**Push Notification in React and Next.js app using Node.js**

[[Reetesh Kumar](https://medium.com/@rajreetesh7?source=post_page-----da39ad1332ef--------------------------------)](https://medium.com/@rajreetesh7?source=post_page-----da39ad1332ef--------------------------------)

[Reetesh Kumar](https://medium.com/@rajreetesh7?source=post_page-----da39ad1332ef--------------------------------)

·

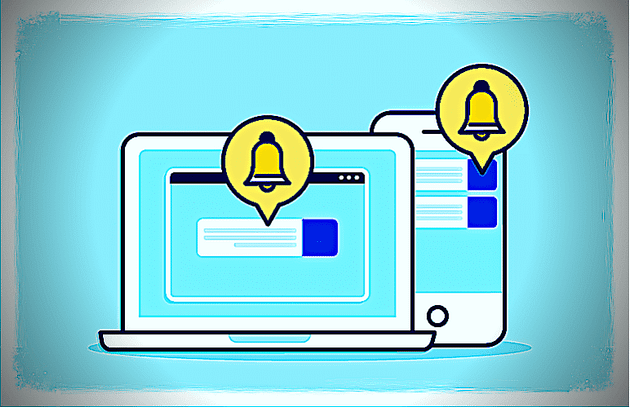
Follow

4 min read

·

Dec 31, 2023

163



Push Notification in Web apps

Push notification is a great way to keep your users engaged with your app. Server can send push notifications to the client even when the client is not active.

Having said that, let’s learn how to implement push notification in React and Next.js app using Node.js.

**How Push Notification Works in Web Apps?**

Push notifications in web apps involve a combination of browser APIs, service workers, and a server that can send push messages.

If the browser is offline, the message is queued until the browser comes online. Users can optin to receive these push notifications from a website. Subscribers will then get the notifications on both mobile and desktop browsers. This is why web push notifications are also called browser notifications.

**What is Service Worker?**

service worker is a script that runs in the background of a web application, separate from the web page. It enables features like background sync, push notifications, and caching.

Service worker is important for push notification because it is the one that receives the push notification from the server and shows it to the user.

In order to register a service worker, we need to create a file called sw.js in our public folder of Next.js or React app.

// public/sw.js  
  
self.addEventListener("push", (event) => {  
 const data = event.data.json();  
 const title = data.title;  
 const body = data.body;  
 const icon = data.icon;  
 const url = data.data.url;  
  
 const notificationOptions = {  
 body: body,  
 tag: "unique-tag", // Use a unique tag to prevent duplicate notifications  
 icon: icon,  
 data: {  
 url: url, // Replace with the desired URL for redirecting user to the desired page  
 },  
 };  
  
 self.registration.showNotification(title, notificationOptions);  
});

We implemented the push event listener in the service worker. The push event is fired when the server sends a push notification to the client.

**Registering Service Worker in React and Next.js**

We need to register the service worker in our React and Next.js app. We will use the useEffect hook to register the service worker.

// app/page.js -> Next.js  
// src/index.js -> React  
  
useEffect(() => {  
 if ("serviceWorker" in navigator) {  
 navigator.serviceWorker  
 .register("/sw.js")  
 .then((registration) => {  
 console.log("Registration successful");  
 })  
 .catch((error) => {  
 console.log("Service worker registration failed");  
 });  
 }  
}, []);

We are registering the service worker in the useEffect hook. We are checking if the serviceWorker is available in the navigator object. If it is available, we are registering the service worker.

**Sending Push Notification from Server**

We will use the [web-push](https://www.npmjs.com/package/web-push) library to send push notification from our Node.js server.

To send push notification using web-push, we need to create a VAPID key pair. We can create a VAPID key pair using the web-push library.

npm install web-push -g  
  
web-push generate-vapid-keys

You will get public and private key which we can store in our environment variable.

// server.js  
  
import express from "express";  
import webpush from "web-push";  
import dotenv from "dotenv";  
dotenv.config();  
  
const app = express();  
  
const vapidKeys = {  
 publicKey: process.env.VAPID\_PUBLIC\_KEY,  
 privateKey: process.env.VAPID\_PRIVATE\_KEY,  
};  
  
  
webpush.setVapidDetails(  
 "test@gmail.com"  
 vapidKeys.publicKey,  
 vapidKeys.privateKey  
)  
  
let subscriptions = [];  
  
app.post("/subscribe", (req, res) => {  
 const subscription = req.body;  
 subscriptions.push(subscription);  
  
 res.status(201).json({status: "success"});  
});  
  
app.post("/send-notification", (req, res) => {  
 const notificationPayload = {  
 title: "New Notification",  
 body: "This is a new notification",  
 icon: "https://some-image-url.jpg",  
 data: {  
 url: "https://example.com",  
 },  
 };  
  
 Promise.all(  
 subscriptions.map((subscription) =>  
 webpush.sendNotification(subscription, JSON.stringify(notificationPayload))  
 )  
 )  
 .then(() => res.status(200).json({ message: "Notification sent successfully." }))  
 .catch((err) => {  
 console.error("Error sending notification");  
 res.sendStatus(500);  
 });  
});  
  
app.listen(4000, () => {  
 console.log("Server started on port 4000");  
});

We setup the server using Express.js. We created two routes /subscribe and /send-notification.

In the /subscribe route, we are storing the subscription object in the subscriptions array. We will use this array to send push notification to all the subscribers. you can store the subscription object in the database as well.

In the /send-notification route, we are sending the push notification to all the subscribers. We are using the webpush.sendNotification method to send the push notification.

We are sending the push notification to all the subscribers using the Promise.all method. You can send the push notification to a specific user by filtering the subscriptions array.

**Subscribing to Push Notification**

We need to subscribe to push notification in order to receive the push notification. We will use the serviceWorker object to subscribe to push notification.

// app/page.js -> Next.js  
// src/index.js -> React  
  
// We modified the useEffect hook to subscribe to push notification  
  
useEffect(() => {  
 if ("serviceWorker" in navigator) {  
 const handleServiceWorker = async () => {  
 const register = await navigator.serviceWorker.register("/sw.js");  
  
 const subscription = await register.pushManager.subscribe({  
 userVisibleOnly: true,  
 applicationServerKey: "VAPID\_PUBLIC\_KEY",  
 });  
  
 const res = await fetch("http://localhost:4000/subscribe", {  
 method: "POST",  
 body: JSON.stringify(subscription),  
 headers: {  
 "content-type": "application/json",  
 },  
 });  
  
 const data = await data.json();  
 console.log(data);  
 };  
 handleServiceWorker();  
 }  
}, []);

We are subscribing to push notification in the useEffect hook. We are using the pushManager.subscribe method to subscribe to push notification. Once user is subscribed to push notification, we are sending the subscription object to the server.

**Sending Push Notification from Server**

// run server.js  
  
node server.js

Now open Postman and send a post request to http://localhost:4000/send-notification route. You will receive the push notification in your browser.

**Conclusion**

In this article, we learned how to implement push notification in React and Next.js app using Node.js. wae implemented a basic push notification system. You can extend this system to send push notification to a specific user or group of users.

That’s it for this article. I hope you found it useful. If you have any questions or feedback, please share me on [Twitter](https://twitter.com/iMBitcoinB).