Notes

Cloud Messaging

See the Firebase Cloud Messaging docs for web.

manifest.json

```
{
  "gcm_sender_id": "103953800507"
}
```

Request permission in browser

```
// index.html
const messaging = firebase.messaging();
messaging
  .requestPermission()
  .then(function() {
    // Get Instance ID token. Initially this makes a network call, once retrieved
    // subsequent calls to getToken will return from cache.
    messaging.getToken()
    .then(function(currentToken) {
      if (currentToken) {
        sendTokenToServer(currentToken);
        updateUIForPushEnabled(currentToken);
      } else {
        // Show permission request.
        console.log('No Instance ID token available. Request permission to
generate one.');
        // Show permission UI.
        updateUIForPushPermissionRequired();
        setTokenSentToServer(false);
      }
    })
    .catch(function(err) {
      console.log('An error occurred while retrieving token. ', err);
      showToken('Error retrieving Instance ID token. ', err);
      setTokenSentToServer(false);
    });
  }
  .catch(function(err) {
    console.log('Unable to get permission to notify.', err);
  });
```

Monitor token refresh

```
// index.html
// Callback fired if Instance ID token is updated.
messaging.onTokenRefresh(function() {
  messaging
    .getToken()
    .then(function(refreshedToken) {
      console.log('Token refreshed.');
      // Indicate that the new Instance ID token has not yet been sent to the
      // app server.
      setTokenSentToServer(false);
      // Send Instance ID token to app server.
      sendTokenToServer(refreshedToken);
      // ...
    })
    .catch(function(err) {
      console.log('Unable to retrieve refreshed token ', err);
      showToken('Unable to retrieve refreshed token ', err);
    });
});
```

Catch messages when page is in foreground

```
// index.html
// Handle incoming messages. Called when:
// - a message is received while the app has focus
// - the user clicks on an app notification created by a sevice worker
//   `messaging.setBackgroundMessageHandler` handler.
messaging.onMessage(function(payload) {
  console.log('Message received. ', payload);
  // ...
});
```

Create serviceWorker

> You need a serviceWorker to listen for messages in the background

```
// firebase-messaging-sw.js
// Give the service worker access to Firebase Messaging.
// Note that you can only use Firebase Messaging here, other Firebase libraries
// are not available in the service worker.
importScripts('https://www.gstatic.com/firebasejs/4.8.1/firebase-app.js');
importScripts('https://www.gstatic.com/firebasejs/4.8.1/firebase-messaging.js');

// Initialize the Firebase app in the service worker by passing in the
// messagingSenderId.
firebase.initializeApp({
  messagingSenderId: 'YOUR-SENDER-ID',
});

// Retrieve an instance of Firebase Messaging so that it can handle background
// messages.
const messaging = firebase.messaging();
```

Send message to single recipient

```
// Cloud Function
// This registration token comes from the client FCM SDKs.
var registrationToken = 'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...';

// See the "Defining the message payload" section below for details
// on how to define a message payload.
var payload = {
  notification: {
    title: 'Title of your push notification',
    body: 'Body of your push notification',
    click_action: 'https://dummypage.com',
  },
  data: {
    score: '850',
    time: '2:45',
  },
};

// Send a message to the device corresponding to the provided
// registration token.
admin
  .messaging()
  .sendToDevice(registrationToken, payload)
  .then(function(response) {
    // See the MessagingDevicesResponse reference documentation for
    // the contents of response.
    console.log('Successfully sent message:', response);
  })
  .catch(function(error) {
    console.log('Error sending message:', error);
  });
```

Send multi-cast message

```
// Cloud Function
// These registration tokens come from the client FCM SDKs.
var registrationTokens = [
  'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...',
  // ...
  'ecupwIfBy1w:APA91bFtuMY7MktgxA3Au_Qx7cKqnf...',
];

//...
admin
  .messaging()
  .sendToDevice(registrationTokens, payload)
  .then(function(response) {
    //...
  });
```

Send device-group message

> See managing device groups

```
// Cloud Function
var notificationKey = 'some-notification-key';

//...
admin
  .messaging()
  .sendToDeviceGroup(notificationKey, payload)
  .then(function(response) {
    // ...
  });
```

Send topic message

> See managing device groups

```
// Cloud Function
// The topic name can be optionally prefixed with "/topics/".
var topic = 'highScores';

//...

admin
  .messaging()
  .sendToTopic(topic, payload)
  .then(function(response) {
    //...
  });
```

Send to condition

> Conditions support only two operations per expression

```
// Cloud Function
// Define a condition which will send to devices which are subscribed
// to either the Google stock or the tech industry topics.
var condition = "'stock-GOOG' in topics || 'industry-tech' in topics";

//...
admin
  .messaging()
  .sendToCondition(condition, payload)
  .then(function(response) {
    //...
  });
```

Message options

```
// Cloud Function
var registrationToken = 'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...';

var payload = {
  notification: {
    title: 'Urgent action needed!',
    body: 'Urgent action is needed to prevent your account from being disabled!',
  },
};

// Set the message as high priority and have it expire after 24 hours.
var options = {
  priority: 'high',
  timeToLive: 60 * 60 * 24,
};

admin
  .messaging()
  .sendToDevice(registrationToken, payload, options)
  .then(function(response) {
    //...
  });
```

Subscribe to topic

```
// Cloud Function
var registrationTokens = [
  'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...',
  // ...
  'ecupwIfBy1w:APA91bFtuMY7MktgxA3Au_Qx7cKqnf...',
];

admin
  .messaging()
  .subscribeToTopic(registrationTokens, topic)
  .then(function(response) {
    //...
  });
```

Subscribe to topic

```
// Cloud Function
const registrationTokens = [
  'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...',
  // ...
  'ecupwIfBy1w:APA91bFtuMY7MktgxA3Au_Qx7cKqnf...',
];

const topic = 'highScores';

admin
  .messaging()
  .subscribeToTopic(registrationTokens, topic)
  .then(function(response) {
    //...
  });
```

Unsubscribe to topic

```
// Cloud Function
const registrationTokens = [
  'bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...',
  // ...
  'ecupwIfBy1w:APA91bFtuMY7MktgxA3Au_Qx7cKqnf...',
];

const topic = 'highScores';

admin
  .messaging()
  .unsubscribeFromTopic(registrationTokens, topic)
  .then(function(response) {
    //...
  });
```