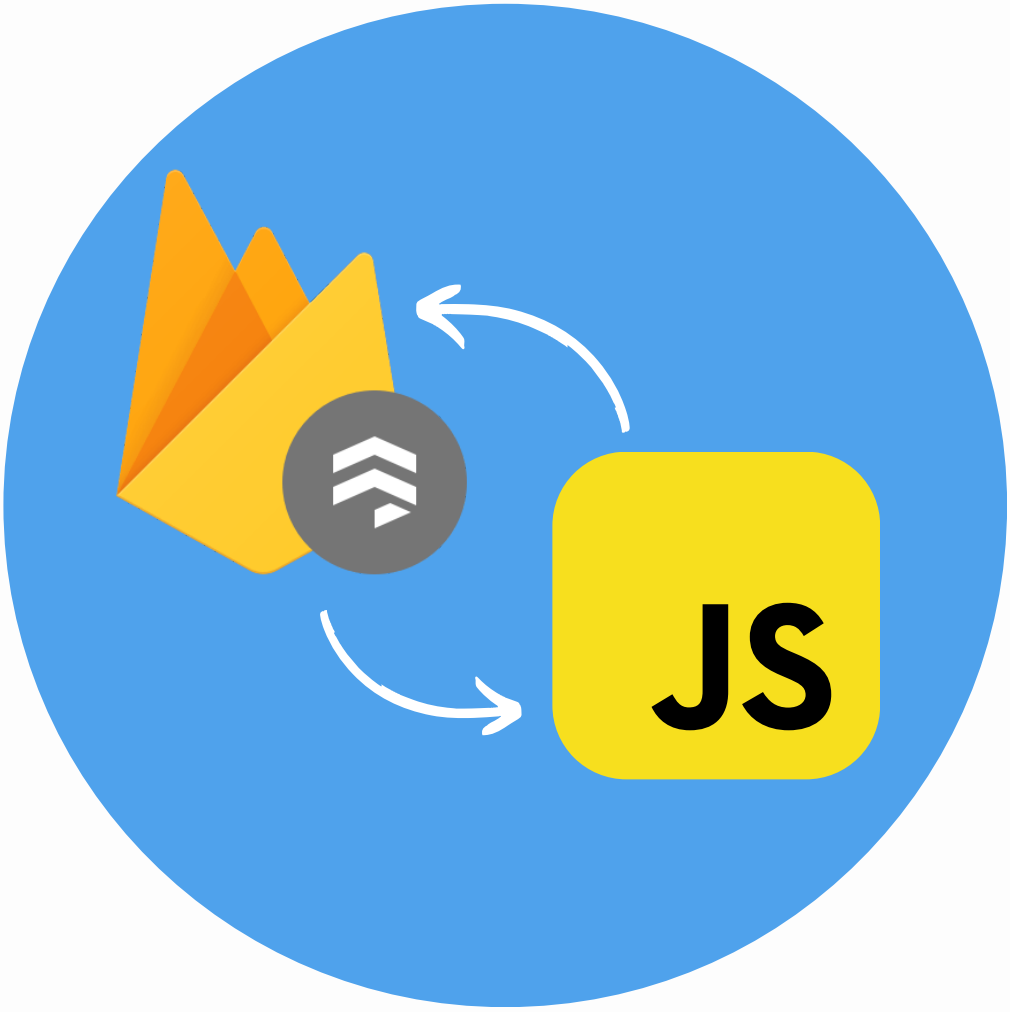
Firebase 9

Firestore Database CRUD Queries

Using JavaScript



SoftAuthor.com

Raja Tamil

Create / Add Data

Using addDoc()

The **addDoc()** method is one of the two ways to add document data to the Firestore Database in Firebase Version 9 (modular).

1. addDoc()
2. setDoc()

In order to use the **addDoc()** method, we need to import three methods from the Firebase Firestore module.

1. **getDatabase()** → Firestore Database where we want to add data as a document.
2. **collection()** → Where we want to add collection name and database references.
3. **addDoc()** → Where we actually pass our data along with the collection name and db references.

```
import {getFirestore, collection, addDoc} from "firebase/firestore";

addDoc(collection(db, "cities"), {
    name: "Ottawa",
    province: "Ontario",
    country : "Canada",
    code: "613"
})
.then(docRef => {
    console.log(docRef.id); // Document ID
})
.catch(error => {
    console.log(error);
})
```

Create / Add Data
Using setDoc()

Using the setDoc() method, you can add a document to the Firestore Database by creating:

- Auto-generated ID or
- Custom ID

The below code example shows how to create/add document data to the Firestore Database with a custom ID using the setDoc() method.

```
import {getFirestore, doc, setDoc} from "firebase/firestore";

const db = getFirestore(app);


setDoc(doc(db, "cities", "my.custom.id@gmail.com"), {
    name: "Ottawa",
    province: "ON",
    country : "Canada",
})
.then(() => {
    console.log("Data has been successfully added.")
})
.catch(error => {
    console.log(error);
})
```

Update Data
Using setDoc()

Using the setDoc() method, you can also update either the

- **Entire document** or
- **Specific document field**

Update Entire Document Data
Using setDoc()

When you run the below code, the setDoc() method will replace everything with new values mentioned in the data object.

```
import {getFirestore, doc, setDoc} from "firebase/firestore";
const db = getFirestore();

const docRef = doc(db, "cities", "jXdt5bRUwJov9dtCf7M6");
const data = {
    name: "Ottawa",
    province: "ON",
    country : "Canada",
};

setDoc(docRef, data)
.then(() => {
    console.log("Entire document has been updated successfully.");
})
.catch(error => {
    console.log(error);
})
```

Update A Specific Document Field
Using setDoc()

When you run the below code, the setDoc() method will replace everything with new values mentioned in the data object.

You can also add {merge:true} as a third argument of setDoc() method. When you do, 4 significant behaviours will occur which are:

1. If the data object is empty, the query does nothing in the Firestore document when merge:true is set.
2. If the data object has an existing field name and value matching exactly in the Firestore document, the query does nothing.
3. If the data object has any existing field name with the new value, the query will update the value of that field in the Firestore document.
4. If the data object has a field name with a value that does not exist in the Firestore document, the query will add that field to the document.

```
import {getFirestore, doc, setDoc} from "firebase/firestore";
const db = getFirestore();

const docRef = doc(db, "cities", "jXdt5bRUwJov9dtCf7M6");
const data = {
    name: "Ottawa",
    province: "ON",
    country : "Canada",
};

setDoc(docRef, data)
.then(() => {
    console.log("Entire document has been updated successfully.");
})
.catch(error => {
    console.log(error);
})
```

Update A Document
Using updateDoc()

Using the **updateDoc()** method, you can update an existing document with three actions.

1. **Add** A New Document Field.
2. **Update** A Value of An Existing Document Field.
3. **Delete** A Document Field.

1. Add A Document Field
Using updateDoc()

Using the **updateDoc()** method, you can add one or more fields to a Firestore document.

```
import {getFirestore, doc, updateDoc} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "yftq9RGp4jWNSyBZ1D6L");

const data = {
    code: "613", // New Field, does not exist in the document
};

updateDoc(docRef, data)
.then(()=> {
    console.log("New Document Field has been added!");
})
.catch(error => {
    console.log(error);
})
```

2. Update Existing Document Field
Using updateDoc()

Using the **updateDoc()** method, you can update the values of one or more fields in a Firestore document.

```
import {getFirestore, doc, updateDoc} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "yftq9RGp4jWNSyBZ1D6L");

const data = {
    code: "705", // Update A Value of An Existing Field
};

updateDoc(docRef, data)
.then(()=> {
    console.log("New Document Field has been added!");
})
.catch(error => {
    console.log(error);
})
```

3. Delete Existing Document Field
Using updateDoc()

Using the **updateDoc()** method, you can also delete one or more fields from a Firestore document.

```
import {getFirestore, doc, updateDoc, deleteField} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "yftq9RGp4jWNSyBZ1D6L");
const data = {
    code: deleteField()
};

updateDoc(docRef, data)
.then(() => {
    console.log("Code Field has been deleted successfully");
})
.catch(() => {
    console.log(error);
```

Delete An Entire Document
Using deleteDoc()

The the **deleteDoc()** method lets you delete an entire document from a collection in the Firestore Database.

```
import {getFirestore, doc, deleteDoc} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "10QnsDfdgD6wHcpKHJnl");

deleteDoc(docRef)
.then(() => {
    console.log("Entire Document has been deleted successfully.")
})
.catch(error => {
    console.log(error);
})
```

Get / Read All Documents
Using getDocs()

The **getDocs()** method lets you get all the documents from a specific collection in the Firestore Database.

via CDN

```
import { getFirestore, collection, getDocs } from
"https://www.gstatic.com/firebasejs/9.8.4/firebase-firestore.js";

const db = getFirestore();

const colRef = collection(db, "cities");

try {
    const docsSnap = await getDocs(colRef);
    docsSnap.forEach(doc => {
        console.log(doc.data());
        console.log(doc.id);
    })
} catch (error) {
    console.log(error);
}
```

Get / Read Data With Real-Time Updates
Using onSnapShot()

The **onSnapshot()** method gets documents from a collection and listens for any database changes, then updates the front-end automatically without refreshing the browser!

JS via CDN

```
import { getFirestore, collection, onSnapshot } from
"https://www.gstatic.com/firebasejs/9.9.3/firebase-firestore.js";

const db = getFirestore();
const dbRef = collection(db, "cities");

onSnapshot(dbRef, docsSnap => {
  docsSnap.forEach(doc => {
    console.log(doc.data());
  })
});
```

Get / Read Documents
Using Where Clause

The **where** clause is used to filter data in the query based on one or more specific conditions.

```
import {getFirestore, doc, getDoc} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "l3bcSGs2vZBIc3RODwp");

try {
    const docSnap = await getDoc(docRef);
    if(docSnap.exists()) {
        console.log(docSnap.data());
    } else {
        console.log("Document does not exist")
    }

} catch(error) {
    console.log(error)
}
```

Get / Read Documents
Using Multiple **Where** Clauses

The where clause is used to filter data in the query based on one or more specific conditions.

```
import { getFirestore, collection, query, where, getDocs }
from "firebase/firestore";

const db = getFirestore();
const collectionRef = collection(db, "Audience");

const q = query(collectionRef,
          where("country", "==", "USA"),
          where("age", ">=", 18),
);

const docSnap = await getDocs(q);

docSnap.forEach((doc) => {
  console.log(doc.data());
});
```

OUTPUT:

The above query will get all the documents from an Audience collection where the country is equal to the USA and age is 18 or above.

Get / Read A Document By ID
Using getDoc()

The **getDoc()** method allows us to get a specific document by ID from a collection in Firestore Database.

```
import {getFirestore, doc, getDoc} from "firebase/firestore";

const db = getFirestore();

const docRef = doc(db, "cities", "l3bcSGs2vZBIc3RODwp");

try {
    const docSnap = await getDoc(docRef);
    if(docSnap.exists()) {
        console.log(docSnap.data());
    } else {
        console.log("Document does not exist")
    }

} catch(error) {
    console.log(error)
}
```