Inserting Data into a Collection

To insert data into a collection, we have three main options:

- Through the JavaScript console.
- From inside a JavaScript file.
- Using a form in the interface.

We'll see how to use all of these options in this course, but using the JavaScript console is the simplest place to start, so that's what we'll focus on for the time being. Inside the console, write the following:

```
PlayersList.insert
```

This is the syntax we use to manipulate collections. We start with the variable assigned to a collection, and then we attach a function to it. In this case we're using the insert function but there's a range of functions available including `find`, `update`, and `remove` which we will discuss later.

When passing data into any of the Mongo functions, we need to format that data using the **JSON** syntax. If you're not familiar with JSON, this is what it looks like:

```
{
    name: 'David',
    score: 0
}
```

Keys

The data is wrapped in a pair of curly braces. We've also defined a pair of keys. These keys are known as `name` and `score`. In Mongo terminology, keys are referred to as fields. Next, we've defined the values associated with these keys. Lastly note that the keys and values are separated with commas. This is because the JSON format ignores whitespace, so commas are used to structure our data.

Insert this data into the `PlayersList` collection:

```
PlayersList.insert({
    name: 'David',
    score: 0
});
```

Because of this code, a **document** will be created inside the `playersList` collection. Documents in Mongo are equivalent to rows in SQL, and at this stage we want to create one document for every player we want in our collection.

Before we continue, insert a few more players into the collection. Define a unique value for the name field so we can distinguish between the various players in the list.

Notice that after creating each document, a random jumble of numbers and letters appears in the console. This jumble is a unique ID that's automatically created by Mongo and associated with each document. It's known as the **primary key**, and it'll be important later on.