Getting friendly with the Mongo shell

Installing & running

Assume you have MongoDB downloaded; if not head to MongoDB website and click the download button. It will take you to relevant place for your OS.

When it's downloaded installed and configured you'll need to start it running - this is all in the instructions and is different for different OS's, so we won't go into it here.

If it is running you will be able to drop into the Mongo shell with the following command:

```
mongo
```

Note how you now have a different symbol as your command prompt. You're now in the MongoDB shell!

Some basic commands

Show all databases

To see all of the databases on your machine run:

```
shows dbs
```

If you've got a new install, you'll probably just the `local` database.

Select a database

Select a database you want to run commands on

```
use local
```

Note that you get a confirmation of the switch in the shell.

Show the collections in the current database

```
show collections
```

Create a database

In MongoDB you don't really have to do anything to create a database, you just `use` it for the first time.

First let's `show dbs` to confirm the list.

Then we'll `use` a database that doesn't exist yet: `use meantest`

This switches to database `meantest` even though it doesn't technically exist yet. Run `show collections` and `show dbs` and you'll see that nothing new has been created.

So, you can `use` a database that doesn't exist yet, but it won't be created until something is added to it. So let's start by adding a collection.

Create a collection

Remember, you must `use` the database you want to create the collection in before running the command.

```
db.createCollection("tech")
```

We get a quick confirmation message, and now we can run `show dbs` and `show collections` again and see that we have now created a database with a collection.

Let's add some data.

Create a document

To add a document we need to `insert` it in the collection in the current active database. As we've seen before the document is a JSON object.

```
db.tech.insert(
  {
    name : "MongoDB",
    role : "Database
  }
)
```

You can add this line by line into the shell and it won't run until the final closing bracket is added. When it runs you should see a confirmation message of the number of docs inserted.

```
WriteResult({ "nInserted" : 1 })
```

That's good, but how do we see it? By running a basic query on the database.

Finding all docs in a collection

For a quick start we'll ask Mongo to show us all of the documents in the collection with a simple command.

```
db.tech.find()
```

That will return the document like this:

```
{ "_id" : ObjectId("56b280da091ed27a0f4a731c"), "name" : "MongoDB", "role" : "Database" }
```

We can see the data in there, including the `_id` field created by MongoDB, as we discussed in the previous video.

Making the output more readable

We can format the output of the `find` operation to make it a bit more readable by chaining a `pretty()` method to the end of the query method like this:

```
db.tech.find().pretty()
```

And now the output looks like this:

```
{
    "_id" : ObjectId("56b280da091ed27a0f4a731c"),
    "name" : "MongoDB",
    "role" : "Database"
}
```

This isn't a big deal right now, but is very handy when your documents are much larger and more complex.

Adding multiple documents to a collection

To insert multiple documents to a collection you run the same `insert` method, but pass it an array of objects instead.

```
db.tech.insert(
  [
    {
      name : "Node.js",
      role : "Platform"
    },
    {
      name : "Express",
      role : "Web application server"
    },
    {
      name : "Angular",
      role : "Front-end framework"
    }
  ]
)
```

Note that this time we get more confirmation from in the shell.

Validating the insert worked as planned

Find all documents: `db.tech.find()`

```
{ "_id" : ObjectId("56b280da091ed27a0f4a731c"), "name" : "MongoDB", "role" : "Database" }
{ "_id" : ObjectId("56b28448091ed27a0f4a731d"), "name" : "Node.js", "role" : "Platform" }
{ "_id" : ObjectId("56b28448091ed27a0f4a731e"), "name" : "Express", "role" : "Web applicati
{ "_id" : ObjectId("56b28448091ed27a0f4a731f"), "name" : "Angular", "role" : "Front-end fra
```

Or make it pretty: `db.tech.find().pretty()`

```
{
    "_id" : ObjectId("56b280da091ed27a0f4a731c"),
    "name" : "MongoDB",
    "role" : "Database"
}
{
    "_id" : ObjectId("56b28448091ed27a0f4a731d"),
    "name" : "Node.js",
    "role" : "Platform"
}
{
    "_id" : ObjectId("56b28448091ed27a0f4a731e"),
    "name" : "Express",
    "role" : "Web application server"
}
{
    "_id" : ObjectId("56b28448091ed27a0f4a731f"),
    "name" : "Angular",
    "role" : "Front-end framework"
}
```
