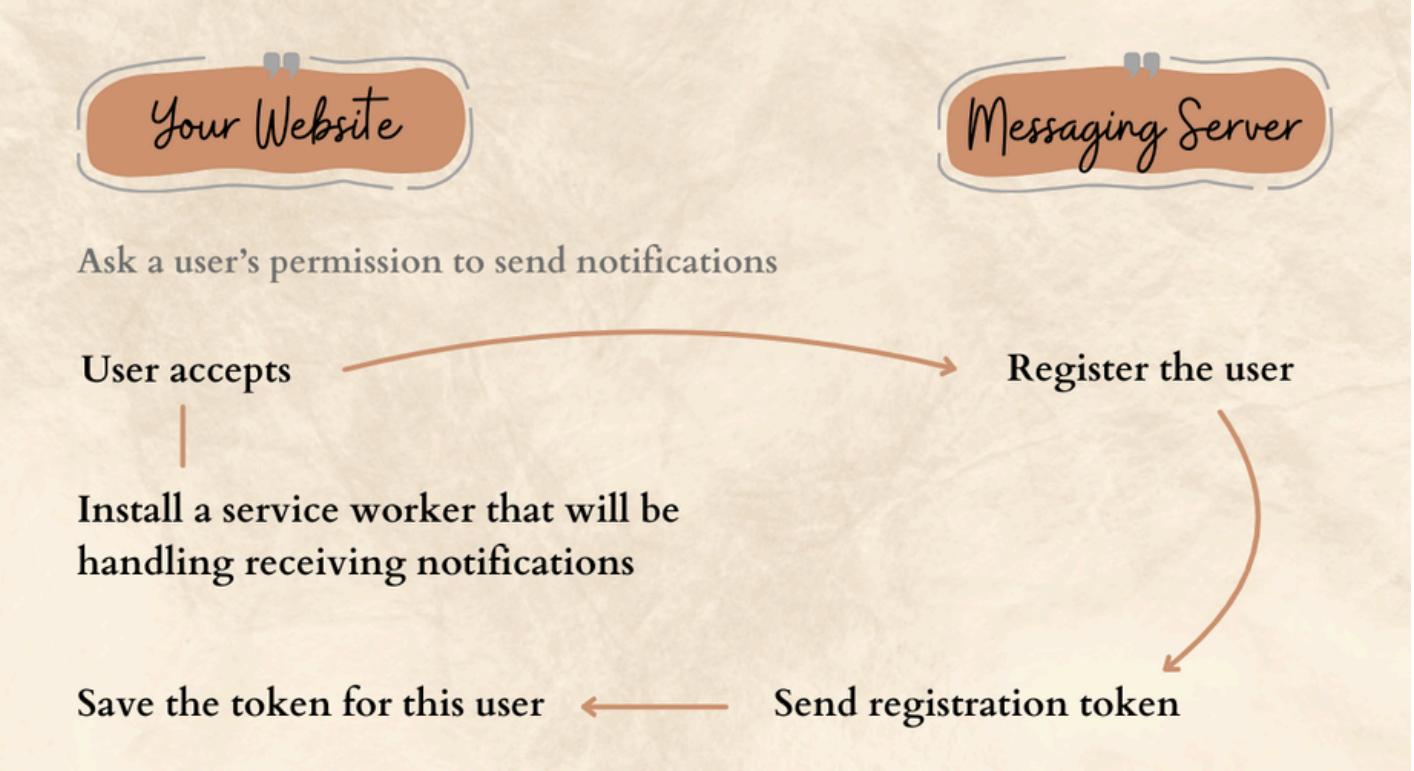
Firebase Push Notification in JavaScript Apps

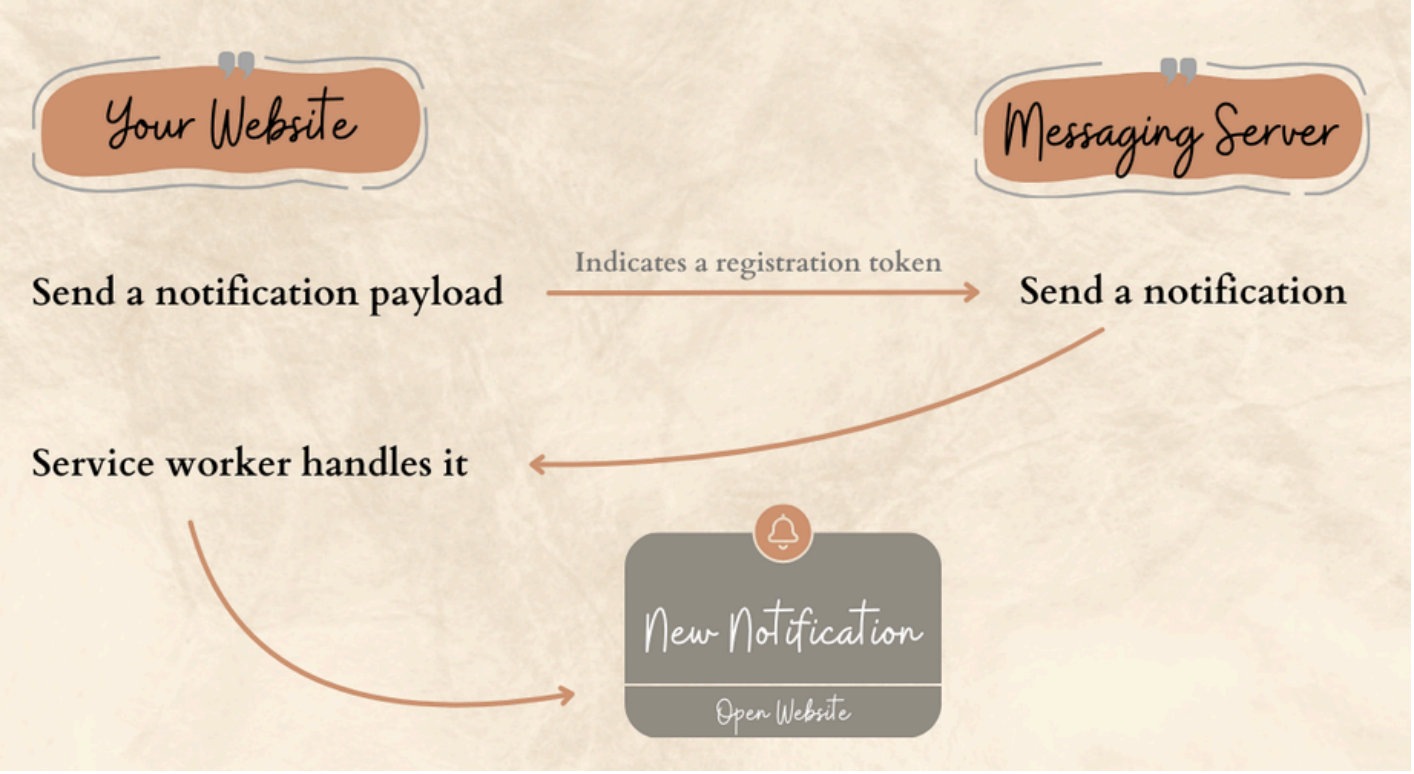
Firebase Push notification have become an integral part of engaging users and keeping them connected with your web application. In this blog post, we'll explore the implementation of push notifications using Firebase for JavaScript applications. By leveraging **Firebase Cloud Messaging (FCM)**, we can seamlessly integrate real-time communication and user engagement into our web apps.

How Do Firebase Push Notifications Work?

To enable push notifications on your website, the process starts with seeking the user's permission to receive notifications. If the user agrees, a service worker is installed in the background to handle these notifications.



Simultaneously, a request is sent to a messaging server to register the user, providing a means for the server to send notifications. Once registered, the messaging server issues a unique token for the user, essentially an address for directing push notifications. This token is saved for future reference.



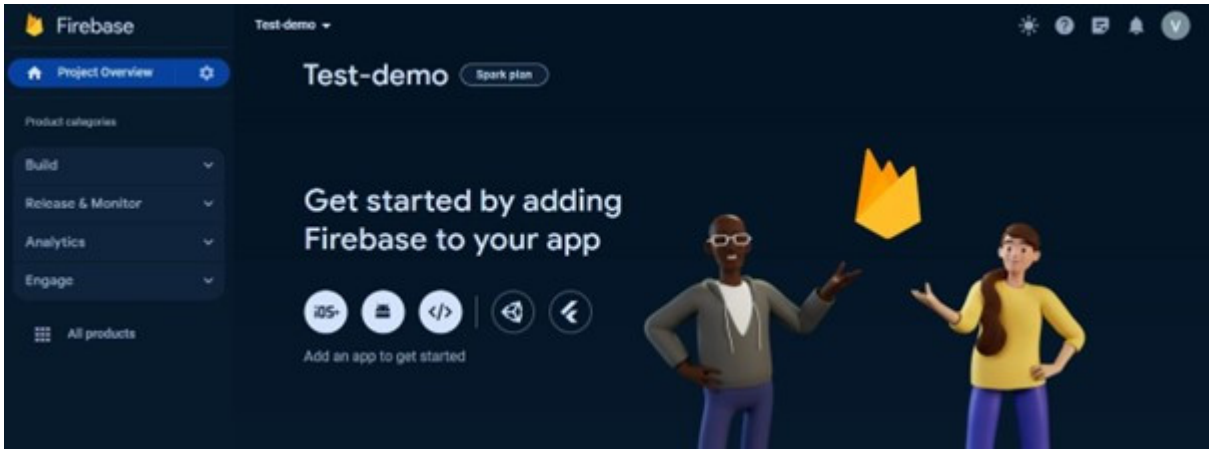
When it's time to send a notification, the saved token is used to inform the messaging server about the intended recipient. The user's browser, equipped with the service worker, then

constructs and displays the notification based on the received message. This streamlined process ensures that users who opt-in for notifications can seamlessly receive and view them through the installed service worker.

Setting Up Firebase in Your JavaScript App

Before diving into push notifications, let's begin by setting up Firebase in your JavaScript application. We'll guide you through the steps of creating a Firebase project, obtaining your configuration settings, and initializing Firebase in your app.

1. Create a project: To create a project, go to the Firebase Console and start a new project by following the provided steps. Once the project is successfully created, you'll see the following dashboard.



2. Create an app: Proceed to create a new application. Choose the appropriate options based on your requirements, such as iOS, Android, and others. We'll select web as we are implementing it for the React.js application.

In the second step, you will get the configurations for the Firebase SDK that we will need for our React app. You can also find the same in project settings.

Front-end set up

Step 1

Create a new React application, though this algorithm is applicable to any JavaScript application, irrespective of the library or framework being used

Step 2

Install Firebase SDK using your preferred package installer.

npm install firebase

Step 3

Then create a **firebase.js** file in the src directory initialize Firebase and begin using the SDKs. You can copy and paste the configuration content from the Firebase Console project settings.

```
import { initializeApp } from "firebase/app";
import { getMessaging, getToken, onMessage } from "firebase/messaging";

const firebaseConfig = {
  apiKey: "xxxx",
  authDomain: "xxx.firebaseapp.com",
  projectId: "xxx",
  storageBucket: "xxx.appspot.com",
  messagingSenderId: "xxx",
  appId: "xxx",
  measurementId: "xxx",
};

// Initialize Firebase
const app = initializeApp(firebaseConfig);
const messaging = getMessaging(app);
```

Step 4

Find a place where you want to ask a user about the notification permission. For example, in the App.js file, I have this button that says “**enable push notifications**“.

```
import { useState } from "react";

import { getFirebaseToken } from "./firebase";

export default function App() {
  const [showNotificationBanner, setShowNotificationBanner] = useState(
    Notification.permission === "default"
  );

  const handleGetFirebaseToken = () => {
    getFirebaseToken()
      .then((firebaseToken) => {
        console.log("Firebase token: ", firebaseToken);
        if (firebaseToken) {
          setShowNotificationBanner(false);
        }
      })
      .catch((err) =>
        console.error("An error occured while retrieving firebase token. ", err)
      );
  };

  return (
    <div className="app">
      {showNotificationBanner && (
        <div className="notification-banner">
          <span>The app needs permission to</span>
          <button onClick={handleGetFirebaseToken}>
            enable push notifications.
          </button>
        </div>
      )}
    </div>
  );
}
```

Step 5

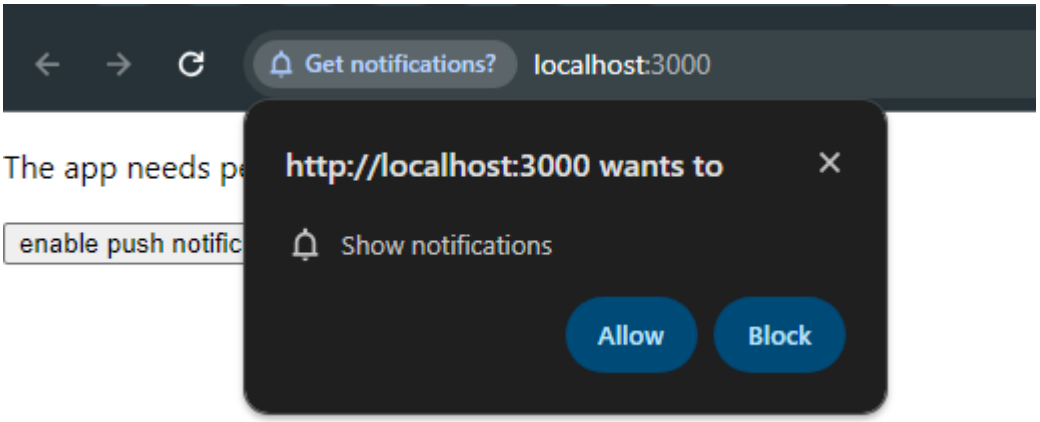
Implement the **getFCMToken()** in the firebase.js file. This function ensures that your application has access to a service worker registration. If one already exists, it returns it. If not, it registers a new service worker and returns the registration.

1 RWH <RX FDQILQGWKH YDSLGNH\ LQWKH) &0 FRQVROH

```
export const getOrRegisterServiceWorker = () => {
  if ("serviceWorker" in navigator) {
    return window.navigator.serviceWorker
      .getRegistration("/")
      .then((serviceWorker) => {
        if (serviceWorker) return serviceWorker;
        return window.navigator.serviceWorker.register(
          "/firebase-messaging-sw.js",
          {
            scope: "/",
          }
        );
      });
  }
  throw new Error("The browser doesn`t support service worker.");
};

export const getFirebaseToken = () =>
  getOrRegisterServiceWorker().then((serviceWorkerRegistration) => {
    return getToken(messaging, {
      vapidKey:"xxx",
      serviceWorkerRegistration,
    });
  });
```

With this code, we start up the Firebase library and create the getFCMToken() function. This function gets a registration token from Firebase Cloud Messaging (FCM) and asks the user for notification permission. It interacts with FCM to register the user only if permission is granted; otherwise, it throws an error, which you catch in the catch block.



After the user allows the notifications, you obtain an FCM token, a unique identifier for the user in the FCM system, used for sending notifications. It's crucial to store this token somewhere. Typically, you'd send it to a server, which then saves it in the database for that user. Without storing the tokens, you can't send notifications to users. For this, the Firebase Admin SDK is needed, available in server environments.

There are exceptions. In cases where you only want to subscribe users to notifications, like in a newsletter, you may not need to store the FCM tokens. Firebase manages them, and you can manually send notifications from the Console. However, automatic (programmatic) sending is not possible because you lack the individual user tokens needed for differentiation.

Step 6

The final step is to incorporate a service worker responsible for managing notifications from Firebase Cloud Messaging (FCM).

Create a file named **firebase-messaging-sw.js**, and ensure it is available at the root of your web app. I have this file in the public folder.

```
importScripts(
  "https://www.gstatic.com/firebasejs/9.10.0/firebase-app-compat.js"
);
importScripts(
  "https://www.gstatic.com/firebasejs/9.10.0/firebase-messaging-compat.js"
);

var firebaseConfig = {
  apiKey: "YOUR_API_KEY",
  authDomain: "YOUR_AUTH_DOMAIN",
  projectId: "YOUR_PROJECT_ID",
  storageBucket: "YOUR_STORAGE_BUCKET",
  messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
  appId: "YOUR_APP_ID",
};

firebase.initializeApp(firebaseConfig);

const messaging = firebase.messaging();
```

Manage Notifications

Foreground Notifications: These are notifications received while the application is open. To handle such notifications, incorporate the provided code into your firebase.js file and use the function in your components.

```
export const onForegroundMessage = () =>
  new Promise((resolve) => onMessage(messaging, (payload) => resolve(payload)));
```

Background Notifications: Received when the app is not open, these are aptly named background notifications. To manage these notifications, add the provided code to your firebase.js file.

```
messaging.onBackgroundMessage((payload) => {
  console.log("Received background message: ", payload);

  const notificationTitle = payload?.data?.title;
  const notificationOptions = {
    body: payload?.data?.body,
  };

  self.registration.showNotification(notificationTitle, notificationOptions);
});
```

In summary, leveraging Firebase for push notifications in JavaScript apps enhances user engagement and real-time communication. This guide simplifies the process, enabling you to seamlessly integrate this valuable feature into your web application. If you're looking for expert assistance in implementing these strategies, consider partnering with a top-tier **Java Development Company** for a seamless and effective solution. Elevate your app development with the right expertise!

Originally published by: **Firebase Push Notification in JavaScript Apps**