

CIS 658 Web Architectures

Cloud DataBase

Firestore II



GRAND VALLEY
STATE UNIVERSITY®

Lecturer: **Dr. Yong Zhuang**

Recall

- Why Cloud Data Stores?
- SQL vs. noSQL
- Firebase: Cloud Firestore
 - Web App Creation and Local Setup
 - Collections and Documents Data Model
 - Create Documents
 - own document ID, automatic document ID, from an array
 - Read Documents
 - all documents, a specific document, documents with `where()`

CRUD: Firestore Update Functions

CRUD: Update Doc (change a simple field)

// Update a record with **a known primary key**

```
UPDATE students SET phone = "616-616-6161" WHERE gnumber = "G71884"
```

SQL

SQL table

Gnumber	Name	Phone
G81291	Abby	517-123-4567
G71884	Ally	269-333-4444 → 616-616-6161
G53181	Annie	616-777-3332

```
// Assume saved data has the
// following structure
type StudentType = {
  gnumber: string; // PrimaryKey
  name: string;
  phone: string;
};
```

```
import { doc, updateDoc, DocumentReference } from "firebase/firestore";
// After initialization
const docRef: DocumentReference = doc(db, "students/G71884");
// add a new simple data
updateDoc(docRef, { phone: "616-616-6161" }).then(() => {
  console.debug("Update successful");
});
```

Firestore in TS

CRUD: Update Doc (change a simple field)

```
// Update a record when primary key is UNKNOWN  
UPDATE students SET phone = "616-616-6161" WHERE name = "Abby"
```

SQL

SQL table

Gnumber	Name	Phone
G81291	Abby	517-123-4567
G71884	Ally	269-333-4444
G53181	Annie	616-777-3332

→ 616-616-6161

```
// Assume saved data has the  
// following structure  
type StudentType = {  
  gnumber: string; // PrimaryKey  
  name: string;  
  phone: string;  
};
```

```
import {  
  collection, CollectionReference, doc, getDocs, QueryDocumentSnapshot,  
  QuerySnapshot, query, updateDoc, where,  
} from "firebase/firestore";  
const myCol: CollectionReference = collection(db, "students");  
const qr = query(myCol, where("name", "==", "Abby"));  
getDocs(qr).then((qs: QuerySnapshot) => {  
  qs.forEach(async (qd: QueryDocumentSnapshot) => {  
    const myDoc = doc(db, qd.id);  
    await updateDoc(myDoc, { phone: "616-616-6161" });  
  });  
});
```

Firestore in TS

CRUD: Update array field in a Document

db

States (Collection)

MI

Name: Michigan
Capital: Lansing
univ: ["GVSU", "Calvin", "MSU", "UMich"]

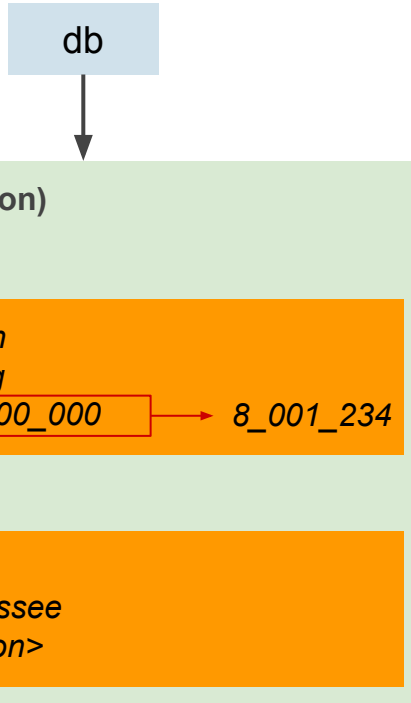
FL

Name: Florida
Capital: Tallahassee
cities: <Collection>

```
// After initialization
import {
  updateDoc,
  arrayRemove,
  arrayUnion,
  DocumentReference,
  doc,
} from "firebase/firestore";
const mich: DocumentReference = doc(db, "states/MI");
// add a new JS/TS array
updateDoc(mich, { universities: ["GVSU", "Calvin", "XYZ"] }).then(() => {
  console.debug("Update successful");
});
// updated erroneous entry in the array
updateDoc(mich, {
  universities: arrayRemove("XYZ"),
}).then(() => {
  console.debug("Update successful");
});
// Add more entries in the array
updateDoc(mich, {
  universities: arrayUnion("MSU", "UMich"),
}).then(() => {
  console.debug("Update successful");
});
```

Firestore in TS

CRUD: Update array field in a Document



```
import {
  updateDoc,
  increment,
  DocumentReference,
  doc,
} from "firebase/firestore";
const mich: DocumentReference = doc(db, "states/MI");
updateDoc(mich, {
  // Add 1234 to the current population
  population: increment(1234),
}).then(() => {
  console.debug("Update successful");
});
```

Firestore in TS

CRUD: Firestore Delete Functions

CRUD: Delete one Document

```
// Delete a record with a known primary key  
DELETE FROM students WHERE gnumber = "G71884"
```

SQL

SQL table

Gnumber	Name	Phone
G81291	Abby	517-123-4567
G71884	Ally	269-333-4444
G53181	Annie	616-777-3332

```
// Assume saved data has the  
// following structure  
type StudentType = {  
  gnumber: string; // PrimaryKey  
  name: string;  
  phone: string;  
};
```

```
import { deleteDoc, doc } from "firebase/firestore";  
// Delete the entire document  
const toRemove = doc(db, "students/G71884");  
deleteDoc(toRemove).then(() => {  
  console.debug("Student G71884 removed");  
});
```

Firestore in TS

CRUD: Delete one Document (unknown Doc ID)

```
// Update a record when primary key is UNKNOWN
```

```
DELETE FROM students WHERE name = "Abby"
```

SQL

SQL table

Gnumber	Name	Phone
G81291	Abby	517-123-4567
G71884	Ally	269-333-4444
G53181	Annie	616-777-3332

```
// Assume saved data has the  
// following structure  
type StudentType = {  
  gnumber: string; // PrimaryKey  
  name: string;  
  phone: string;  
};
```

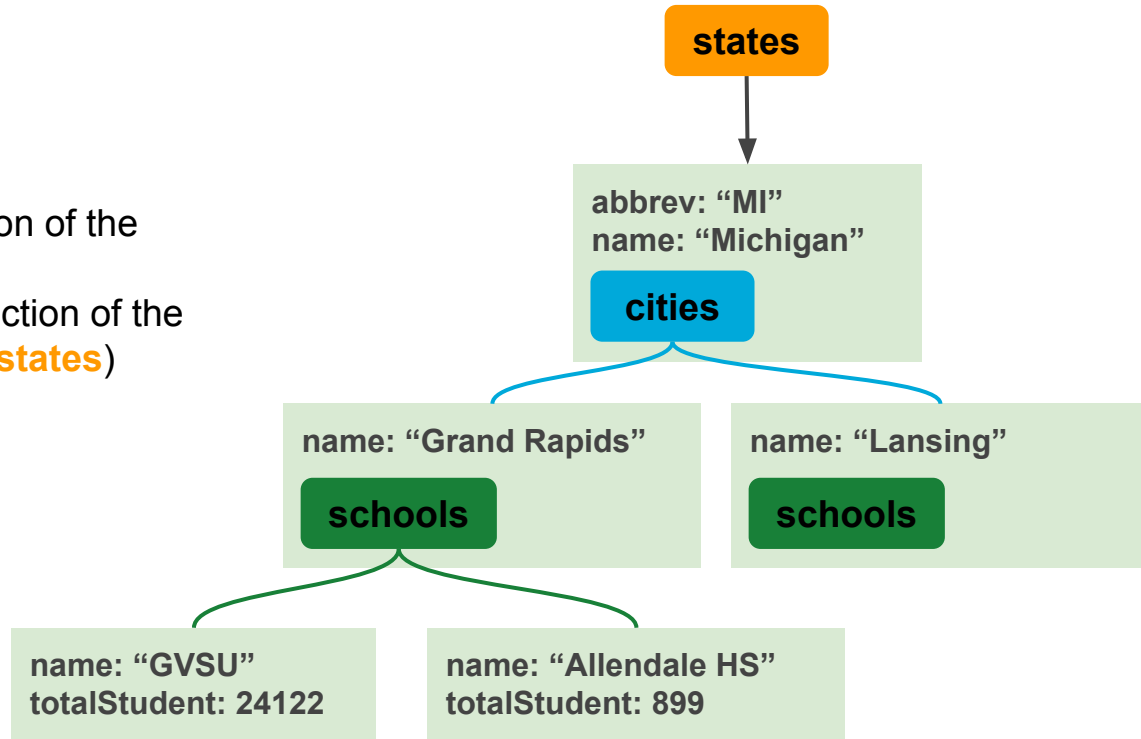
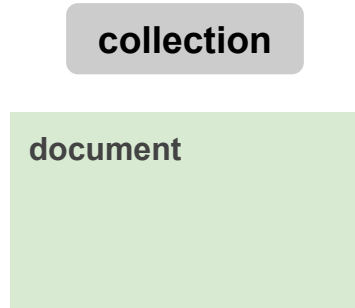
```
import { deleteDoc, doc, collection, CollectionReference,  
  QueryDocumentSnapshot, QuerySnapshot, query, where, getDocs,  
} from "firebase/firestore";  
const myCol: CollectionReference = collection(db, "students");  
const qr = query(myCol, where("name", "==", "Abby"));  
getDocs(qr).then((qs: QuerySnapshot) => {  
  qs.forEach(async (qd: QueryDocumentSnapshot) => {  
    const myDoc = doc(db, qd.id);  
    await deleteDoc(myDoc);  
  });  
});
```

Firestore in TS

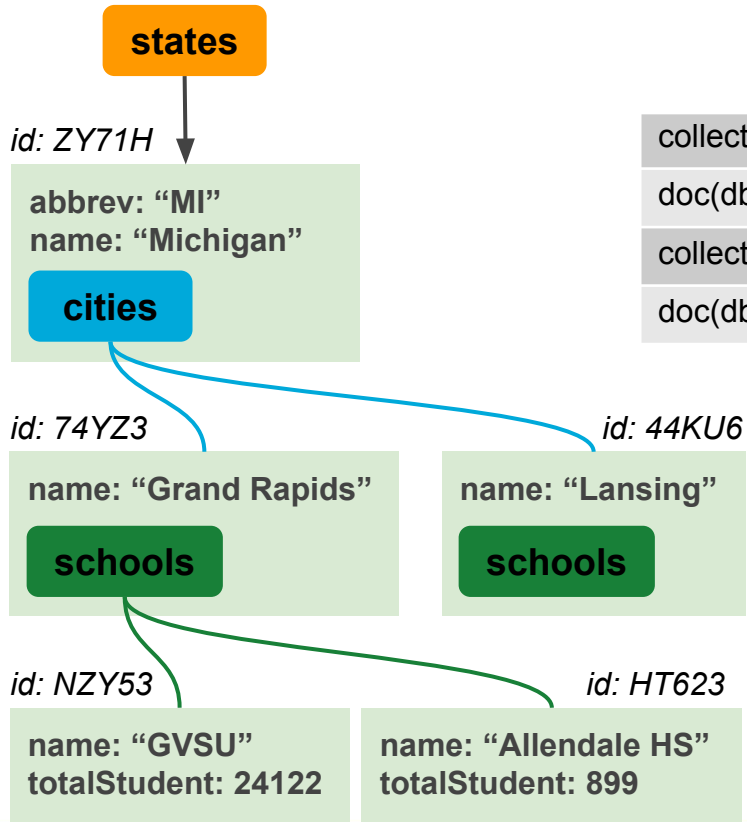
SubCollections: One-to-Many Relationship

One-to-Many Relationships

- One **state** has many **cities**
- One **city** has many **schools**
- In Firebase Firestore
 - **cities** becomes a subcollection of the **states** collection
 - **schools** becomes a subcollection of the **cities** (sub)collection (under **states**)



Sub-Collections



collection(db, "states/ZY71H/cities")	Cities in Michigan
doc(db, "states/Z71H/cities/74Y23")	City of Grand Rapids
collection(db, "states/ZY71H/cities/74Y23/schools")	Schools in GR
doc(db, "states/ZY71H/cities/74Y23/schools/NZY53")	GVSU

Operations on SubCollections

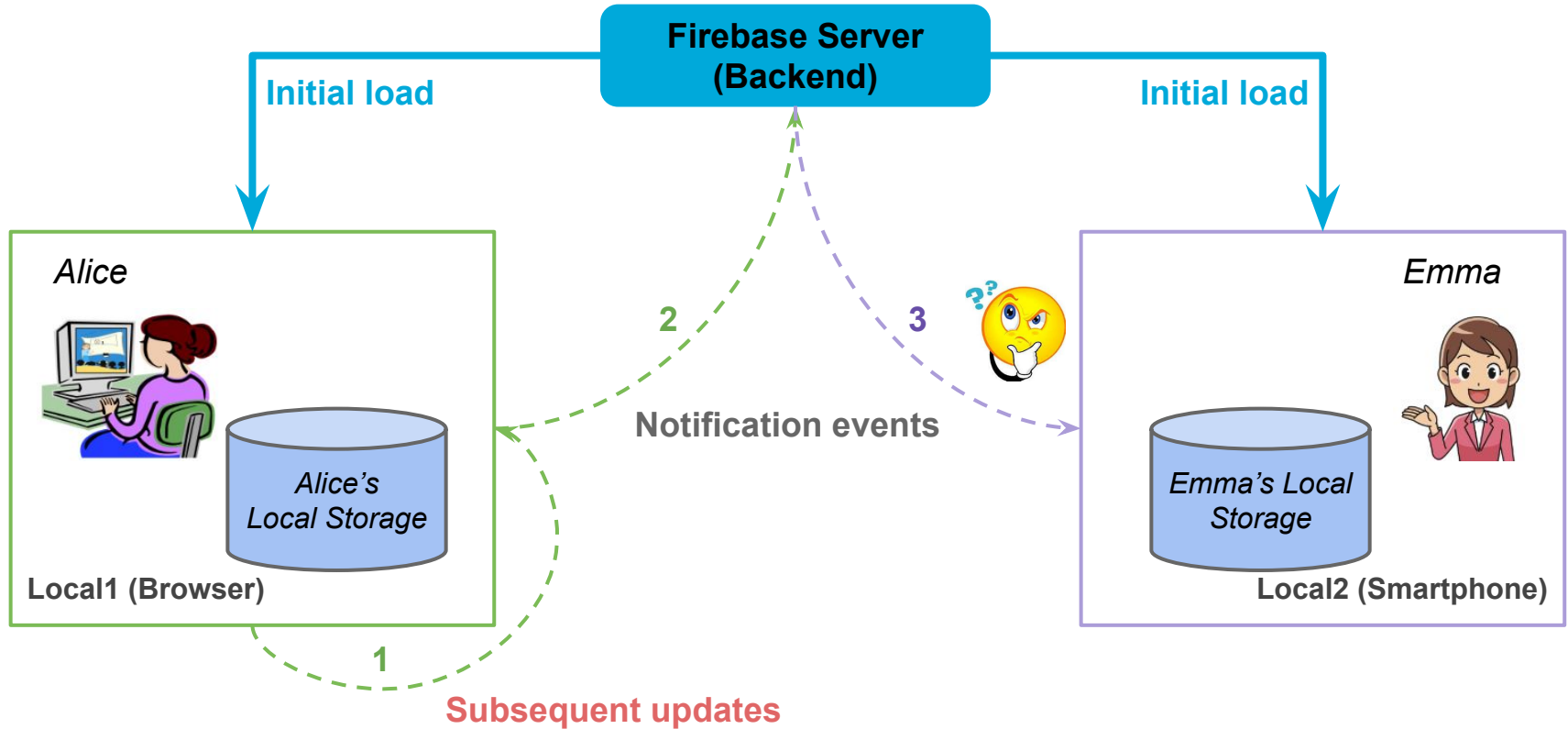
collection(db, "states/ZY71H/cities")	Cities in Michigan
doc(db, "states/Z71H/cities/74Y23")	City of Grand Rapids
collection(db, "states/ZY71H/cities/74Y23/schools")	Schools in GR
doc(db, "states/ZY71H/cities/74Y23/schools/NZY53")	GVSU

```
// Add a new city in Michigan
const miCities = collection(db, "states/ZY71H/cities");
await addDoc(miCities, {
  name: "Holland",
  /* more details on Holland */
});
```

```
// Update Grand Rapids details
const grDoc = doc(db, "states/Z71H/cities/74Y23");
await updateDoc(grDoc, { subwayAvailable: false });
```

```
// Get all schools in Grand Rapids
const grSchools = collection(db, "states/ZY71H/cities/74Y23/schools");
getDocs(grSchools).then((qs: QuerySnapshot) => {
  qs.forEach((qd: QueryDocumentSnapshot) => {
    const skool = qd.data() as SchoolType;
  });
});
```

Local Storage, Local Events, & Global Events



**Collection/Doc Update Listener(s) Benefit:
Interactive Web App**

Listening to Field Updates on a SINGLE doc

db

buildings

MAK

name: Mackinac Hall
location: Allendale
depts: ["CIS", "Math"]
numStaffs: 134
rooms: <SubCollection>

PAD

name: Padnos Hall
location: Allendale

DEV

name: DeVos Hall
location: Pew

```
import { doc, onSnapshot, DocumentSnapshot } from "firebase/firestore";
const mac = doc(db, "buildings/MAK");
// Listen to updates on a single document
const unsubscribe = onSnapshot(mac, (snapshot: DocumentSnapshot) => {
  const newData = snapshot.data();
  console.log("Document has been updated to", newData);
});

// Later ...

// Stop listening to changes
unsubscribe();
```

Firestore in TS

Will NOT receive notifications on updates of the doc subcollections (rooms in the example)

Listening to Updates on a Collection of Docs

db

buildings

MAK

name: Mackinac Hall
location: Allendale
depts: ["CIS", "Math"]
numStaffs: 134
rooms: <SubCollection>

PAD

name: Padnos Hall
location: Allendale

DEV

name: DeVos Hall
location: Pew

```
import { collection, onSnapshot, QuerySnapshot } from "firebase/firestore";
const bldColl = collection(db, "buildings");
const unsubscribe = onSnapshot(bldColl, (s: QuerySnapshot) => {
  for (let chg of s.docChanges()) {
    const newData = chg.doc.data();
    const updateAction = chg.type; // "added", "modified", "removed"
    console.log(chg.doc.id, "has been", updateAction, newData);
  }
});

// Later ...

// Stop listening to changes
unsubscribe();
```

Firestore in TS

Handle listen errors

db

buildings

MAK

name: Mackinac Hall
location: Allendale
depts: ["CIS", "Math"]
numStaffs: 134
rooms: <SubCollection>

PAD

name: Padnos Hall
location: Allendale

DEV

name: DeVos Hall
location: Pew

Detach listeners when leaving a page or unmounting a component.

```
import { doc, onSnapshot, DocumentSnapshot } from "firebase/firestore";
const mac = doc(db, "buildings/MAK");
// Listen to updates on a single document
const unsubscribe = onSnapshot(
  mac,
  (snapshot: DocumentSnapshot) => {
    const newData = snapshot.data();
    console.log("Document has been updated to", newData);
  },
  (error) => {
    // ...
  }
);
// Later ...

// Stop listening to changes
unsubscribe();
```

Firestore in TS

Firebase Firestore & Vue.js: Listening Functionality

- How to bind a textbox input's value with a Firestore document value?
- When modifying the value in the textbox, how do you update the corresponding document value in Firestore?
- How to detach this listener?



```
<template>
  <input v-model="data" @input="updateFirestore" />
</template>

<script setup lang="ts">
  // Import necessary functions...
  const updateFirestore = async () => {
    await updateDoc(docRef, { fieldName: data.value });
  };
</script>
```

```
<template>
  <input v-model="data" />
</template>

<script setup lang="ts">
  // Import necessary functions...
  const data = ref("");
  const docRef = doc(db, "collectionName", "docId");
  onSnapshot(docRef, (docSnapshot) => {
    data.value = docSnapshot.data()?.fieldName;
  });
</script>
```

```
<script setup lang="ts">
  // Import necessary functions...
  const unsubscribe = onSnapshot(docRef, docSnapshot => { ... });
  onUnmounted(() => {
    unsubscribe();
  });
</script>
```

Summary

- Firebase: Cloud Firestore
 - CRUD Operations: Update
 - Update Doc (change a simple field): known Doc ID, unknown Doc ID
 - Update array field in a Document
 - CRUD Operations: Delete one Document: known Doc ID, unknown Doc ID
 - SubCollections: One-to-Many Relationship
 - Listening to Field Updates on a SINGLE doc
 - Listening to Updates on a Collection of Docs

Exercises

- What is the difference between a collection and a document?
- What is the difference between the functions addDoc() and setDoc()?
- What is the difference between the functions getDocs() and getDoc()?