CodeWithChris Firestore Cheat Sheet

Accessing the database

```
import Firebase

let db = Firestore.firestore()
```

Creating data

```
// Creates a new collection if "collection-id" doesn't already exist
let collection = db.collection("collection-id")

// Creates a new document if "document-id" doesn't already exist
collection.document("your-document")

// Creates a new document with a unique ID
collection.document()

// Creates a new document with a unique ID, sets it data to the specified value
collection.addDocument(data: [
    "string-property": "example",
    "number-property": 0
])
```

These methods all return a reference to the created value.

Updating data

```
let document = db.collection("collection-id").document("document-id")

// Overwrites the documents data with the specified fields
document.setData(documentData: ["example-key": "example"])
document.setData(documentData: [:], merge: false)

// Adds the specified data to the document's existing data
document.setData(documentData: [:], merge: true)
document.updateData([:])

// These methods can take a completion handler to deal with errors, for example
document.updateData([:]) { error in
    // Handle error appropriately
}

// You can also set data for specific keys, for example
document.setValue("value", forKey: "value-key")
document.setValuesForKeys(["value-key": "value"])
```

Reading data

```
let collection = db.collection("collection-id")

// Get a single document
collection.document("document-id").getDocument { (documentSnapshot, error) in
    // Do something with documentSnapshot's data, handle errors, etc
}

// Get all documents for a collection
collection.getDocuments { (querySnapshot, error) in
    // Access the documents of querySnapshot, handle errors, etc
}

// Get all documents for a collection now and every time a document is changed
collection.addSnapshotListener { (querySnapshot, error) in
    // Get all documents from the query
    querySnapshot?.documents
    // Get only documents that have changed since last query
    querySnapshot?.documentChanges
}
```

Deleting data

```
let document = db.collection("collection-id").document("document-id")

// Delete a field from a document
document.updateData(["key": FieldValue.delete()])

// Delete a document from a collection
document.delete()

// Specify a handler to deal with errors
document.delete() { error in
    // Handle errors appropriately
}
```

Deleting an entire collection from a client is not recommended. You can delete collections from the Cloud Firestore UI, or from a server.

Querying data

```
let collection = db.collection("collection-id")

// Get all documents who's field is in the specified list
collection.whereField("key", in: ["apples", "bananas", "cherries"])

// Similarly, all documents who's field is not in a specified list
collection.whereField("key", notIn: ["apples", "bananas", "cherries"])

// Get all documents who's array with key field contains a value
collection.whereField("key", arrayContains: "value")
collection.whereField("key", arrayContainsAny: ["value1", "value2"])

// Get all documents where field meets the criteria
collection.whereField("key", isEqualTo: "value")
collection.whereField("key", isNotEqualTo: "value")
collection.whereField("key", isLessThan: "value")
collection.whereField("key", isGreaterThan: "value")
collection.whereField("key", isLessThanOrEqualTo: "value")
collection.whereField("key", isGreaterThanOrEqualTo: "value")

// You can then perform various operations with the query, for example
collection.whereField(...).getDocuments {...}
```

Not equal queries (queries including `notIn`, `isNotEqualTo`, etc) may return many documents in a collection. You may want to limit the number of results from the query.

`in`, `notIn` and `array-contains-any` queries support up to 10 comparsion values.

Compound queries

```
let collection = db.collection("collection-id")

// Specify multiple criteria
collection
    .whereField("key1", isEqualTo: "value1")
    .whereField("key2", isEqualTo: false)

// You can perform range and not-in queries only on a single field

// ALLOWED:
collection
    .whereField("key", isGreaterThanOrEqualTo: "value")
    .whereField("key", isLessThanOrEqualTo: "value")

// NOT ALLOWED:
collection
    .whereField("key1", isGreaterThan: "value")
    .whereField("key2", isGreaterThan: "value")
```

To run a compound query, you need to create a composite index in the Firestore UI. To do this, run the query once, and when it returns with an error, select the provided link in the debug console and create the query.

Visit https://firebase.google.com/docs/firestore/query-data/indexing for more information on composite indexes.