Web Front-End Development
Week 7: Firebase

The data for web sites is often stored in files or a database that is separate from your web pages. We've looked at how to use APIs to access data, now let's look at how we can store data we need to drive our web site.
There are many ways to do this, we're going to use Firebase. Firebase is a Backend as a Service (BaaS) platform that provides a realtime database on various platforms. Providing this as a service means you don't have to set up a database, write all the code to synchronize the data, and figure out security and authentication. Firebase stores data as JSON in the cloud in a NoSQL database.
Features: https://firebase.google.com/products/

- Database
- Cloud Storage
- Authentication
- Hosting
- Cross platform support – web, iOS, Android

Firebase was founded by Andrew Lee and James Tamplin in 2011, officially launched in April 2012, and purchased by Google two years later.

Firebase
https://firebase.google.com/ Get Started
Log in using your gmail account
Create New Project
Tickets
Firebase Console
Add Firebase to your web app
Copy the code snippet and paste it at the end of the body of your HTML file but before any other JavaScript. This imports the Firebase JavaScript SDK and initializes it for your Firebase Project. (make sure "storageBucket" has a value, otherwise close and reopen the dialog.)
You also need to add the library for the Cloud Firestore(must be after firebase-app.js)

```
<script src="https://www.gstatic.com/firebasejs/7.1.0/firebase-firestore.js"></script>
```

Once in the project you can always click on Settings | Project Settings and scroll down to get the required Firebase SDK code snippet.

Add Firebase to your JavaScript Project
https://firebase.google.com/docs/web/setup

Cloud Firestore
https://firebase.google.com/docs/firestore/quickstart
From the console's navigation pane, select Database, then click Create database for Cloud Firestore.
Start in test mode.

- Good for getting started with the mobile and web client libraries

- Allows anyone to read and overwrite your data.

Select the multi-regional location nam5(us-central)

Cloud Firestore Data Model
https://firebase.google.com/docs/firestore/data-model
Cloud Firestore is a NoSQL, document-oriented database. Unlike a SQL database, there are no tables or rows. Instead, you store data in documents, which are organized into collections.
Documents

- Document is the unit of storage
- A document is a lightweight record that contains fields, which map to values.
- Each document is identified by a unique id or name
- Each document contains a set of key-value pairs.
- All documents must be stored in collections.
- Documents within the same collection can all contain different fields or store different types of data in those fields.
- However, it's a good idea to use the same fields and data types across multiple documents, so that you can query the documents more easily.
- The names of documents within a collection are unique. You can provide your own keys, such as user IDs, or you can let Cloud Firestore create random IDs for you automatically.

Collection

- A collection is a container for documents
- A collection contains documents and nothing else.
- A collection can't directly contain raw fields with values, and it can't contain other collections.
- You can use multiple collections for different related data (orders vs users)

You do not need to "create" or "delete" collections. After you create the first document in a collection, the collection exists.
Let's create the first one using the console so we know our collection exists.

In the console go into Database | Data
Create a collection called orders.
Now create a document using auto id with the fields email, name, payment, and team.

Note that because we're in test mode our security rules are public and you'll see the following in the rules tab:

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write;
    }
  }
}
```

In JavaScript we get a reference to our firestore database.
var database = firebase.firestore();

Then we can get a reference to this collection.
var docRef = database.collection("orders");

Example:
Write data to Firebase https://firebase.google.com/docs/firestore/manage-data/add-data

- firebase_form_cfs.html, firebase_tickets_cfs.js
    - get a reference to our firebase database, child "orders"
    - create a JS object with the data(name, email, team, and payment) from the form as key/value pairs
    - write the data to the database using add()
        - add() lets Firestore generate a unique key every time a new document is added
            - you can get the id from the object returned
        - these ids do not provide any kind of ordering
    - set() would overwrite data so if it exists, otherwise it creates it
    - doc(new_id) lets you set the unique id
    - look in the Firebase console to see the data added to your database as JSON
- firebase_form2_cfs.html, firebase_tickets2_cfs.js
    - create a billing_address object with all billing info
    - create a payment object with all payment info
        - in real life you wouldn't store credit card data in the clear

Read data once https://firebase.google.com/docs/firestore/query-data/get-data

- Use .get() to get a document

Use Firestore listeners to automatically get updates
https://firebase.google.com/docs/firestore/query-data/listen

- firebase_orders_cfs.html, firebase_orders_cfs.js
    - You can *listen* to a document with the `onSnapshot()` method.
    - An initial call using the callback you provide creates a document snapshot immediately with the current contents of the single document.
    - Each time the contents change, another call updates the document snapshot.
    - The data from Firebase is delivered as a snapshot in JSON format
    - Leave this page open and add a new order and you'll see it update

Firestore also makes it easy to perform queries and order your data.