Retrieving Data

Now that we have some data in our collection, we can retrieve that data. We'll do that through the interface in the next section, but for now, let's just retrieve the data using the console.

Inside the console, write the following:

```
PlayersList.find();
```

Here we're using the `find` function which allows us to retrieve data from a specified collection. We are not passing anything into the function, though, which means that this statement will retrieve all of the data from the collection. Still, won't receive the world's most readable output in the console. Our application can make sense of it, but we can't. To account for this, use the same function again, but this time around, attach a `fetch` function to the end of it:

```
PlayersList.find().fetch();
```

This will convert the retrieved data into an array, which is much easier for us to read.

Convert Retrieved Data into an Array

You'll notice that each document in the collection is represented by individual objects. If we click on the downward facing arrows, we can see the data associated with each document. Each document has:

- `_ID` field which contains the unique ID of the player.
- `name` field which contains the name of the player.
- `score` field, which contains the score of the player.

What if we wanted to retrieve a selection of data from the collection, rather than all of the data? To do this, pass a JSON-formatted query into the `find` function:

```
PlayersList.find({name: 'David'}).fetch();
```

This function will retrieve all of the documentsin the collection with the `name` field as equal to “David”. In our case, this will only retrieve one document, but if our collection contained multiple players named David, all of those players would be returned based on this query.

We can count the number of documents returned by the `find` function:

```
PlayersList.find().count();
```

This statement will count all of the documents in the collection.