Storing and Querying Data in a Firebase Realtime Database with JavaScript

Table of Contents

1 Overview

Google's Firebase Realtime Database is a NoSQL database that can be used to store and retrieve data for applications on Android, iOS, Web, C++, and Unity platforms. Firebase is a feature-rich application development platform, including user management and authentication, storage, hosting, app distribution, Google analytics, crash analytics, performance monitoring, A/B testing, remote configuration, and more.

This document provides several examples of Firebase database interaction, in real time, with a Web frontend using JavaScript (JS).

2 Setting up a Firebase Realtime Database

A detailed discussion of configuring a Firebase Realtime Database is beyond the scope of this document, but the process is straightforward. Here is a high-level workflow:

1. Add a Firebase Project
    a. Name the project
    b. Enable or Disable Google Analytics
    c. If enabled, configure Google Analytics
    d. Create the project
2. Add an Application
    a. Select the "Web" as the application platform
    b. Register the app
    c. Copy the Firebase SDK scripts for use in your JS application
3. Create a database
    a. Select a database mode (production or test)
    b. Select a database location
    c. Select the Realtime Database option
    d. Select, configure, and publish database rules

3 Firebase Realtime Database and JavaScript / jQuery

3.1.1 Integrate Firebase with your HTML files

In order to use a Firebase database with your application, the following JavaScript files will need to appear at the bottom of the body tag for any HTML page that will interact with the Firebase database.

```
<!--jQuery JS file -->
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<!--Firebase References-->
<script src="https://www.gstatic.com/firebasejs/6.0.4/firebase-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/6.0.4/firebase-database.js"></script>
```

3.1.2 Initialize Firebase in your application's JS file

Your main JavaScript file must be initialized with the Firebase database's configuration parameters. Add the SDK parameters from the Firebase project settings to your application's JavaScript file, for example:

```
"$(document).ready(function () {

// Set Firebase Configuration
    var firebaseConfig = {
        apiKey: "AIzaSyC9Xj0lZfEHTfHO0jkpJq1whqJtUW6KABs",
        authDomain: "trainscheduler-81cb1.firebaseapp.com",
        databaseURL: "https://trainscheduler-81cb1.firebaseio.com",
        projectId: "trainscheduler-81cb1",
        storageBucket: "trainscheduler-81cb1.appspot.com",
        messagingSenderId: "304178649914",
        appId: "1:304178649914:web:660ea99647252144ec4e03"
    };

// Initial Firebase with the Configuration Settings
    firebase.initializeApp(firebaseConfig);

// Initialize a database variable
    var database = firebase.database();
});"
```

3.1.3 Manipulate Firebase Data

Once the database has been initialized within your JavaScript file, you can begin storing data in and retrieving data from the Realtime Database, in real time.

3.1.3.1 Retrieve Existing Database Records

You can retrieve a snapshot of records for an existing database using "query.once" in your JS file or script tag. In the example below, we retrieve a snapshot of existing database records using jQuery and then loop through the data set, displaying the results on our frontend.

```
query.once("value")
    .then(function (snapshot) {
        snapshot.forEach(function (childSnapshot) {
            // Pull stored train information from the database
            var trainName = childSnapshot.child("name").val();
            var trainDestination = childSnapshot.child("destination").val();
            var firstTrain = childSnapshot.child("first").val();
            var trainFrequency = childSnapshot.child("frequency").val();
            // Calculate Next Arrival and Minutes Away
            var currentTime = moment();
            var firstTrainConverted = moment(firstTrain, "HH:mm").subtract(1, "years");
            var diffTime = moment().diff(moment(firstTrainConverted), "minutes");
            var timeRemainder = diffTime % trainFrequency;
            var minutesAway = trainFrequency - timeRemainder;
            var nextArrival = moment(currentTime, 'HH:mm A').add(minutesAway,
'minutes').format("HH:mm A")
            // Create a row of data for the database
            var fullList = $("<tr>").append(
                $("<th scope='row'>").text(trainName),
                $("<td>").text(trainDestination),
                $("<td>").text(trainFrequency),
                $("<td>").text(nextArrival),
                $("<td>").text(minutesAway)
            );
            // Append the row to the existing table
            $("#train-table > tbody").append(fullList);
        });
```

3.1.3.2 Add a Database Record

Using "database.ref().push()", we can add a new record to the database. In the example below, when the user clicks the "submit button" on the frontend, we automatically create a new database record with a randomly generated unique key.

```
    $("#submit").on("click", function (event) {
          // Prevents form action default
          event.preventDefault();
              // Generate random userKey
              userID = Math.random().toString(36).substring(2, 15) +
Math.random().toString(36).substring(2, 15);
              // Assign date/time added to a variable
              userAdded = moment().format('lll');
              // Create database record for session with userKey
              var newDater = {
                  user: userID,
                  added: userAdded,
                  name: "",
                  source: "",
                  place: "",
                  date: "",
                  year: "",
                  month: "",
                  day: "",
                  lng: "",
                  lat: "",
                  restaurants: "",
                  desserts: "",
                  movies: ""
              };
              // Push new record to database
              database.ref().push(newDater);
      });
```

3.1.3.3 Update a Database Record

Using "database.ref().update()", we can update an existing database record. In the example below, we update a user record in the database based on its unique key.

```
async function updateDatabase() {
          // Get longitude and latitude from userPlace
          var convAPIkey = "c833b0a3e4104de495176d7252219568";
          var convQueryURL = "https://api.opencagedata.com/geocode/v1/json?countrycode=us&q=" +
userPlace + "&key=" + convAPIkey + "&language=en&pretty=1"
          await $.ajax({
              type: "GET",
              url: convQueryURL,
              datatype: "json",
              success: function (response) {
                  latitude = response.results[0].geometry.lat;
                  longitude = response.results[0].geometry.lng;
              }
          });
          // Update record in database based on Record Key
          var recordRef = database.ref(userRecordKey);
          recordRef.update({
              "name": userName,
              "source": placeSource,
              "place": userPlace,
              "date": userDate,
              "year": dateYear,
              "month": dateMonth,
              "day": dateDay,
              "lng": longitude,
              "lat": latitude,
              "restaurants": needRestaurant,
              "desserts": needDessert,
              "movies": needMovies
          });
}
```

3.1.3.4 Delete a Database Record

Similar to "database.ref().remove()", we can use a record's unique key to delete it from the database. In the example below, on "exit button" click on the frontend, we delete the user record from the database based on its unique key.

```
$("#exit").click(function () {
        // Check to see if there is a record in the database for the user
        if (userID !== "") {
            // If record exists, delete it
            var deleteRef = database.ref(userRecordKey);
            function delAssets() {
                deleteRef.remove();
                console.log("Deleted: " + userRecordKey);
            };
            delAssets();
            // Open farewell.html in the same tab
            window.open("goodbye.html", '_self');
        } else {
            // Open farewell.html in the same tab
            window.open("goodbye.html", '_self');
        }
    })
```

4 Firebase JS Documentation

For the full JavaScript documentation set for Firebase, please see https://firebase.google.com/docs/reference/js.