* to delete entire database
use fruitsDB // switch to the database
db.dropDatabase()

* Use Mongoose in database
@ npm i mongoose

* Schema: blueprint or structure of how it would look like

```
* const mongoose = require('mongoose');
mongoose.connect("mongodb://localhost:2701/fruitsDB"
,{useNewUrlParser:true});
```
name of database

```
const fruitSchema = new mongoose.Schema({
    name: String,
    rating: Number,
    review: String,
});
// creating a mongoose model
const Fruit = mongoose.model("Fruit", fruitSchema);
```
name of the collection
* will automatically be converted to (fruits)

```
document
const fruit = new Fruit({
    name: "Apple",
    rating: 7,
    review: "Good",
})
fruit.save(); will be saved in Fruits collection in
FruitsDB database
```

```
shopDB  0.000GB
> show dbs
admin     0.000GB
config    0.000GB
fruitsDB  0.000GB
local     0.000GB
shopDB    0.000GB
> use fruitsDB
switched to db fruitsDB
> show collections
fruits
```

* const personSchema = new mongoose.Schema ({
    name: String,
    age: Number,
    gender: String,
});

document
const Person = mongoose.model ("Person", personSchema);
const person = new Person ({
    name: John,
    age: 20,
    gender: male,
});
person.save();

whenever you run node app.js People will be saved again & again.

* Model.insertMany()
allows us to insert an array of documents (objects) in a collection

documents to add
Fruit.insertMany ([kiwi, orange, banana], function (err) {
name of model
    if (err) { console.log ("error") };
    else { console.log ("successfully saved") };
});

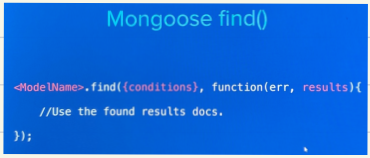
const kiwi = new Fruit ({name: "Kiwi", rating: 10, review: "Very good"})

// to find

model ← what you want to find

```
Fruit.find({}, function (err, fruits) {
    if (err) {console.log (err);}
    else {console.log (fruits);}
});
```

here we are finding the whole collection



- fruits.forEach ((fruit) => {
    console.log (fruit.name)
  });

at the end to close the database connection

mongoose.connection.close ();

```
Fruit.find(function(err, fruits){
  if (err) {
    console.log(err);
  } else {

    mongoose.connection.close();

    fruits.forEach(function(fruit){
      console.log(fruit.name);
    });
  }
});
```

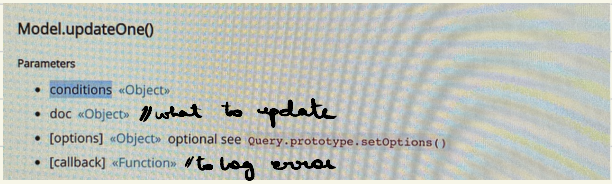
Data Validation

```
const fruitSchema = new mongoose.Schema ({
  name: String,
  rating: {
    type: Number,
    min: 1,
    max: 10
  },
  review: String
});

const Fruit = mongoose.model("Fruit", fruitSchema);

const fruit = new Fruit ({
  name: "Apple",
  rating: 34,
  review: "Pretty solid as a fruit."
});

fruit.save();

const personSchema = new mongoose.Schema ({
  name: String,
  age: Number
});
```

```
const FruitsSchema = new mongoose.Schema ({
    name: {
        type: String,
        required: [true, "Please specify name"]
    },
    rating: {
        type: Number,
        min: 1,
        max: 10,
    }
    review: String
});
```

if no name is entered

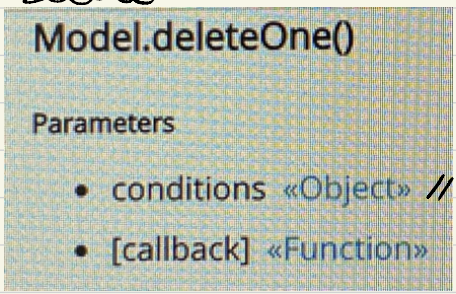
Updating & Deleting in Mongoose



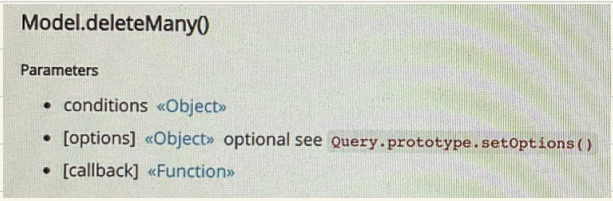

```
Fruit.updateOne({id: "123456"}, {name: "Peach"},
function (err) {
  if (err) {console.log (err)}
  else {console.log ("success")};
});
```

will add if not there and update if there

Delete




```
Fruit.deleteOne({name:"Peach"}, function (err) {
  if (err) {console.log (err);}
  else { console.log ("sucess")}
});
```



model

```
Person.deleteMany({name: "John"}, function(err){
  if(err){console.log(err)};
  else{console.log("success")};
```

Relationships and embedding document

```
const personSchema = new mongoose.Schema ({
  name: String,
  age:  Number,
  favouriteFruit: fruitSchema
});

const Person = mongoose.model("Person", personSchema);
```

```
const pineapple = new Fruit({
  name: "Pineapple",
  score: 9,
  review: "Great fruit."
});

pineapple.save();

const person = new Person({
  name: "Amy",
  age: 12,
  favouriteFruit: pineapple
});

person.save();
```

```
const mango = new Fruit({
  name: "Mango",
  score: 6,
  review: "Decent fruit."
});

mongo.save();

Person.updateOne({name: "John"}, {favouriteFruit: mango}, function(err){
  if (err) {
    console.log(err);
  } else {
    console.log("Succesfully updated the document");
  }
})
```

log(number) : number

Adding Database to our Todo-List

1. npm install
2. npm i mongoose
3. const mongoose = require('mongoose');
4. mongoose.connect("mongodb://localhost:27017/todolist", {useNewURLParser: true});
5. const itemsSchema = {
   name: String
   };
6. const Item = mongoose.model("Item", itemsSchema);

Mongoose Schema

```
const <schemaName> = {
    <fieldName> : <FieldDataType>,
    ...
}
```

Mongoose Model

```
const = mongoose.model(
    <"SingularCollectionName">,
    <schemaName>
);
```

Mongoose Document

```
const <constantName> = new <ModelName> ({
    <fieldName> : <fieldData>,
    ...
});
```

```
const item1 = new Item({
  name: "Welcome to your todolist!"
});

const item2 = new Item({
  name: "Hit the + button to add a new item."
});

const item3 = new Item({
  name: "<-- Hit this to delete an item."
});

const defaultItems = [item1, item2, item3];
```

```
Item.insertMany(defaultItems, function(err){
  if (err) {
    console.log(err);
  } else {
    console.log("Successfully savevd default items to DB.");
  }
});
```

Mongoose insertMany()

```
<ModelName>.insertMany(<documentArray>, function(err){
    //Deal with error or log success.
});
```

* find is used to retrieve multiple documents that matches the specified query criteria
* findOne is used to retrieve single document which occurs first that matches the specified query criteria.
* cd over to the exact file
* node app.js

* Mongoose find()

```
<ModelName>.find({conditions}, function(err, results){
    //Use the found results docs.
});
```

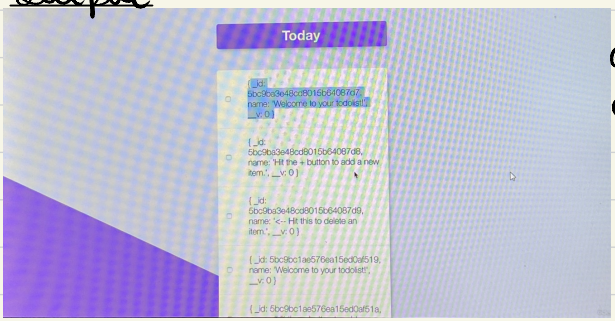
*
```
app.get("/", function(req, res) {
  Item.find({}, function(err, foundItems){
    console.log(foundItems);
  });
  res.render("list", {listTitle: "Today", newListItems: items});
});
```
↳ will contain all the items which we have found

will throw an error because items is not defined

* start nodemon app.js

*
```
app.get("/", function(req, res) {
  Item.find({}, function(err, foundItems){
    res.render("list", {listTitle: "Today", newListItems: foundItems});
  });
});
```

Output



2 Problems
1. will save everytime we run nodemon
2. cannot access the value

* rs // to restart the server

1. in list.ejs

```
<%- include("header") -%>

<div class="box" id="heading">
  <h1> <%= listTitle %> </h1>
</div>

<div class="box">
  <% for (let i=0; i<newListItems.length; i++) { %>
    <div class="item">
      <input type="checkbox">
      <p><%= newListItems[i].name %></p>
    </div>
  <% } %>

  <form class="item" action="/" method="post">
    <input type="text" name="newItem" placeholder="New Item" autocomplete="off">
    <button type="submit" name="list">+</button>
  </form>
</div>

<%- include("footer") -%>
```

or

```
<% newListItems.forEach(function(item){ %>
    <div class="item">
      <input type="checkbox">
      <p><%= item.name </p>
    </div>
<% }) %>
```

* db which db we are using

db.dropDatabase() delete database

```
app.get("/", function(req, res) {

  Item.find({}, function(err, foundItems){

    if (foundItems.length === 0) {
      Item.insertMany(defaultItems, function(err){
        if (err) {
          console.log(err);
        } else {
          console.log("Successfully savevd default items to DB.");
        }
      });
    } else {
      res.render("list", {listTitle: "Today", newListItems: foundItems});
    }
  });

});
```

here we are adding items only if the are no items present in the array

but there is a problem we are not rendering the added items

* insertMany returns an array of object

Solution

```
app.get("/", function(req, res) {

  Item.find({}, function(err, foundItems){

    if (foundItems.length === 0) {
      Item.insertMany(defaultItems, function(err){
        if (err) {
          console.log(err);
        } else {
          console.log("Successfully savevd default items to DB.");
        }
      });
      res.redirect("/");
    } else {
      res.render("list", {listTitle: "Today", newListItems: foundItems});
    }
  });

});
```

here if length = 0 than it will add items and redirect again "/" but this time it will not have length = 0 and will go to else statement

* To add data in the todo-list

```
app.post("/", function(req, res){

  const itemName = req.body.newItem;

  const item = new Item({
    name: itemName
  });

  item.save();


});
```

this will add the newItem in database but will not show on website

```
app.post("/", function(req, res){

  const itemName = req.body.newItem;

  const item = new Item({
    name: itemName
  });

  item.save();

  res.redirect("/");


});
```

will show the new item on website

* in style.CSS

.form-item

* in list.ejs

```
<%  newListItems.forEach(function(item){ %>

  <form action="/delete" method="post">
    <div class="item">
      <input type="checkbox" name="checkbox" onChange="this.form.submit()">
      <p><%=item.name%></p>
    </div>

  </form>
<%  }) %>
```

value = "<%= item._id %>"

now in app.js it will console id of the item on which checkbox is clicked

```
app.post("/delete", function(req, res){
  console.log(req.body.checkbox);
});
```

* to delete an item

Mongoose findByIdAndRemove()

* you have to give callback or else it wont be executed

```
<ModelName>.findByIdAndRemove(<Id>, function(err){

    //Handle any errors or log success.

});
```

```
app.post("/delete", function(req, res){
  const checkedItemId = req.body.checkbox;

  Item.findByIdAndRemove(checkedItemId, function(err){
    if (!err) {
      console.log("Successfully deleted checked item.");
      res.redirect("/");
    }
  });
});
```

* delete app.get /work

Express Route Parameters

```
app.get("/category/:<paramName>", function(req, res){

    //Access req.params.paramName

});
```

```
app.get("/:customListName", function(req, res){
const customListName = req.
params.customListName;
});
```

* const listSchema = {
    name: String,
    items: [itemsSchema],
};
const List = mongoose.model("List", listSchema);

```
app.get("/:customListName", function(req, res){
  const customListName = req.params.customListName;

  const list = new List({
    name: customListName,
    items: defaultItems → already created this
  });

  list.save();

});
```

Problem: whenever we enter a route a list will will be created with routes name.

* find returns an array back
* findOne returns an object back

* Mongoose findOne()

```
<ModelName>.find({conditions}, function(err, results){

    //Use the found results docs.

});
```

to find if the document with a given name already exists or not

```
List.findOne({name:customListName},function(err, foundlists){
    if (!error) {
        if (!foundList){
            console.log("Doesn't exist");
        }else { console.log("Exists"); }
```

*

```
app.get("/:customListName", function(req, res){
  const customListName = req.params.customListName;

  List.findOne({name: customListName}, function(err, foundList){
    if (!err){
      if (!foundList){
        //Create a new list
        const list = new List({
          name: customListName,
          items: defaultItems
        });

        list.save();
      } else {
        //Show an existing list

        res.render("list", {listTitle: foundList.name, newListItems: foundList.items});
      }
    }
  });
```

Problem
will leave the user hanging after 1 attempt as it will be save

*

```
app.get("/:customListName", function(req, res){
  const customListName = req.params.customListName;

  List.findOne({name: customListName}, function(err, foundList){
    if (!err){
      if (!foundList){
        //Create a new list
        const list = new List({
          name: customListName,
          items: defaultItems
        });
        list.save();
        res.redirect("/" + customListName);
      } else {
        //Show an existing list

        res.render("list", {listTitle: foundList.name, newListItems: foundList.items});
      }
    }
  });
```

Problem
Whatever we post will be posted to the "/" route from any route

In list.ejs

```ejs
<form class="item" action="/" method="post">
  <input type="text" name="newItem" placeholder="New Item" autocomplete="off">
  <button type="submit" name="list" value="<%= listTitle %>">+</button>
</form>
```

In app.js

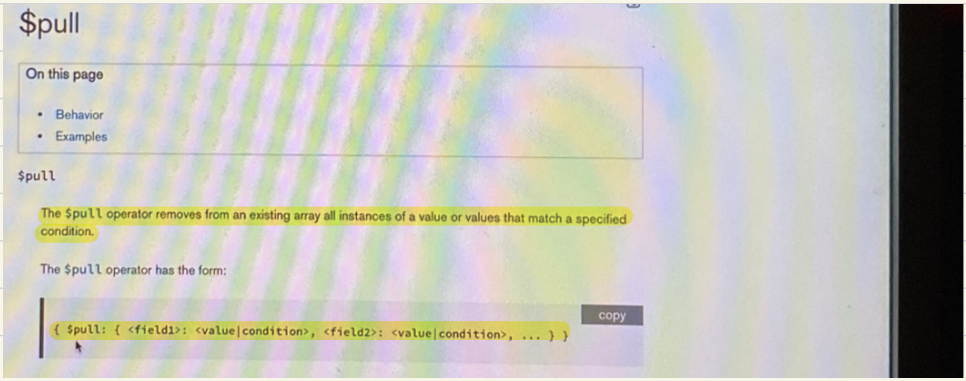
```js
app.post("/", function(req, res){

  const itemName = req.body.newItem;
  const listName = req.body.list;

  const item = new Item({
    name: itemName
  });

  item.save();
  res.redirect("/");
});
```

```js
app.post("/", function(req, res){

  const itemName = req.body.newItem;
  const listName = req.body.list;

  const item = new Item({
    name: itemName
  });

  if (listName === "Today"){
    item.save();
    res.redirect("/");
  } else {
    List.findOne({name: listName}, function(err, foundList){
      foundList.items.push(item);
      foundList.save();
      res.redirect("/" + listName);
    });
  }
});
```

```js
const listSchema = {
  name: String,
  items: [itemsSchema]
};
```

Problem: when deleting an item from specific route, it redirects to the "/" route

```
<%  newListItems.forEach(function(item){ %>

  <form action="/delete" method="post">
    <div class="item">
      <input type="checkbox" name="checkbox" value="<%=item._id%>" onChange="this.
      <p><%=item.name%></p>
    </div>
    <input type="hidden" name="listName" value="<%= listTitle %>"></input>
  </form>
<%  }) %>
```



Mongoose findOneAndUpdate()

```
<ModelName>.findOneAndUpdate(
  {conditions},   what to find
  {updates},   what to update
  function(err, results){}
);
```

Mongoose findOneAndUpdate()

```
<ModelName>.findOneAndUpdate(
  {conditions},
  {$pull: {field: {query}}},
  function(err, results){}
);
```

how where — which one from here you want to pull

```
app.post("/delete", function(req, res){
  const checkedItemId = req.body.checkbox;
  const listName = req.body.listName;

  if (listName === "Today") {
    Item.findByIdAndRemove(checkedItemId, function(err){
      if (!err) {
        console.log("Successfully deleted checked item.");
        res.redirect("/");
      }
    });
  } else {
    List.findOneAndUpdate({name: listName}, {$pull: {items: {_id: checkedItemId}}}, function(err, foundList){
      if (!err){
        res.redirect("/" + listName);
      }
    })
  }

});
```

W033 - Missing semicolon.

```
const listSchema = {
  name: String,
  items: [itemsSchema]
};
```

* Problem: differentiates between lowerCase and Upper Case routes
  eg: localhost:3000/home and localhost:3000/Home

a we use lodash capitalize
1) const _ = require("lodash");

```
app.get("/:customListName", function(req, res){
  const customListName = _.capitalize(req.params.customListName);

  List.findOne({name: customListName}, function(err, foundList){
    if (!err){
      if (!foundList){
        //Create a new list
        const list = new List({
          name: customListName,
          items: defaultItems
        });
        list.save();
        res.redirect("/" + customListName);
      } else {
        //Show an existing list

        res.render("list", {listTitle: foundList.name, newListItems: foundList.items});
      }
    }
  });
```

calling lodash