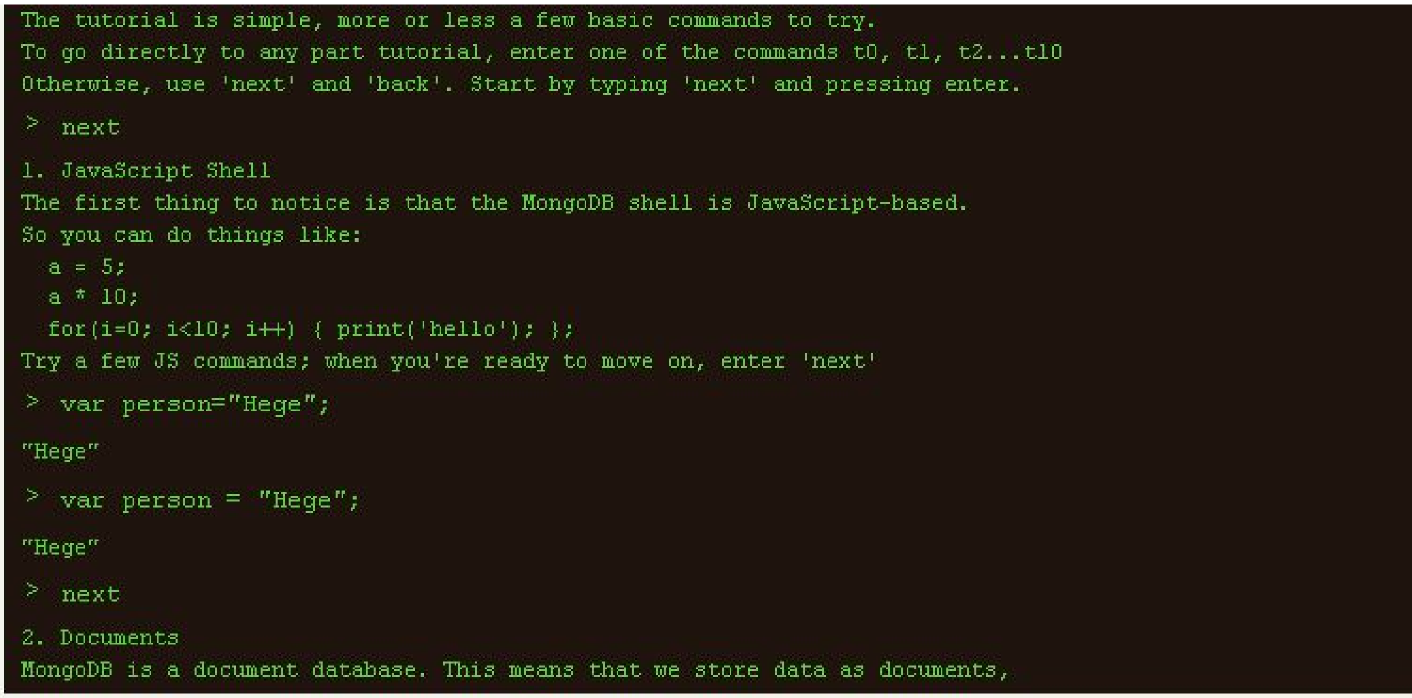
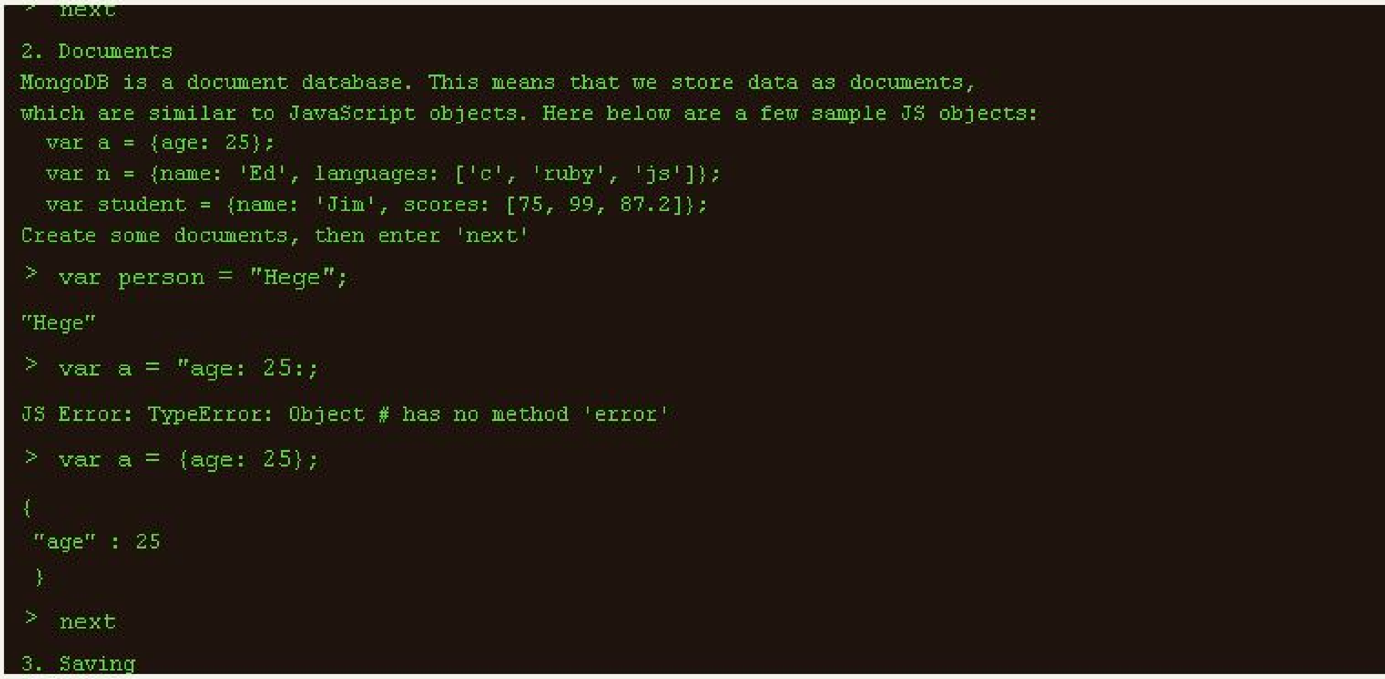
JESUS AMBRIZ

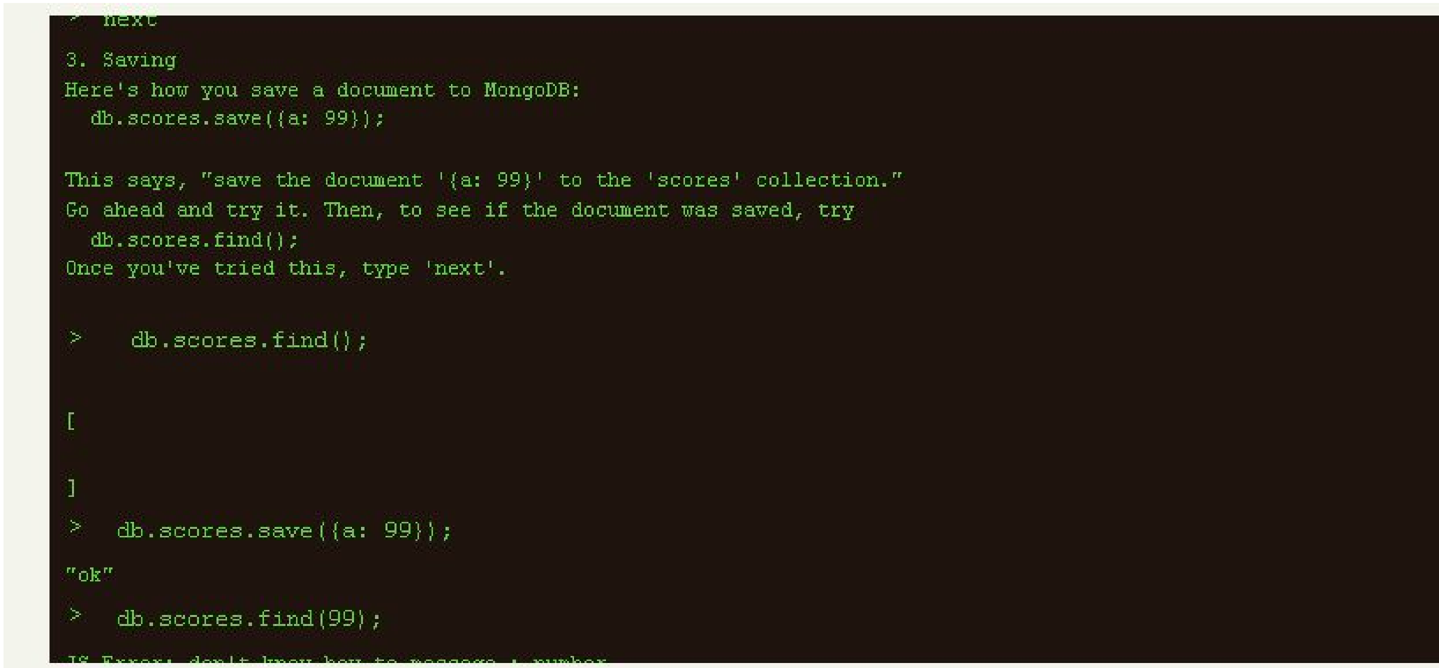


```
The tutorial is simple, more or less a few basic commands to try.
To go directly to any part tutorial, enter one of the commands t0, t1, t2...t10
Otherwise, use 'next' and 'back'. Start by typing 'next' and pressing enter.
> next
1. JavaScript Shell
The first thing to notice is that the MongoDB shell is JavaScript-based.
So you can do things like:
  a = 5;
  a * 10;
  for(i=0; i<10; i++) { print('hello'); };
Try a few JS commands; when you're ready to move on, enter 'next'
> var person="Hege";
"Hege"
> var person = "Hege";
"Hege"
> next
2. Documents
MongoDB is a document database. This means that we store data as documents,
```

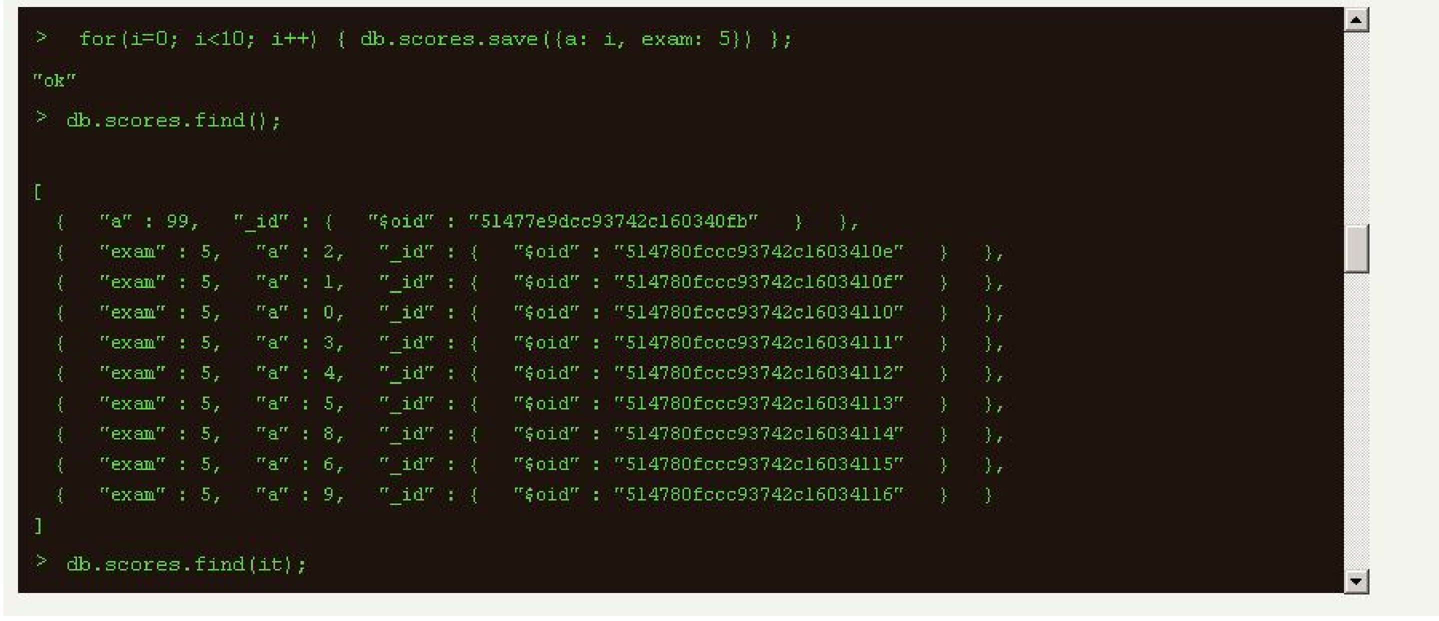


```
> next
2. Documents
MongoDB is a document database. This means that we store data as documents,
which are similar to JavaScript objects. Here below are a few sample JS objects:
  var a = {age: 25};
  var n = {name: 'Ed', languages: ['c', 'ruby', 'js']};
  var student = {name: 'Jim', scores: [75, 99, 87.2]};
Create some documents, then enter 'next'
> var person = "Hege";
"Hege"
> var a = "age: 25:;
JS Error: TypeError: Object # has no method 'error'
> var a = {age: 25};
{
 "age" : 25
 }
> next
3. Saving
```

JESUS AMBRIZ

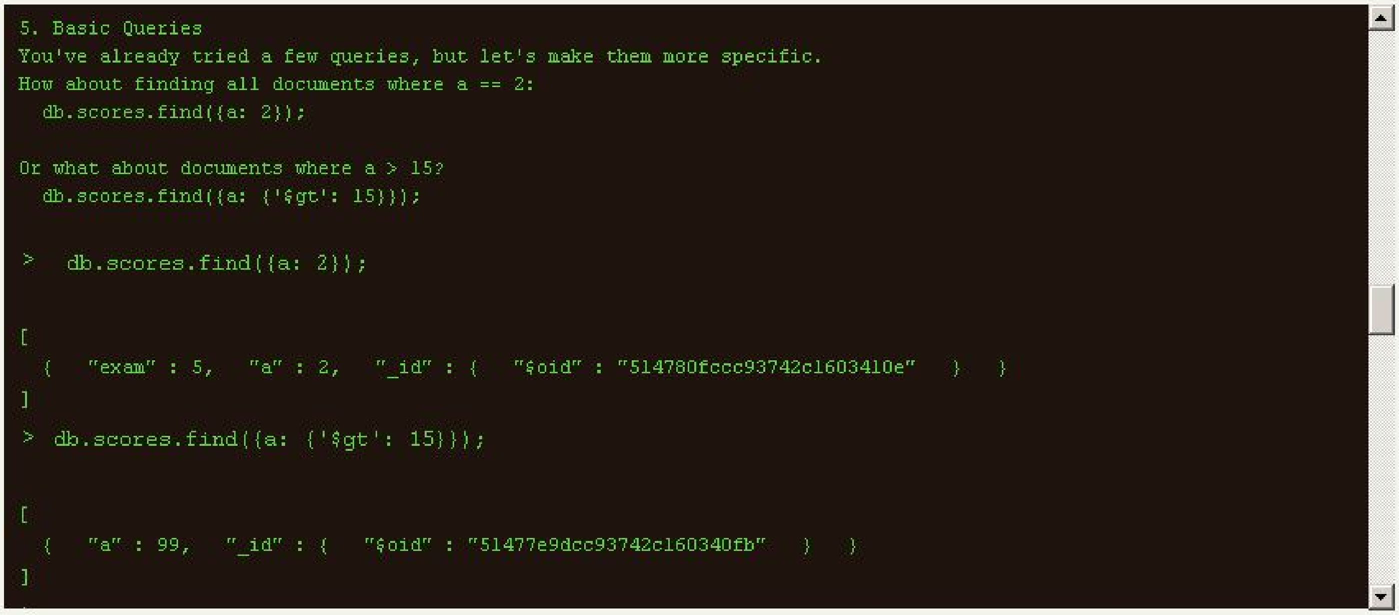


```
> next
3. Saving
Here's how you save a document to MongoDB:
  db.scores.save({a: 99});

This says, "save the document '{a: 99}' to the 'scores' collection."
Go ahead and try it. Then, to see if the document was saved, try
  db.scores.find();
Once you've tried this, type 'next'.


>   db.scores.find();


[


]
>  db.scores.save({a: 99});
"ok"
>  db.scores.find(99);
JS Error: don't know how to massage : number
```

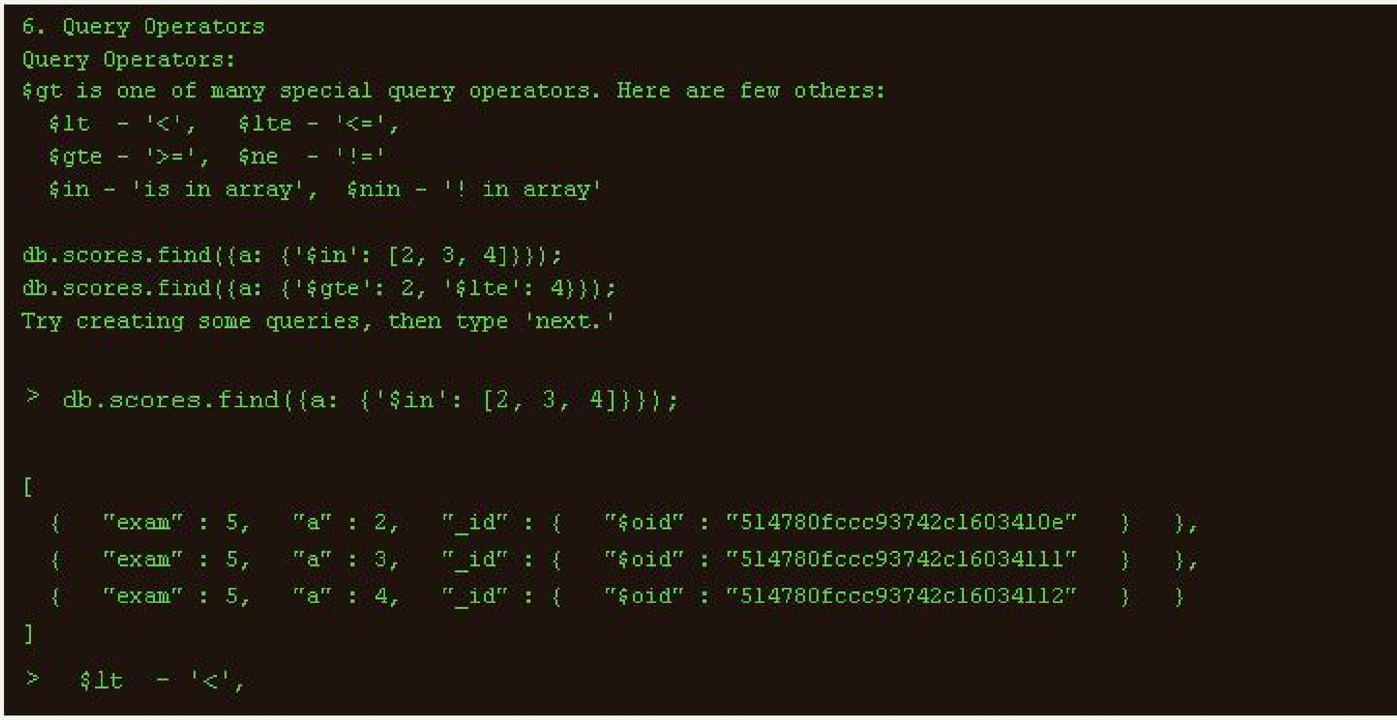


```
>  for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
"ok"
> db.scores.find();


[
  {   "a" : 99,   "_id" : {   "$oid" : "51477e9dcc93742c160340fb"   }   },
  {   "exam" : 5,   "a" : 2,   "_id" : {   "$oid" : "514780fccc93742c1603410e"   }   },
  {   "exam" : 5,   "a" : 1,   "_id" : {   "$oid" : "514780fccc93742c1603410f"   }   },
  {   "exam" : 5,   "a" : 0,   "_id" : {   "$oid" : "514780fccc93742c16034110"   }   },
  {   "exam" : 5,   "a" : 3,   "_id" : {   "$oid" : "514780fccc93742c16034111"   }   },
  {   "exam" : 5,   "a" : 4,   "_id" : {   "$oid" : "514780fccc93742c16034112"   }   },
  {   "exam" : 5,   "a" : 5,   "_id" : {   "$oid" : "514780fccc93742c16034113"   }   },
  {   "exam" : 5,   "a" : 8,   "_id" : {   "$oid" : "514780fccc93742c16034114"   }   },
  {   "exam" : 5,   "a" : 6,   "_id" : {   "$oid" : "514780fccc93742c16034115"   }   },
  {   "exam" : 5,   "a" : 9,   "_id" : {   "$oid" : "514780fccc93742c16034116"   }   }
]
> db.scores.find(it);
```

JESUS AMBRIZ

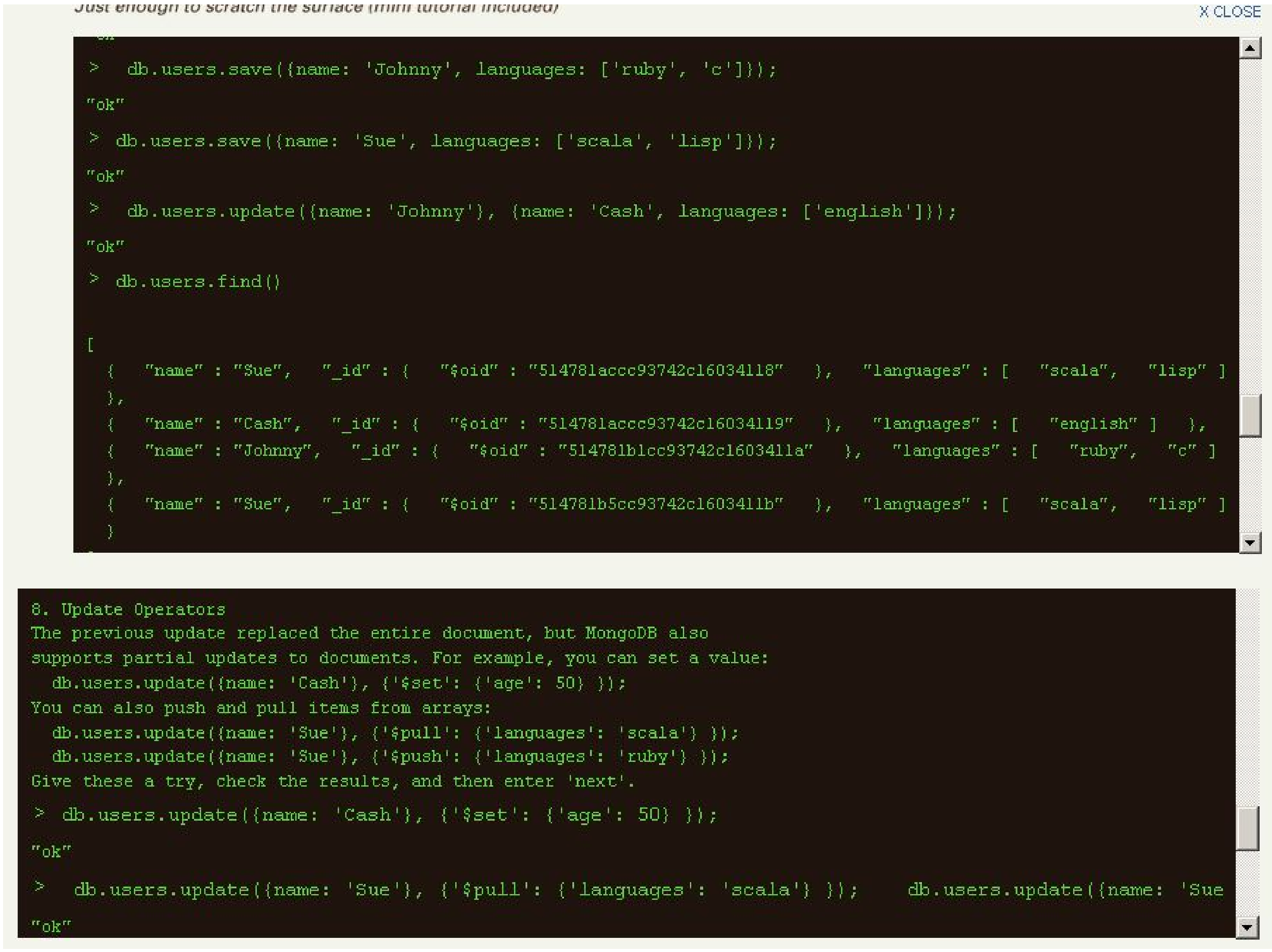


```
5. Basic Queries
You've already tried a few queries, but let's make them more specific.
How about finding all documents where a == 2:
  db.scores.find({a: 2});

Or what about documents where a > 15?
  db.scores.find({a: {'$gt': 15}});

>   db.scores.find({a: 2});


[
  {   "exam" : 5,   "a" : 2,   "_id" : {   "$oid" : "514780fccc93742c1603410e"   }   }
]
> db.scores.find({a: {'$gt': 15}});


[
  {   "a" : 99,   "_id" : {   "$oid" : "51477e9dcc93742c160340fb"   }   }
]
```



```
6. Query Operators
Query Operators:
$gt is one of many special query operators. Here are few others:
  $lt  - '<',   $lte - '<=',
  $gte - '>=',  $ne  - '!='
  $in - 'is in array',  $nin - '! in array'

db.scores.find({a: {'$in': [2, 3, 4]}});
db.scores.find({a: {'$gte': 2, '$lte': 4}});
Try creating some queries, then type 'next.'


>  db.scores.find({a: {'$in': [2, 3, 4]}});


[
  {   "exam" : 5,   "a" : 2,   "_id" : {   "$oid" : "514780fccc93742c1603410e"   }   },
  {   "exam" : 5,   "a" : 3,   "_id" : {   "$oid" : "514780fccc93742c16034111"   }   },
  {   "exam" : 5,   "a" : 4,   "_id" : {   "$oid" : "514780fccc93742c16034112"   }   }
]
>   $lt  - '<',
```

JESUS AMBRIZ



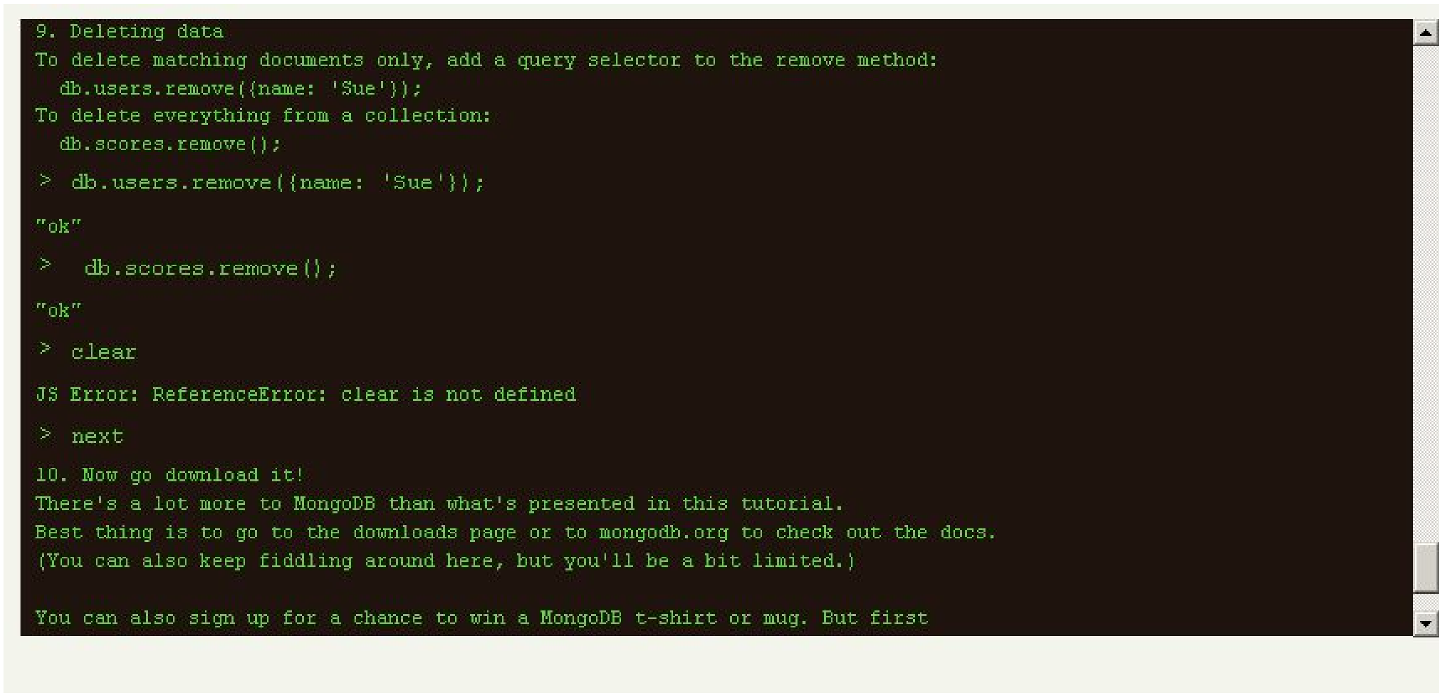
Just enough to scratch the surface (mini tutorial included)

X CLOSE

```
>  db.users.save({name: 'Johnny', languages: ['ruby', 'c']});
"ok"
> db.users.save({name: 'Sue', languages: ['scala', 'lisp']});
"ok"
>  db.users.update({name: 'Johnny'}, {name: 'Cash', languages: ['english']});
"ok"
> db.users.find()

[
  {   "name" : "Sue",   "_id" : {   "$oid" : "51478laccc93742cl6034ll8"   },   "languages" : [   "scala",   "lisp" ]
  },
  {   "name" : "Cash",   "_id" : {   "$oid" : "51478laccc93742cl6034ll9"   },   "languages" : [   "english" ]   },
  {   "name" : "Johnny",   "_id" : {   "$oid" : "514781blcc93742cl60341la"   },   "languages" : [   "ruby",   "c" ]
  },
  {   "name" : "Sue",   "_id" : {   "$oid" : "514781b5cc93742cl60341lb"   },   "languages" : [   "scala",   "lisp" ]
  }
```

```
8. Update Operators
The previous update replaced the entire document, but MongoDB also
supports partial updates to documents. For example, you can set a value:
  db.users.update({name: 'Cash'}, {'$set': {'age': 50} });
You can also push and pull items from arrays:
  db.users.update({name: 'Sue'}, {'$pull': {'languages': 'scala'} });
  db.users.update({name: 'Sue'}, {'$push': {'languages': 'ruby'} });
Give these a try, check the results, and then enter 'next'.
> db.users.update({name: 'Cash'}, {'$set': {'age': 50} });
"ok"
>  db.users.update({name: 'Sue'}, {'$pull': {'languages': 'scala'} });    db.users.update({name: 'Sue
"ok"
```

JESUS AMBRIZ



```
9. Deleting data
To delete matching documents only, add a query selector to the remove method:
  db.users.remove({name: 'Sue'});
To delete everything from a collection:
  db.scores.remove();

>  db.users.remove({name: 'Sue'});

"ok"

>   db.scores.remove();

"ok"

>  clear

JS Error: ReferenceError: clear is not defined

>  next

10. Now go download it!
There's a lot more to MongoDB than what's presented in this tutorial.
Best thing is to go to the downloads page or to mongodb.org to check out the docs.
(You can also keep fiddling around here, but you'll be a bit limited.)

You can also sign up for a chance to win a MongoDB t-shirt or mug. But first
```