response to the browser to display the page the user needs.
- The browser processes the response and displays the requested content.

The first two steps depend on the availability of the server and the domain associated with them. The fifth step depends on the correct configuration of the server. For the experiment, a virtual host was chosen, which already had an initially fully configured web server. For website hosting, we use the Kazakhstani service ps.kz. We rent hosting there and bind 2 subdomains to the existing newtone.kz domain for the react project and for next, react.newtone.kz and nextjs.newtone.kz, respectively, as shown in figure 10.

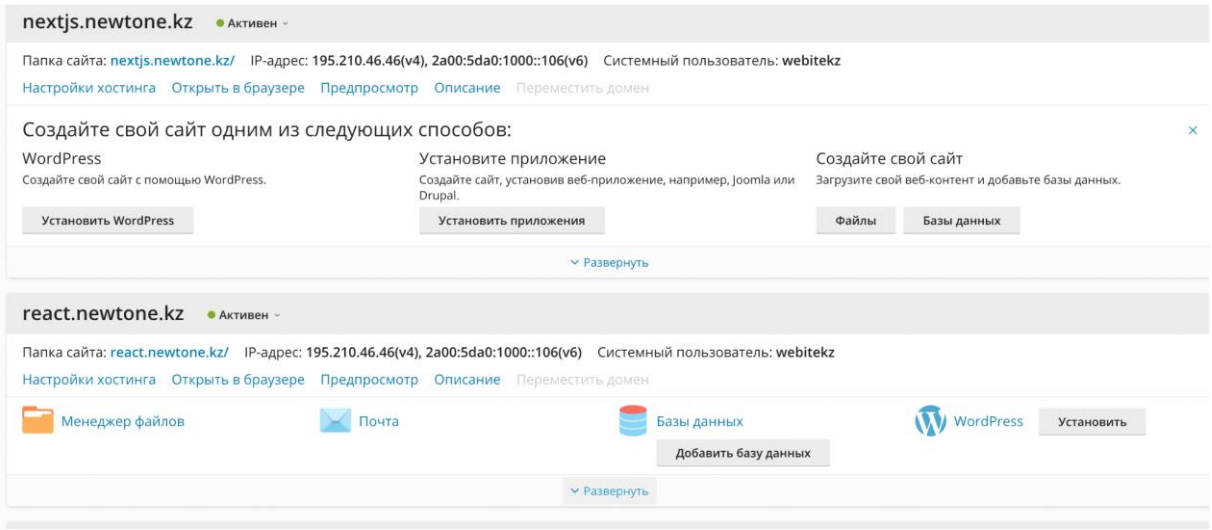

Figure 10: Creating separate subdomains for the experiment

Next, we also install SSL/TCL certificates for two subdomains and set up a redirect from HTTP to HTTPS. After installing subdomains, folders with the same subdomain names were automatically created. They need to fill in the details of our applications. After downloading all the necessary files, we go to the sites https://react.newtone.kz and https://nextjs.newtone.kz and see that our applications are successfully functioning on figures 11 and 12, respectively.

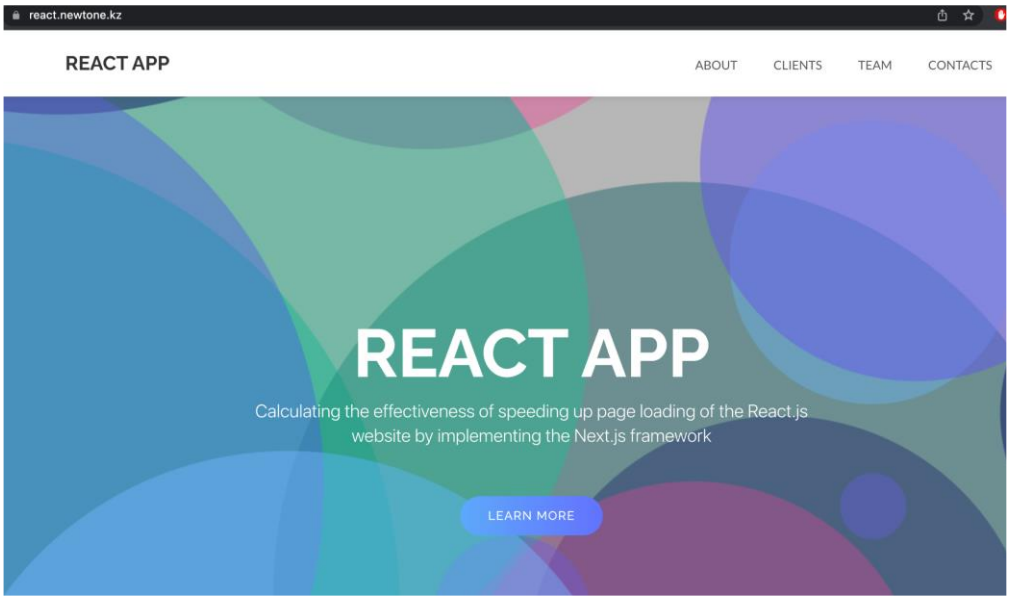

Figure 11: https://react.newtone.kz website

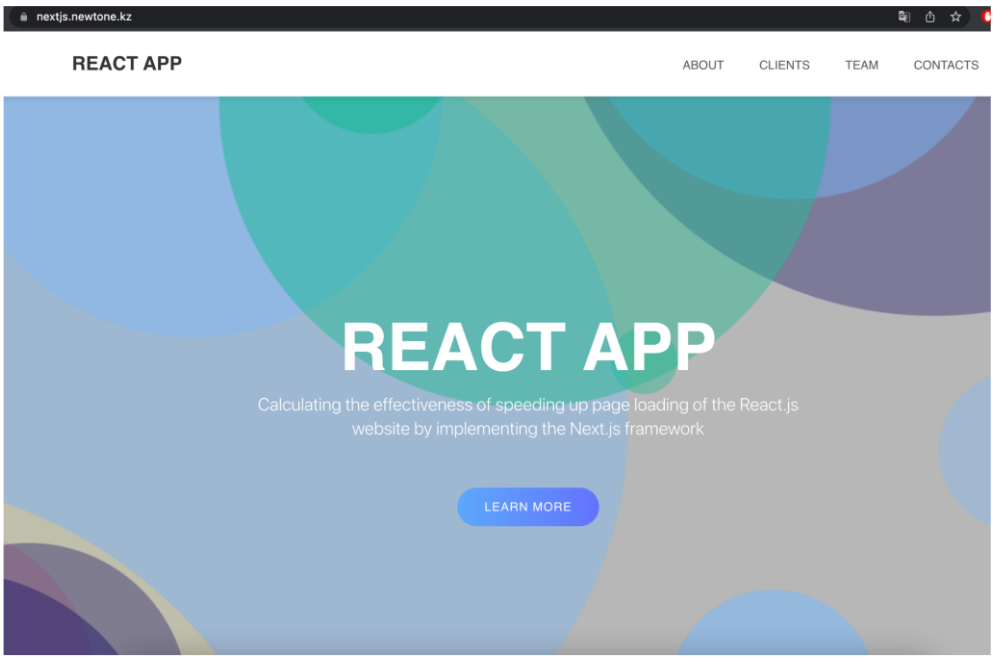

Figure 12: https://nextjs.newtone.kz website

Several popular and independent services will be used to evaluate the site's loading speed:

- pagespeed.web.dev. A service from Google that allows you to receive reports on the speed of loading pages on mobile devices and computers, as well as tips on how to increase this speed.
- pr-cy.ru. service for a complete analysis of the content of the site. The service has

useful tools for webmasters, SEOs, marketers, and copywriters. To test the site's loading speed, the speed_test section is required.

- gtmetrix.com. service for evaluating the performance of a website. The site allows you to select a country from the available list, the server of which will check the speed of the site. The only drawback is the lack of checking the mobile version of the site.

The verification algorithm is to go to each service that is listed above, drive in the link to our sites, and wait for the results. From the general list of results, you must select parameters such as overall rating, first contentful paint, and time to interact. For each of these parameters, the results will be entered into a separate table relative to the mobile and desktop versions of the site.

Results and discussion

The verification algorithm is to go to each service that is listed above, drive in the link to our sites, and wait for the results. From the general list of results, you must select parameters such as overall rating, first contentful paint, and time to interact. For each of these parameters, the results will be entered into a separate table relative to the mobile and desktop versions of the site.

	react.newtone. kz	nextjs.newtone .kz
The size of the final build in megabytes	5.96	3.99

Table 1

The final build size is very important for website loading speed. This size means the maximum amount of data the user will need to download in order to fully work with the website. This means that the smaller the size of the final build, the better it will be for the user. Table 1 shows data regarding build sizes for each of the projects. The results show that the build of nextjs.newtone.kz is 1.97 megabytes less than that of react.newtone.kz.

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newtone.kz	87	87	-

nextjs.new tone.kz	96	96	-

Table 2

Table 2 shows the download speed estimates for mobile versions of sites out of 100. The results of the gtmetrix.com service are not taken into account due to the inability to test the loading of the mobile version of the site on this service. The results show that, on average, independent services score 96 for nextjs.newtone.kz, while for react.newtone.kz the score is 87. This means that, in general, sites on next.js load faster on mobile devices.

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newt one.kz	97	96	70
nextjs.new tone.kz	100	100	79

Table 3

Table 3 shows the download speed estimates for desktop versions of sites out of 100. The situation for desktop devices is the same as for mobile devices. Based on the results of the three services, all three services score higher on next.js than react.js. The average rating of the site nextjs.newtone.kz is 93, while the site react.newtone.kz has 87.

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newt one.kz	2.9	2.9	-
nextjs.new tone.kz	1.1	1.2	-

Table 4

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newt one.kz	0.9	1	2.3
nextjs.new tone.kz	0.3	0.4	1.6

Table 5

Table 4 shows the results of services for the first contentful painting parameter on mobile devices, while Table 5 shows the results of this parameter for desktop devices. The first content parameter is very important in terms of website loading speed. This parameter shows how long after the page starts loading, the user begins to receive the first site data on his device. Based on the results, next.js sites render 1.7 seconds faster on mobile and an average of 0.7 seconds faster on desktop than react.js sites.

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newt one.kz	2.9	2.9	-
nextjs.new tone.kz	2.2	2.3	-

Table 6

	pagespeed .web.dev.	pr-cy.ru	gtmetrix.com
react.newt one.kz	0.9	1	2.3
nextjs.new tone.kz	0.3	0.4	2.1

Table 7

Table 6 and Table 7 show the results of services for the time to interactive parameter on

mobile devices and on desktop devices, respectively. The time to interactive parameter shows how many seconds after the start of loading the web page, the user can start working with the site. Based on the results, Next.js based project starts up 0.65 seconds faster on mobile, and an average of 0.47 seconds faster on desktop than a React-based project.

We expected that the implementation of the next.js framework into a React application when developing from the front-end part of the application would speed up the initial loading of the application. Based on the above aspects and results, we can conclude that the implementation of the next.js framework allows you to speed up the initial loading of the react application.

Conclusion

We have developed two projects on the React.js library and on the Next.js react-based framework with the same functionality and design. We uploaded them to the same server and set up the server so that the test results depend only on the results of the project build. The results were compared against several parameters, such as build size, first contentful paint, and time to interactive parameters, as well as website loading speed scores on mobile and desktop devices. In all of the above results, applications written in the Next.js framework performed better than applications written in React. With the help of Next.js, we accelerated the project using the React.js library.

To improve the research, in the future we can collect more additional parameters to improve our performance indicators. Also, to improve performance, we can increase the list of independent services. One of the ways to develop this research is to write an automated algorithm that would collect results from various services. Also, due to the fact that we used a fairly simple web application, we can modify it or make it non-standard so that the final javascript files are larger than they are now. These changes will improve the reliability of our comparison.

References

1. StackOverflow, "Stack Overflow Developer Survey 2021",2021. [Online] Available: https://insights.stackoverflow.com/survey/2021
2. Meta Platforms, "Getting started – React", 2022. [Online] Available: https://reactjs.org/docs/getting-started.html
3. Coder Academy, "Top 32 Sites Built With ReactJS", Jun 10,2016. [Online] Available: https://medium.com/@coderacademy/32-sites-built-with-reac tjs-172e3a4bed81

4. Maura Monaghan "Website Load Time Statistics: Why SpeedMatters in 2022", April 28, 2022. [Online] Available: https://www.websitebuilderexpert.com/building-websites/website-load-time-statistics/
5. Yoav Einav, "Amazon Found Every 100ms of Latency Costthem 1% in Sales", January 20, 2019. [Online] Available: https://www.gigaspaces.com/blog/amazon-found-every-100m s-of-latency-cost-them-1-in-sales
6. Guillermo Raucha, "Next.js Documentation", 2022. [Online] Available: https://nextjs.org/docs
7. OpenJS Foundation, "Node.js", 2022. [Online] Available: https://nodejs.org/en
8. Meta Platforms, "Create a new React App", 2022. [Online] Available: https://reactjs.org/docs/create-a-new-react-app.html
9. Andrew Nesbitt, "Node-sass library", 2015. [Online] Available: https://github.com/sass/node-sass
10. Bootstrap Team, "Bootstrap 5.1 Documentation", 2022. [Online] Available: https://getbootstrap.com/docs/5.1/getting-started/introduction/
11. Mozilla Corporation, "Service Worker API", Apr 30, 2022. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API
12. Guillermo Raucha, "From React to Next.js", 2022. [Online] Available:https://nextjs.org/learn/foundations/from-react-to-n extjs
13. Ryan Florence & Michael Jackson, "React Router: Declarative Routing for React.js", 2022. [Online] Available: https://v5.reactrouter.com/web/api/Switch
14. Guillermo Raucha, "Routing: Introduction in Next.js", 2022. [Online] Available: https://nextjs.org/docs/routing/introduction