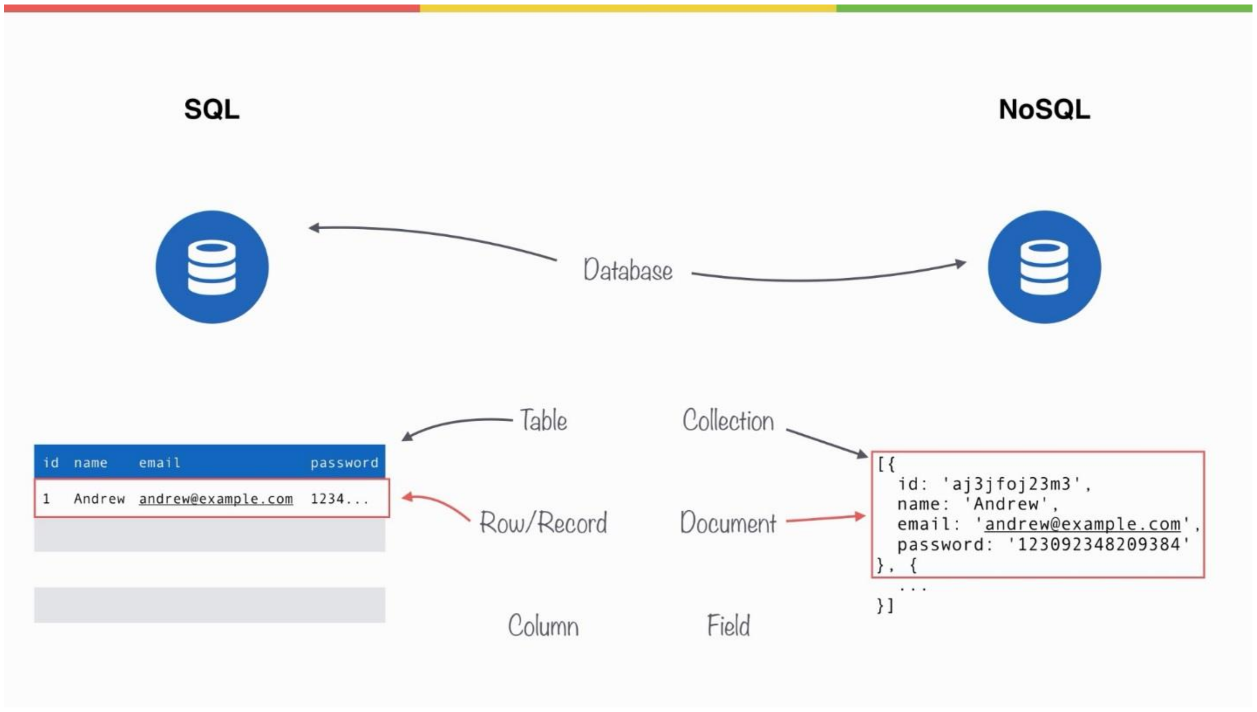
MongoDB

NoSQL Database

What is it
Install
npm install mongodb@2.2.5
References
/http://mongodb.github.io/node-mongodb-native/2.2/api http://mongodb.github.io/node-mongodb-native/2.2/quick-start/quick-start/ https://docs.mongodb.com/

Created by: Wael Zoaiter - Github

SQL vs NoSQL



In **NoSQL** every application has a **Collection** of users for instance, and some other collections ..

And every collection consist of **Documents** for every individual user ..

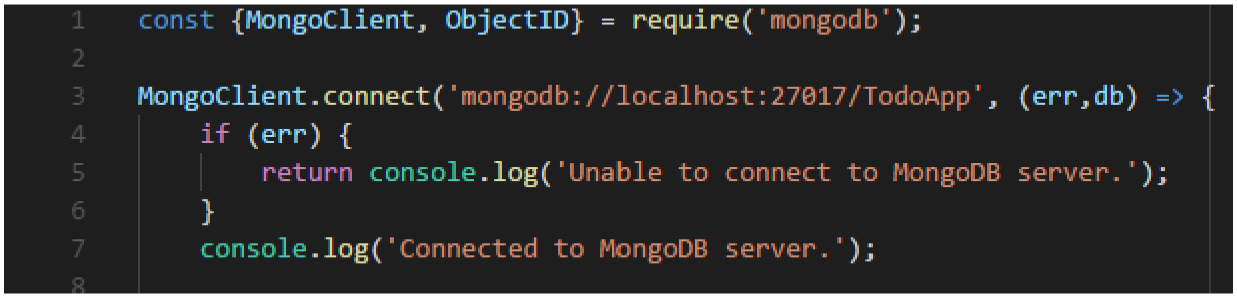
And inside of each Document there is some **Fields** "name field, email field, etc".

In **SQL** every application has a **Table** of users for instance, and some other tables ..

And every table consist of **Records/Rows** for every individual user ..

And inside of each Record there is some **Columns** "name column, email column, etc".

Initilaize

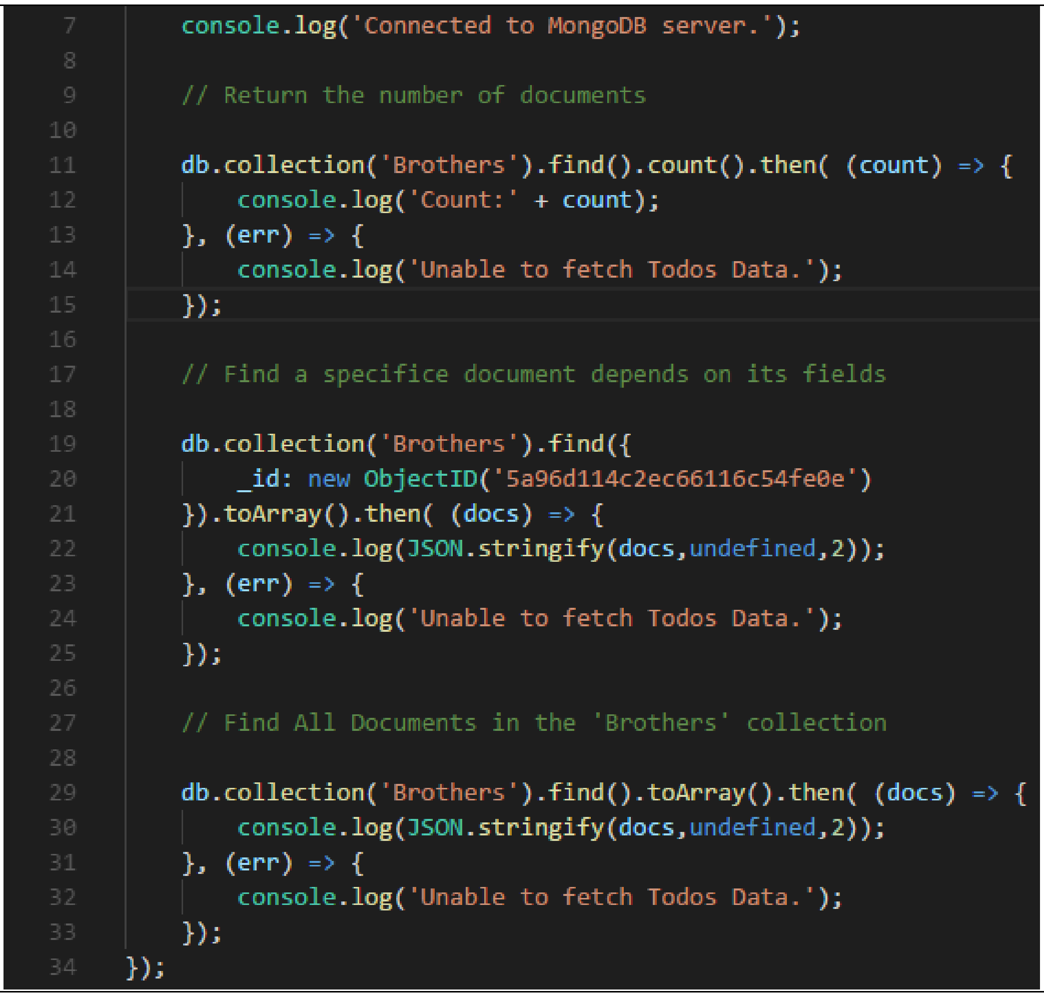


```
1  const {MongoClient, ObjectID} = require('mongodb');
2
3  MongoClient.connect('mongodb://localhost:27017/TodoApp', (err,db) => {
4      if (err) {
5          return console.log('Unable to connect to MongoDB server.');
6      }
7      console.log('Connected to MongoDB server.');
8
```

#1: Get properties from MongoDB library and use it as a variables.

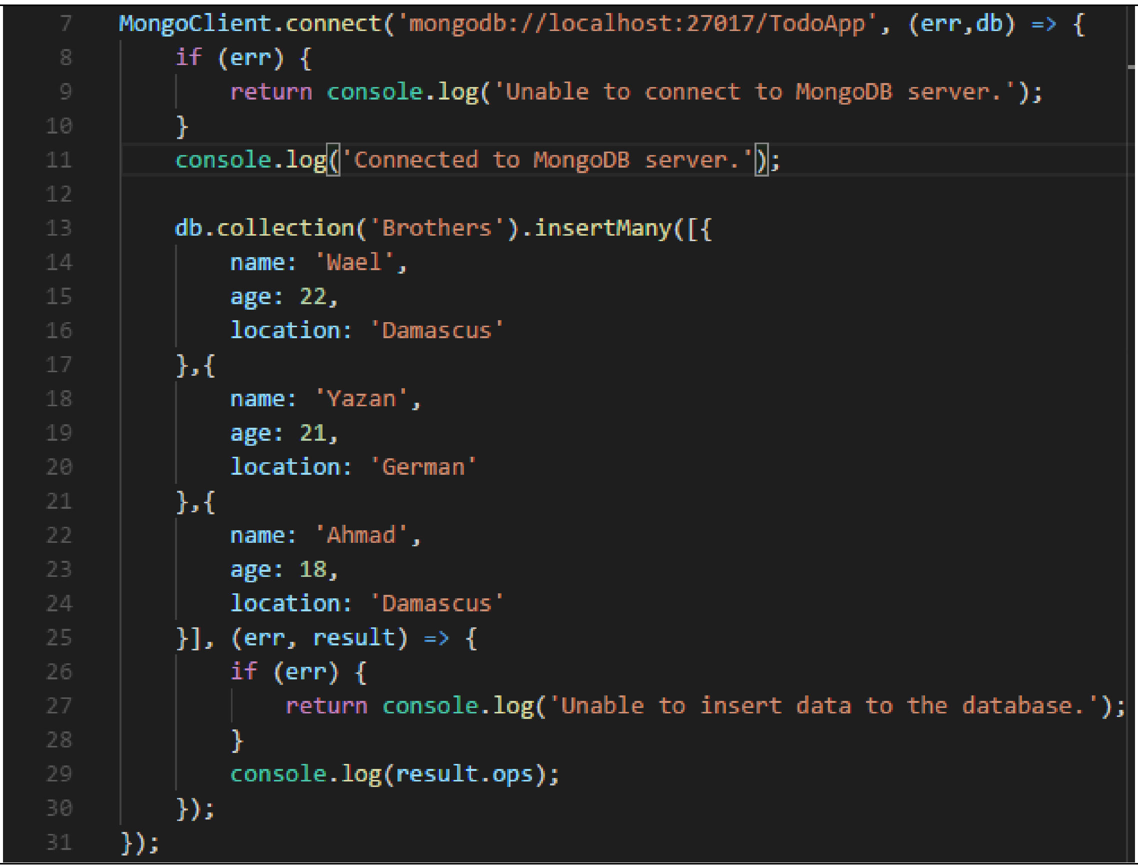
#3: Connect to MongoDB Server with URL, PORT and collection name .. then execute the function.

Fetch/Find Documents



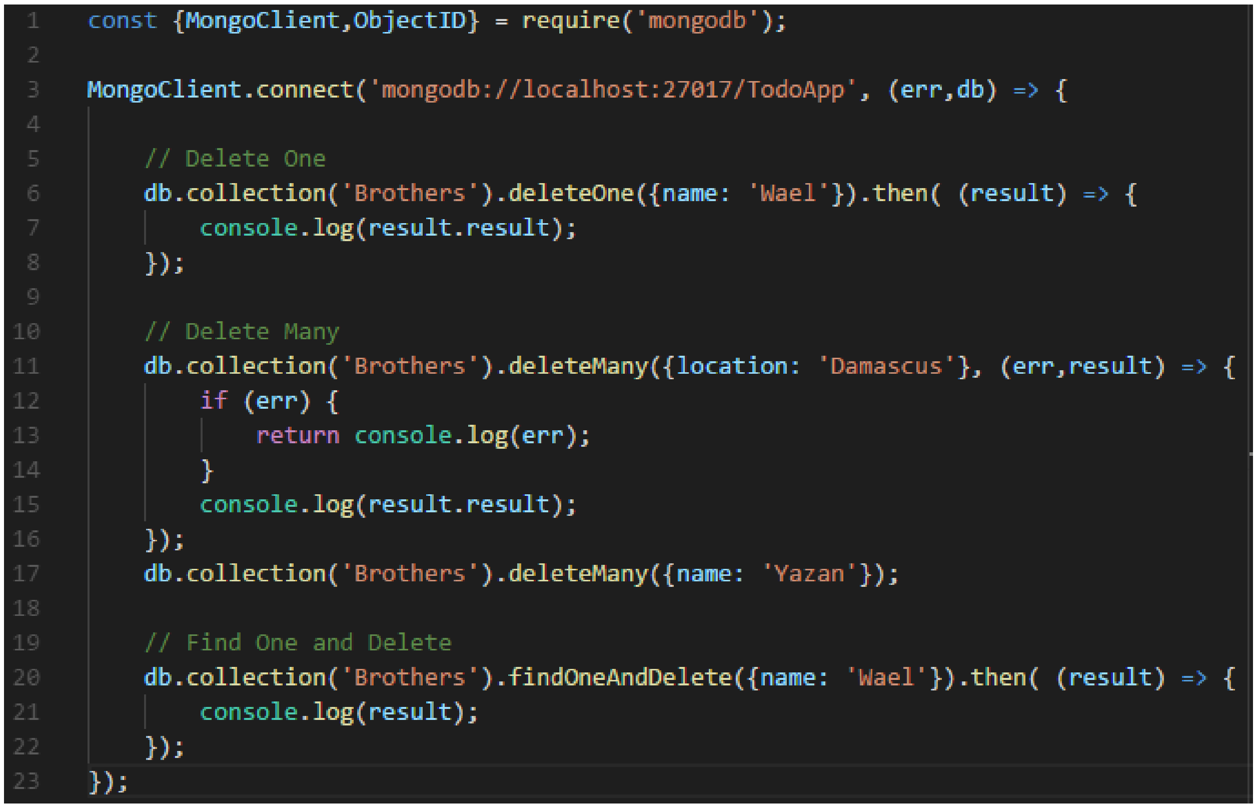
```
    console.log('Connected to MongoDB server.');

    // Return the number of documents

    db.collection('Brothers').find().count().then( (count) => {
        console.log('Count:' + count);
    }, (err) => {
        console.log('Unable to fetch Todos Data.');
    });

    // Find a specifice document depends on its fields

    db.collection('Brothers').find({
        _id: new ObjectID('5a96d114c2ec66116c54fe0e')
    }).toArray().then( (docs) => {
        console.log(JSON.stringify(docs,undefined,2));
    }, (err) => {
        console.log('Unable to fetch Todos Data.');
    });

    // Find All Documents in the 'Brothers' collection

    db.collection('Brothers').find().toArray().then( (docs) => {
        console.log(JSON.stringify(docs,undefined,2));
    }, (err) => {
        console.log('Unable to fetch Todos Data.');
    });
});
```

Insert Documents



```
MongoClient.connect('mongodb://localhost:27017/TodoApp', (err,db) => {
    if (err) {
        return console.log('Unable to connect to MongoDB server.');
    }
    console.log('Connected to MongoDB server.');

    db.collection('Brothers').insertMany([{
        name: 'Wael',
        age: 22,
        location: 'Damascus'
    },{
        name: 'Yazan',
        age: 21,
        location: 'German'
    },{
        name: 'Ahmad',
        age: 18,
        location: 'Damascus'
    }], (err, result) => {
        if (err) {
            return console.log('Unable to insert data to the database.');
        }
        console.log(result.ops);
    });
});
```

Delete Documents

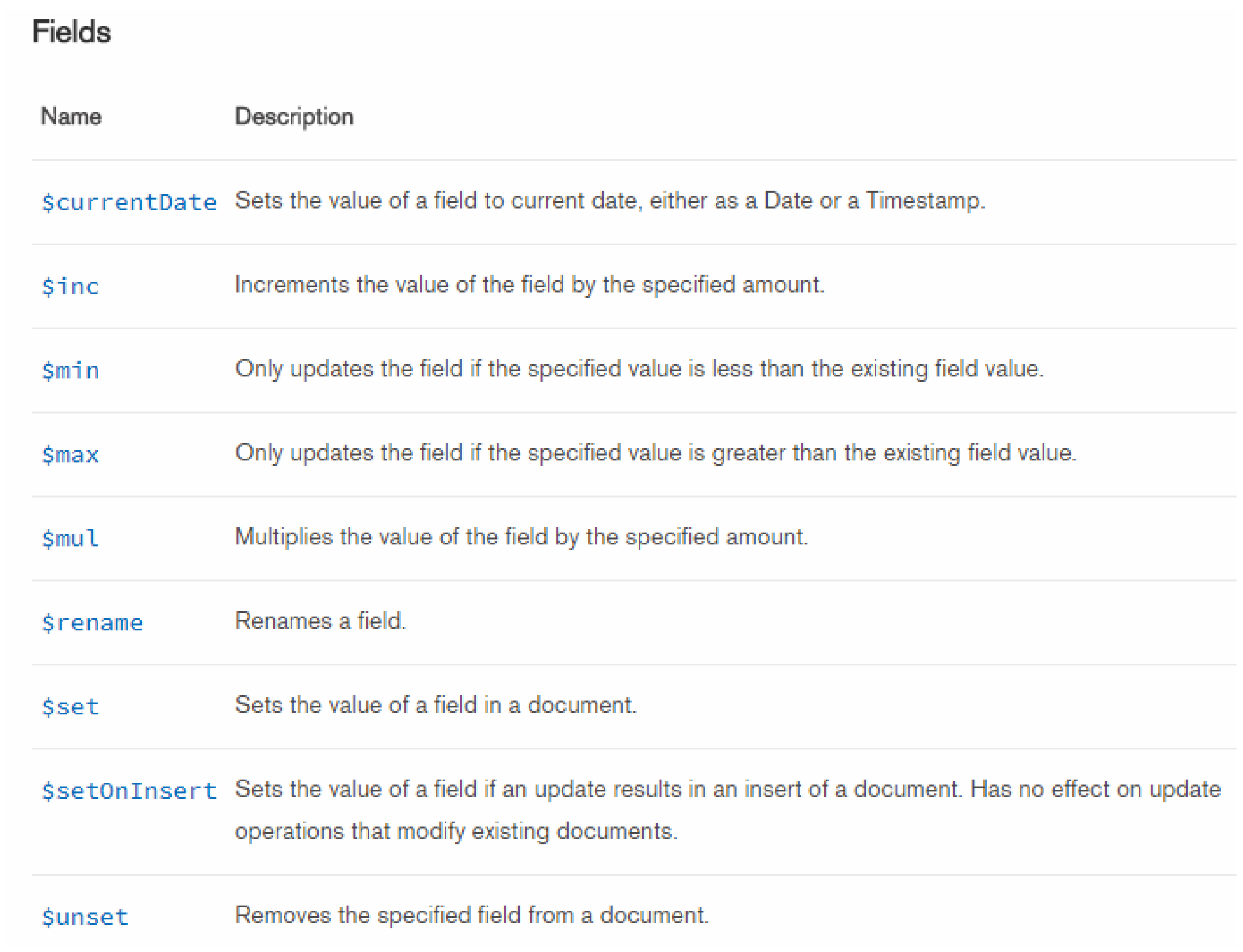


```
const {MongoClient,ObjectID} = require('mongodb');

MongoClient.connect('mongodb://localhost:27017/TodoApp', (err,db) => {

    // Delete One
    db.collection('Brothers').deleteOne({name: 'Wael'}).then( (result) => {
        console.log(result.result);
    });

    // Delete Many
    db.collection('Brothers').deleteMany({location: 'Damascus'}, (err,result) => {
        if (err) {
            return console.log(err);
        }
        console.log(result.result);
    });
    db.collection('Brothers').deleteMany({name: 'Yazan'});

    // Find One and Delete
    db.collection('Brothers').findOneAndDelete({name: 'Wael'}).then( (result) => {
        console.log(result);
    });
});
```

Created by: Wael Zoaiter - Github

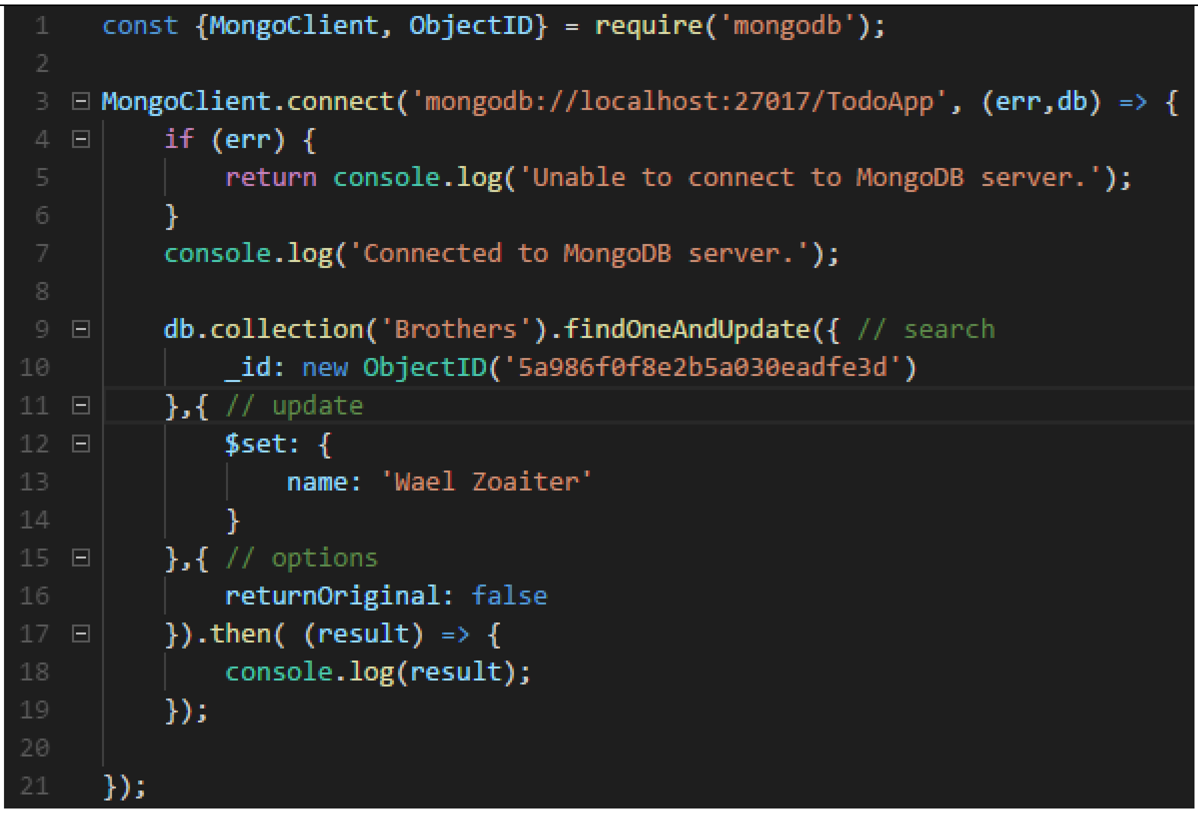
Update Documents

/https://docs.mongodb.com/manual/reference/operator/update



Fields

Name	Description
`$currentDate`	Sets the value of a field to current date, either as a Date or a Timestamp.
`$inc`	Increments the value of the field by the specified amount.
`$min`	Only updates the field if the specified value is less than the existing field value.
`$max`	Only updates the field if the specified value is greater than the existing field value.
`$mul`	Multiplies the value of the field by the specified amount.
`$rename`	Renames a field.
`$set`	Sets the value of a field in a document.
`$setOnInsert`	Sets the value of a field if an update results in an insert of a document. Has no effect on update operations that modify existing documents.
`$unset`	Removes the specified field from a document.



```
const {MongoClient, ObjectID} = require('mongodb');

MongoClient.connect('mongodb://localhost:27017/TodoApp', (err,db) => {
    if (err) {
        return console.log('Unable to connect to MongoDB server.');
    }
    console.log('Connected to MongoDB server.');

    db.collection('Brothers').findOneAndUpdate({ // search
        _id: new ObjectID('5a986f0f8e2b5a030eadfe3d')
    },{ // update
        $set: {
            name: 'Wael Zoaiter'
        }
    },{ // options
        returnOriginal: false
    }).then( (result) => {
        console.log(result);
    });

});
```

Mongoos.js

MongoDB Module

Install

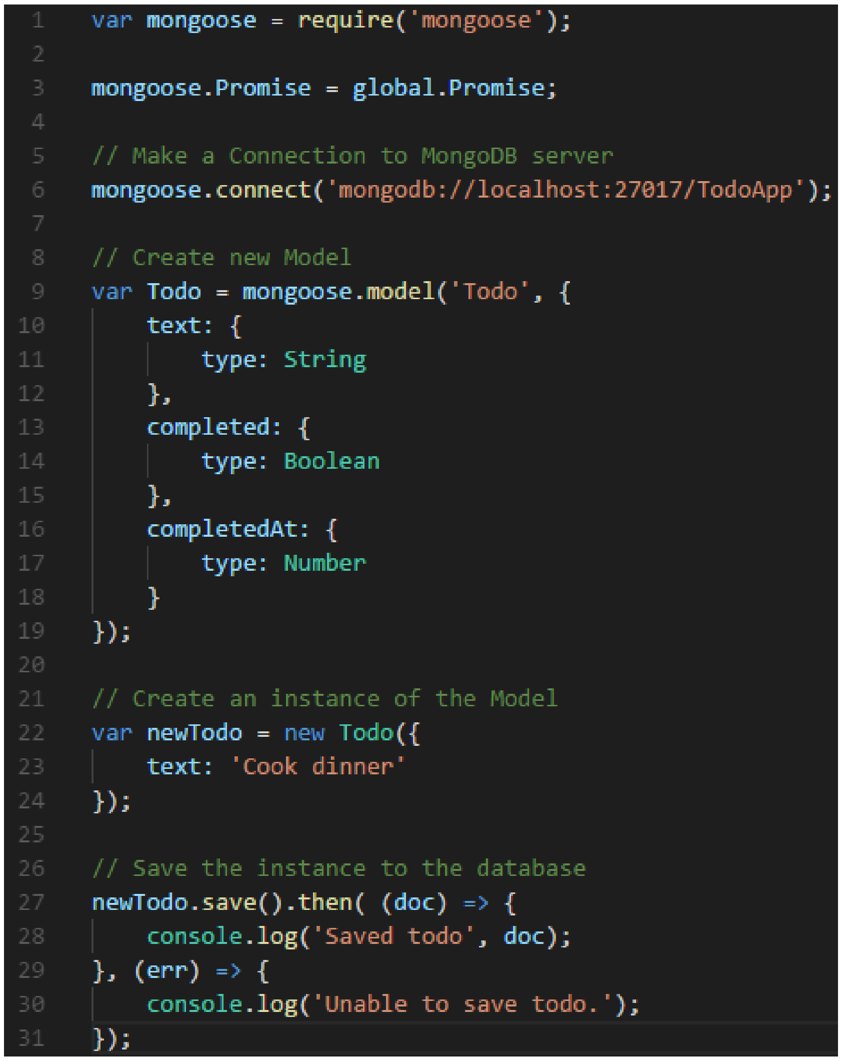
```
npm install mongoose@4.5.9
```

References

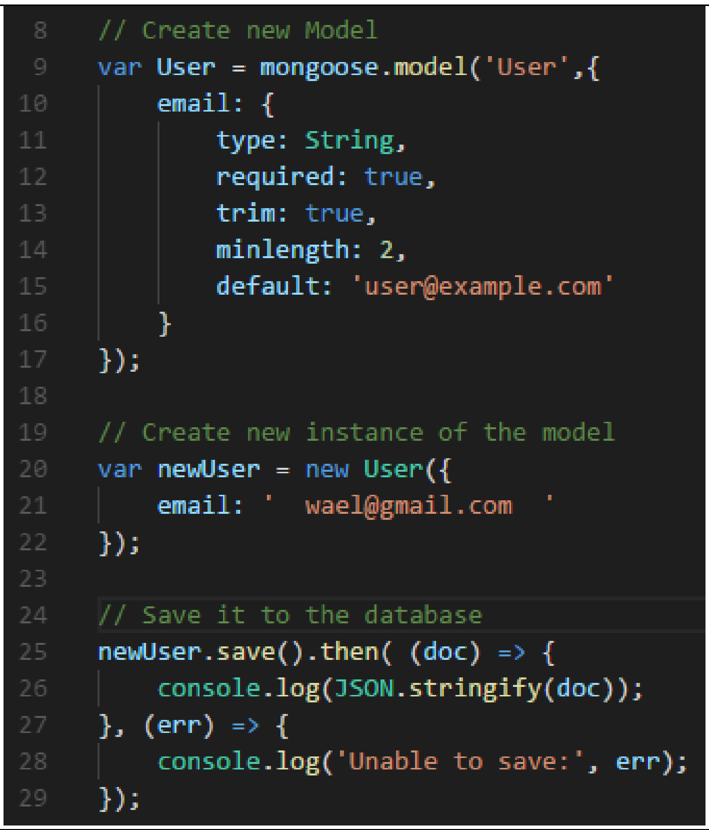
http://mongoosejs.com/docs/guide.html

Created by: Wael Zoaiter - Github

Setting Up

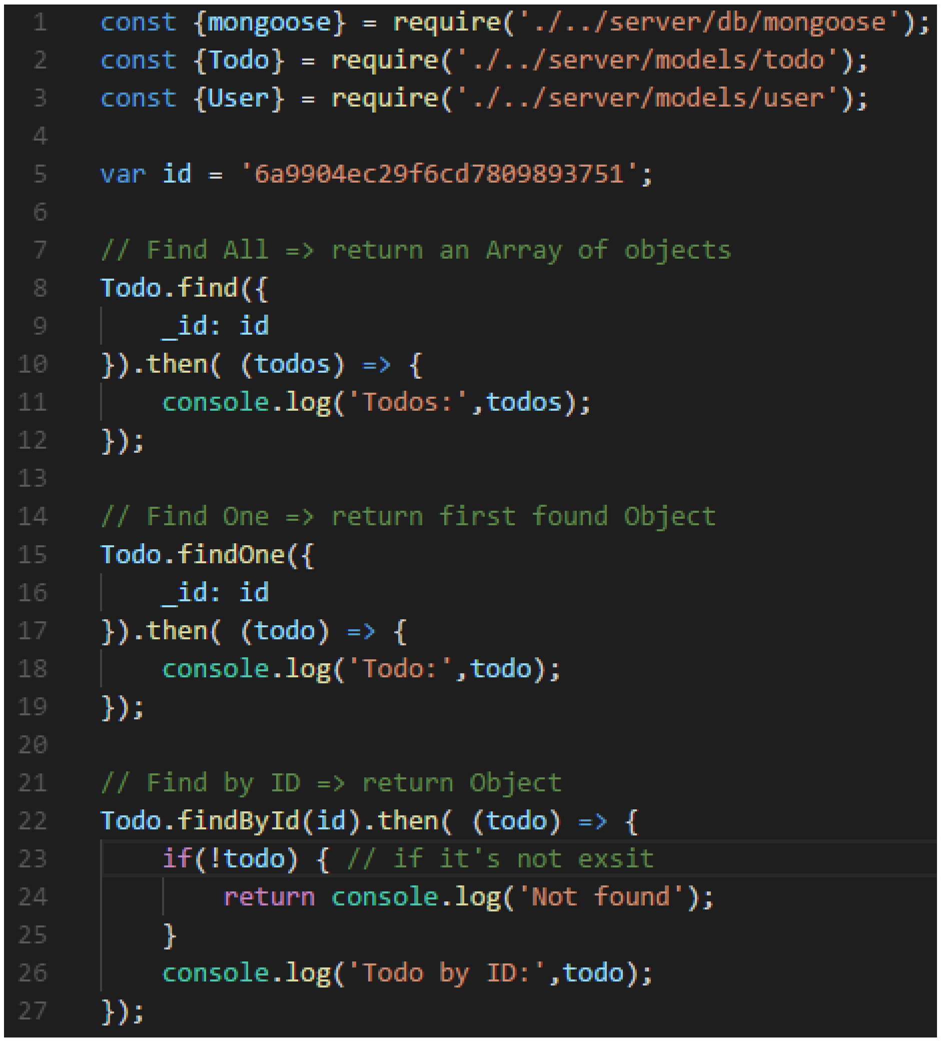


```javascript
var mongoose = require('mongoose');

mongoose.Promise = global.Promise;

// Make a Connection to MongoDB server
mongoose.connect('mongodb://localhost:27017/TodoApp');

// Create new Model
var Todo = mongoose.model('Todo', {
    text: {
        type: String
    },
    completed: {
        type: Boolean
    },
    completedAt: {
        type: Number
    }
});

// Create an instance of the Model
var newTodo = new Todo({
    text: 'Cook dinner'
});

// Save the instance to the database
newTodo.save().then( (doc) => {
    console.log('Saved todo', doc);
}, (err) => {
    console.log('Unable to save todo.');
});
```



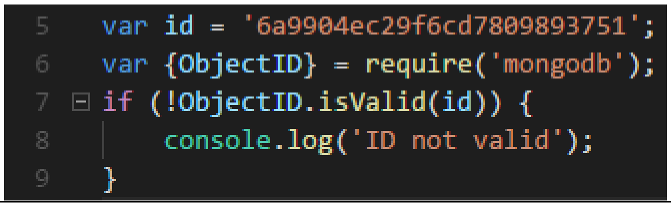
```
// Create new Model
var User = mongoose.model('User',{
    email: {
        type: String,
        required: true,
        trim: true,
        minlength: 2,
        default: 'user@example.com'
    }
});

// Create new instance of the model
var newUser = new User({
    email: '  wael@gmail.com  '
});

// Save it to the database
newUser.save().then( (doc) => {
    console.log(JSON.stringify(doc));
}, (err) => {
    console.log('Unable to save:', err);
});
```

Mongoose Queries (Find)



```
const {mongoose} = require('./../server/db/mongoose');
const {Todo} = require('./../server/models/todo');
const {User} = require('./../server/models/user');

var id = '6a9904ec29f6cd7809893751';

// Find All => return an Array of objects
Todo.find({
    _id: id
}).then( (todos) => {
    console.log('Todos:',todos);
});

// Find One => return first found Object
Todo.findOne({
    _id: id
}).then( (todo) => {
    console.log('Todo:',todo);
});

// Find by ID => return Object
Todo.findById(id).then( (todo) => {
    if(!todo) { // if it's not exsit
        return console.log('Not found');
    }
    console.log('Todo by ID:',todo);
});
```

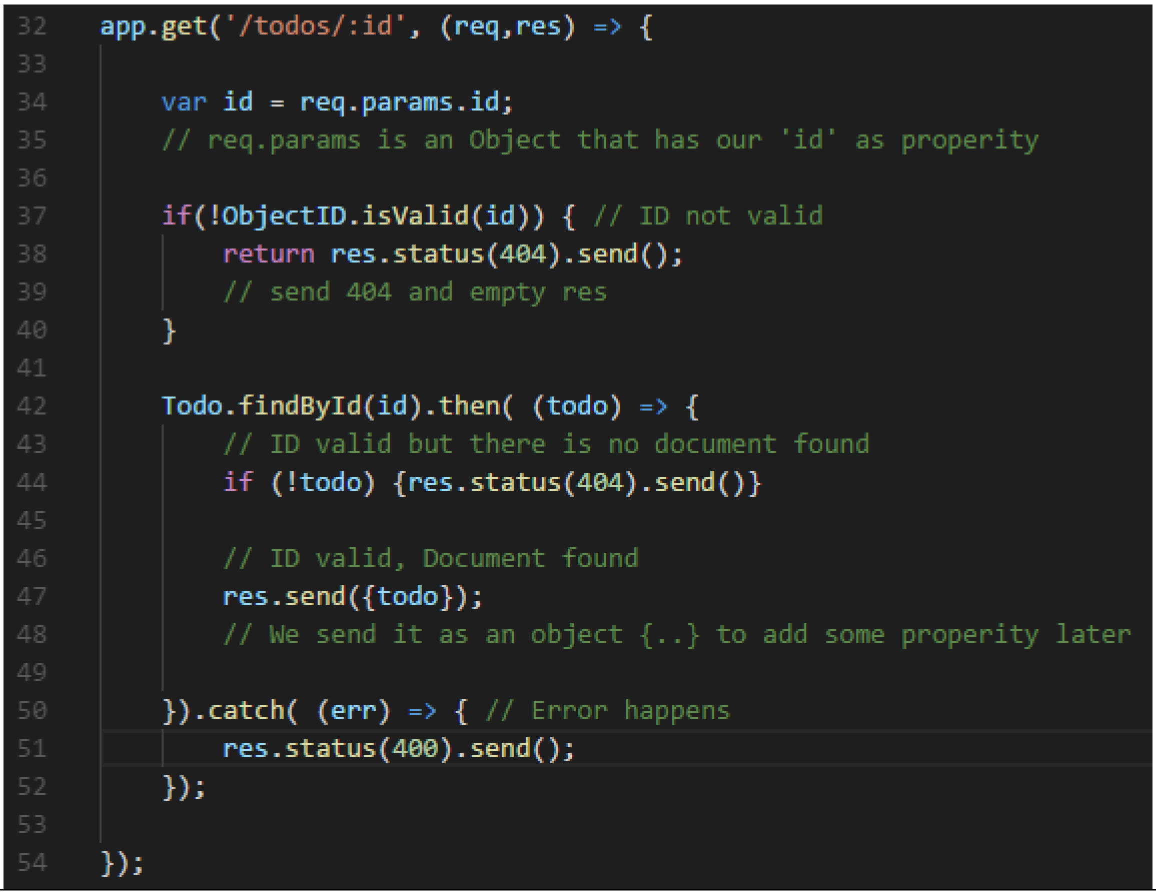
Valid ID



```
var id = '6a9904ec29f6cd7809893751';
var {ObjectID} = require('mongodb');
if (!ObjectID.isValid(id)) {
    console.log('ID not valid');
}
```

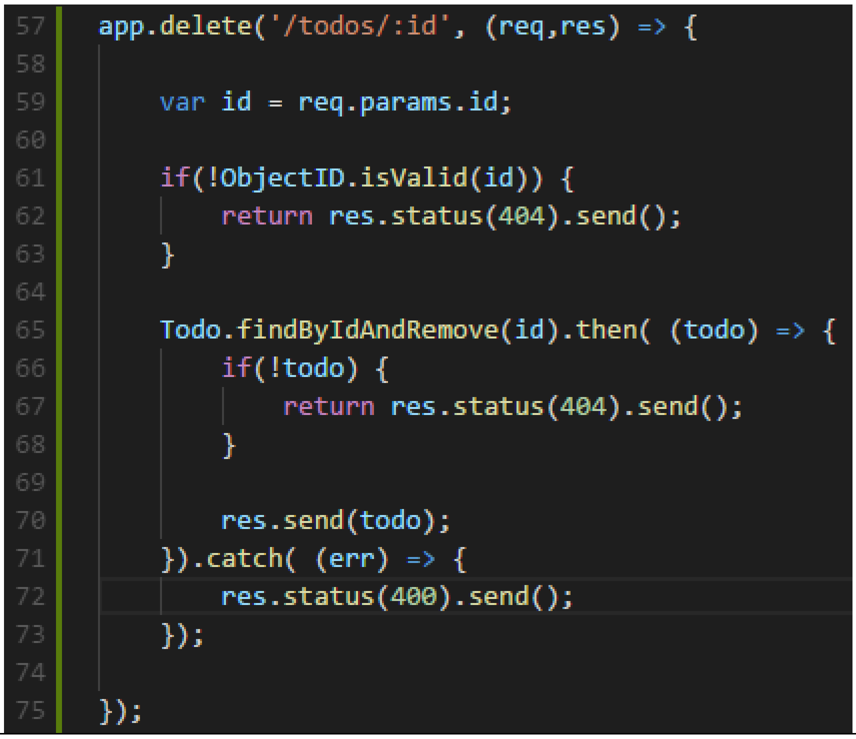
Created by: Wael Zoaiter - Github

Params



```
app.get('/todos/:id', (req,res) => {

    var id = req.params.id;
    // req.params is an Object that has our 'id' as properity

    if(!ObjectID.isValid(id)) { // ID not valid
        return res.status(404).send();
        // send 404 and empty res
    }

    Todo.findById(id).then( (todo) => {
        // ID valid but there is no document found
        if (!todo) {res.status(404).send()}

        // ID valid, Document found
        res.send({todo});
        // We send it as an object {..} to add some properity later

    }).catch( (err) => { // Error happens
        res.status(400).send();
    });

});
```

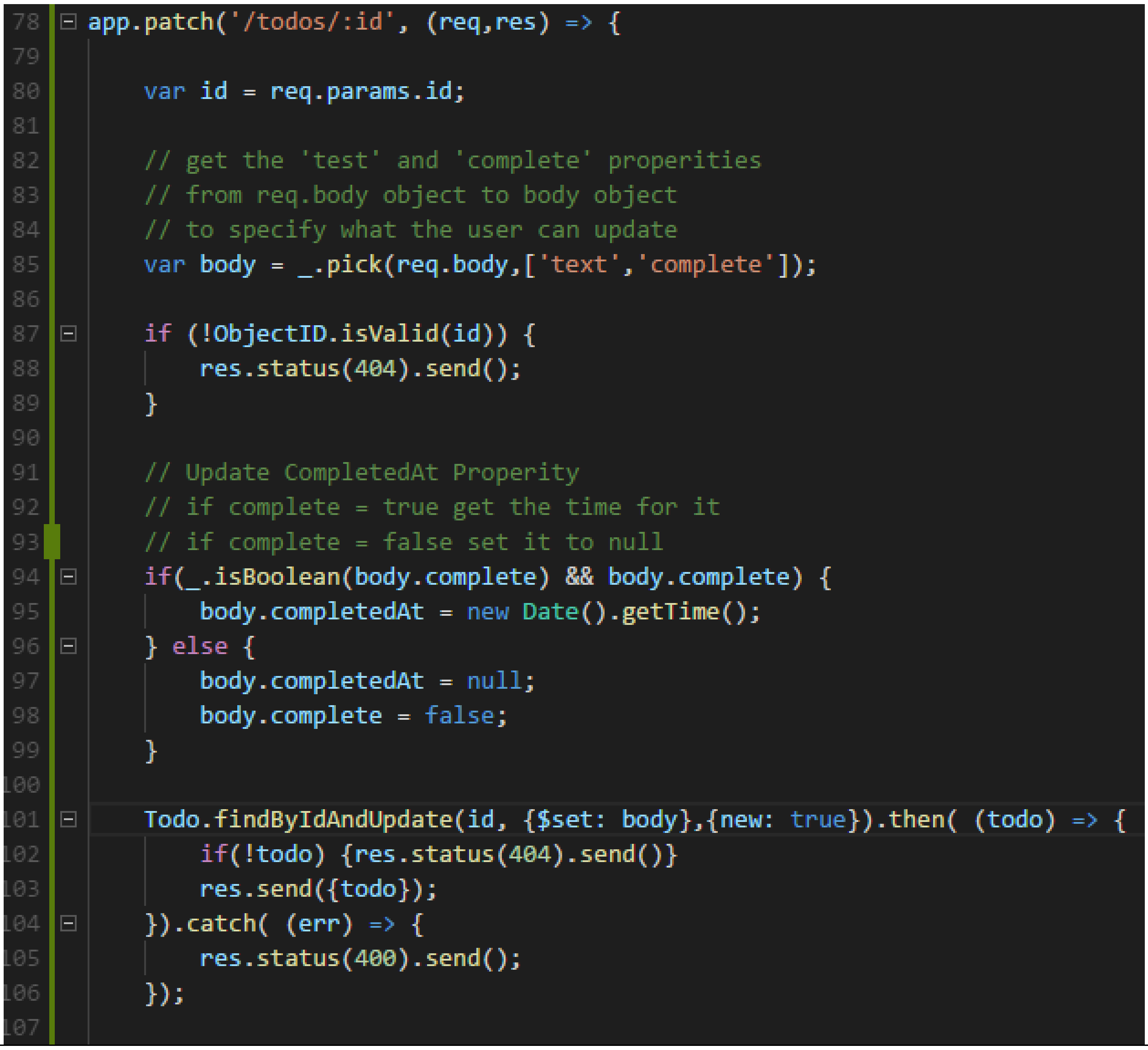
Delete Documents



```
app.delete('/todos/:id', (req,res) => {

    var id = req.params.id;

    if(!ObjectID.isValid(id)) {
        return res.status(404).send();
    }

    Todo.findByIdAndRemove(id).then( (todo) => {
        if(!todo) {
            return res.status(404).send();
        }

        res.send(todo);
    }).catch( (err) => {
        res.status(400).send();
    });

});
```

Created by: Wael Zoaiter - Github

Update Documents



```
app.patch('/todos/:id', (req,res) => {

    var id = req.params.id;

    // get the 'test' and 'complete' properities
    // from req.body object to body object
    // to specify what the user can update
    var body = _.pick(req.body,['text','complete']);

    if (!ObjectID.isValid(id)) {
        res.status(404).send();
    }

    // Update CompletedAt Property
    // if complete = true get the time for it
    // if complete = false set it to null
    if(_.isBoolean(body.complete) && body.complete) {
        body.completedAt = new Date().getTime();
    } else {
        body.completedAt = null;
        body.complete = false;
    }

    Todo.findByIdAndUpdate(id, {$set: body},{new: true}).then( (todo) => {
        if(!todo) {res.status(404).send()}
        res.send({todo});
    }).catch( (err) => {
        res.status(400).send();
    });

```

Created by: Wael Zoaiter - Github