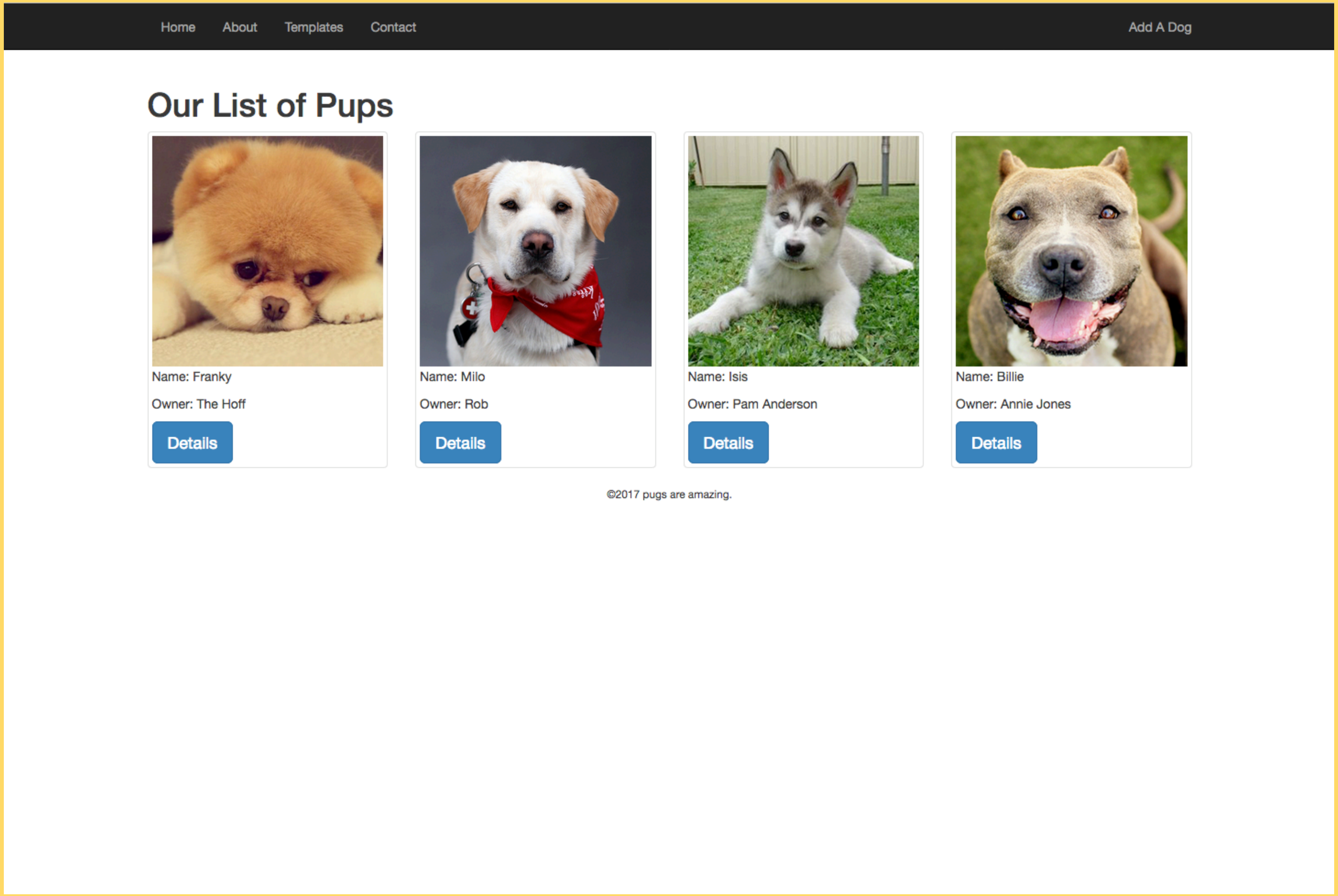
EDIT, DELETE & MONGO DB

Agenda

- Last Week
- Advanced Templates
- Creating a View Page
- Deleting Items
- Adding a Database
- Assignment #1
- In-Class

What did we cover last week?

What was the significance?



Our List of Pups

Name: Franky

Owner: The Hoff

Name: Milo

Owner: Rob

Name: Isis

Owner: Pam Anderson

Name: Billie

Owner: Annie Jones

©2017 pugs are amazing.

More advanced templates with PUG!

We loaded dynamic content and created an “ADD” button.

Today we will add an View page, add a delete button and add MongoDB

We will use JSON files for the first bit until we have MongoDB

Let’s Continue From Last Week

Open “app.js”. We are going to add a unique ID to each Dog when they are added.

```
var express = require('express');
var app = express();
var path = require('path');
var fs = require("fs");
var bodyParser = require('body-parser');
var multer = require('multer');
var upload = multer();
var randomID = require("random-id");
app.set('view engine', 'pug');
app.set('views', './views');
app.use(express.static(path.join(__dirname, 'public')));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: true}));
app.use(upload.array());

...
```

wk4_01/app.js

Let’s Continue From Last Week

Open “app.js”. Now each new dog will have a unique ID.

```
app.post('/', function (req, res) {
    console.log(req.body);
    console.log(req.body.name);
    req.body.img = "/images/doggo.gif";
    req.body.id = randomID(10);
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        data = JSON.parse(data);
        data.push(req.body);
        data = JSON.stringify(data);
        fs.writeFile(__dirname + "/data/users.json", data, function (err) {
            if (err) {
                console.log(err.message);
                return;
            }
            res.render('index', {
                users: data,
                success: "Your Dog Was Added to the List"
            });
            console.log("The file was saved!");
        });
    });
});
```

wk4_01/app.js

Launch the app.js

Open a browser and visit: http://localhost:3000. When you add a dog, the data will create an ID for the new dog. Cool!

```
$ node app.js
```

Make the Edit Button Work

Open “app.js”. We are going to add a unique ID to each Dog when they are added.

```
...

div(class="thumbnail")
    img(src= dogs[i].img)
    p Name: #{dogs[i].name}
    p Owner: #{dogs[i].owner}
    a(class="btn btn-primary btn-lg", href= dogs[i].id role="button") Details

...
```

Add a “.get” in our server

Open “app.js”. We are going to add an app.get. This will return the ID of the dog to the browser.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    res.end( JSON.stringify(req.params));
})
```

Look at our index.pug link and our app.get

The link we create will be “/dogs[i].id” where dogs[i].id is a value from the JSON file. When we create the app.get, we want to make sure the request is the same address.

wk4_01/views/index.pug

```
a(class="btn btn-primary btn-lg", href= dogs[i].id role="button") Details
```

wk4_01/app.js

```
app.get('/:id', function (req, res) {
    // First read existing users.
    res.end( JSON.stringify(req.params));
})
```

Launch the app.js

Open a browser and visit: http://localhost:3000. Click the detail button of a dog, it should return a JSON string to the window. Now our server knows which dog we want to learn more about.

```
$ node app.js
```

Let’s make our Details page show the dog.

Open “app.js”. Replace the old app.get with the one below.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

What Does Our New Code Do?

Opens the JSON file and loads it into a variable. DON'T FORGET TO PARSE THE JSON!

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

What Does Our New Code Do?

Grabs the id of the dog clicked.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

What Does Our New Code Do?

Loops through our array of dogs to find.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

What Does Our New Code Do?

Create a variable for the record.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

What Does Our New Code Do?

Render a new page using the “view.pug” template and pass the user to the page.

```
app.get('/:id', function (req, res) {
    // First read existing users.
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        var userId = req.params.id;
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching `id` value
            if (users[i].id == userId) {
                // we found it
                var user = JSON.stringify(users[i]);
            }
        }
        res.render('view', {
            users: user
        });
    });
});
```

wk4_02/app.js

Are we ready to launch our app?

Hint, the answer is no…but why?

Create a “view.pug” template.

Rename the “_view.pug” to “view.pug” in the “wk4_02/views” folder.

```
doctype html
html
    head
        title= title
        link(rel="stylesheet",href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css")
        link(rel='stylesheet', href='/css/stylesheet.css')

    body
        include ./header.pug
        div(class="container")
            -var dogs = JSON.parse(users)
            h1 All About #{dogs.name}
            div(class="row")
                div(id= dogs.id, class="col-xs-6 col-md-3")
                    div(class="thumbnail")
                        img(src= dogs.img)
                        p Name: #{dogs.name}
                        p Owner: #{dogs.owner}
        script(src='https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js')
        script(src='/js/myscript.js')
        include ./footer.pug
```

wk4_02/views/view.pug

Look at our view.pug link and our app.get

In our template we are creating a local variable called “dogs”. It uses the data sent from the server – users – as the data. This data is JSON and needs to be parsed.

wk4_02/views/view.pug

```
- var dogs = JSON.parse(users)
```

wk4_02/app.js

```
res.render('view', {
   users: user
});
```

Launch the app.js

Open a browser and visit: http://localhost:3000. Click the detail button of a dog, it should return a page that has the details of a dog.

```
$ node app.js
```

Cool…now I can ADD a dog, and VIEW it. Can we DELETE a dog?

Yes, as long as it's not Milo…never delete Milo.

Install Pug using npm

Inside the “wk4_03” folder, type the following:

```
$ npm install method-override
```

Add a delete button to the “view.pug” template

Rename the “_view.pug” to “view.pug” in the “wk4_02/views” folder.

```
form(method='POST', action='/' + dogs.id + '?_method=DELETE')
    button(class='btn.btn-danger', id='submit', type='submit') Delete
```

wk4_03/views/view.pug

Add a “.delete” in our server

Open “app.js”. We are going to add an app.delete. We need to pass the ID of the dog.

```
app.delete('/:id', function (req, res) {
    var userId = req.params.id;
    console.log(userId);

});
```

wk4_03/app.js

Look at our view.pug and our app.delete

The link we create will be “/dogs[i].id” where dogs[i].id is a value from the JSON file. When we create the app.get, we want to make sure the request is the same address.

wk4_03/views/index.pug

```
form(method='POST', action='/' + dogs.id + '?_method=DELETE')
```

wk4_03/app.js

```
app.delete('/:id', function (req, res) {
    var userId = req.params.id;
    console.log(userId);
});
```

Launch the app.js

Open a browser and visit: http://localhost:3000. Navigate to a dog’s details page. From there click delete. What happens? Check the console.

```
$ node app.js
```

Update the JSON In the “.delete” on our server

Open “app.js”. We are going to find the entry with the matching id.

```
app.delete('/:id', function (req, res) {
    var userId = req.params.id;
    console.log(userId);
    fs.readFile(__dirname + "/data/users.json", 'utf8', function (err, data) {
        users = JSON.parse(data);
        // iterate over each element in the array
        for (var i = 0; i < users.length; i++) {
            // look for the entry with a matching 'id' value
            if (users[i].id == userId) {
                // we found it
                users.splice(i, 1);
                //delete users[i];
                //console.log(users);
                //var user = JSON.stringify(users[i]);
            }
        }
        console.log(users);
        users = JSON.stringify(users);
    });
});
```

wk4_04/app.js

Launch the app.js

Open a browser and visit: http://localhost:3000. Navigate to a dog’s details page. From there click delete. What happens? Check the console.

```
$ node app.js
```

Save the change to the JSON file.

Open “app.js”. We are going to find the entry with the matching id.

```
...

        console.log(users);
        users = JSON.stringify(users);
        fs.writeFile(__dirname + "/data/users.json", users, function (err) {
            if (err) {
                console.log(err.message);
                return;
            }
            res.render('index', {
                users: users
            });

            console.log("The file was saved!");
        });

...
```

Launch the app.js

Open a browser and visit: http://localhost:3000. Navigate to a dog's details page. From there click delete. The Dog should be deleted from the JSON file.

```
$ node app.js
```

How can I show a message that the dog was deleted?

As long as it’s not Milo…never delete Milo.

Update the server to send a delete message.

Open “app.js”. Replace with the following code.

```
...

        console.log(users);
        users = JSON.stringify(users);
        fs.writeFile(__dirname + "/data/users.json", users, function (err) {
            if (err) {
                console.log(err.message);
                return;
            }
            res.render('index', {
                users: users,
                deleted: "Your Dog Was Deleted From the List"
            });
            console.log("The file was saved!");
        });

...
```

wk4_06/app.js

Add a conditional in the “header” template.

Open “header.pug”. Replace the following code.

```
...

if(success)
    div(class="container")
        div(class="container alert alert-success")
            strong Success!
            span= success
if(deleted)
    div(class="container")
        div(class="container alert alert-danger")
            strong Success!
            span= deleted

...
```

wk4_06/views/header.pug

Ok, now we are good...I think I can go on break.

Seriously, go on a break...the next slide is the break slide.

BREAK

See you in 10-15 minutes

How to Install MongoDB

https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-os-x/

http://treehouse.github.io/installation-guides/mac/mongo-mac.html

Download the binary files for the desired release of MongoDB.

Download the binaries from https://www.mongodb.org/downloads.
For example, to download the latest release through the shell, issue the following

```
curl -O https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.2.11.tgz
```

Extract the files from the downloaded archive

For example, from a system shell, you can extract through the tar command:

```
tar -zxvf mongodb-osx-x86_64-3.2.11.tgz
```

Copy the extracted archive to the target directory

Copy the extracted folder to the location from which MongoDB will run.

```
mkdir -p mongodb
cp -R -n mongodb-osx-x86_64-3.2.11/ mongodb
```

Ensure the location of the binaries is in the PATH variable

The MongoDB binaries are in the bin/ directory of the archive. To ensure that the binaries are in your PATH, you can modify your PATH.

```
export PATH=<mongodb-install-directory>/bin:$PATH
```

NOTE: Replace <mongodb-install-directory> with the path to the extracted MongoDB archive.

How to Run MongoDB

https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-os-x/

Create the data directory

Before you start MongoDB for the first time, create the directory to which the **mongod** process will write data.

The following example command creates the data directory in your home folder

```
mkdir -p ~/data
```

NOTE: You can save the data to a different location by changing the directory.

Run the Mongo daemon

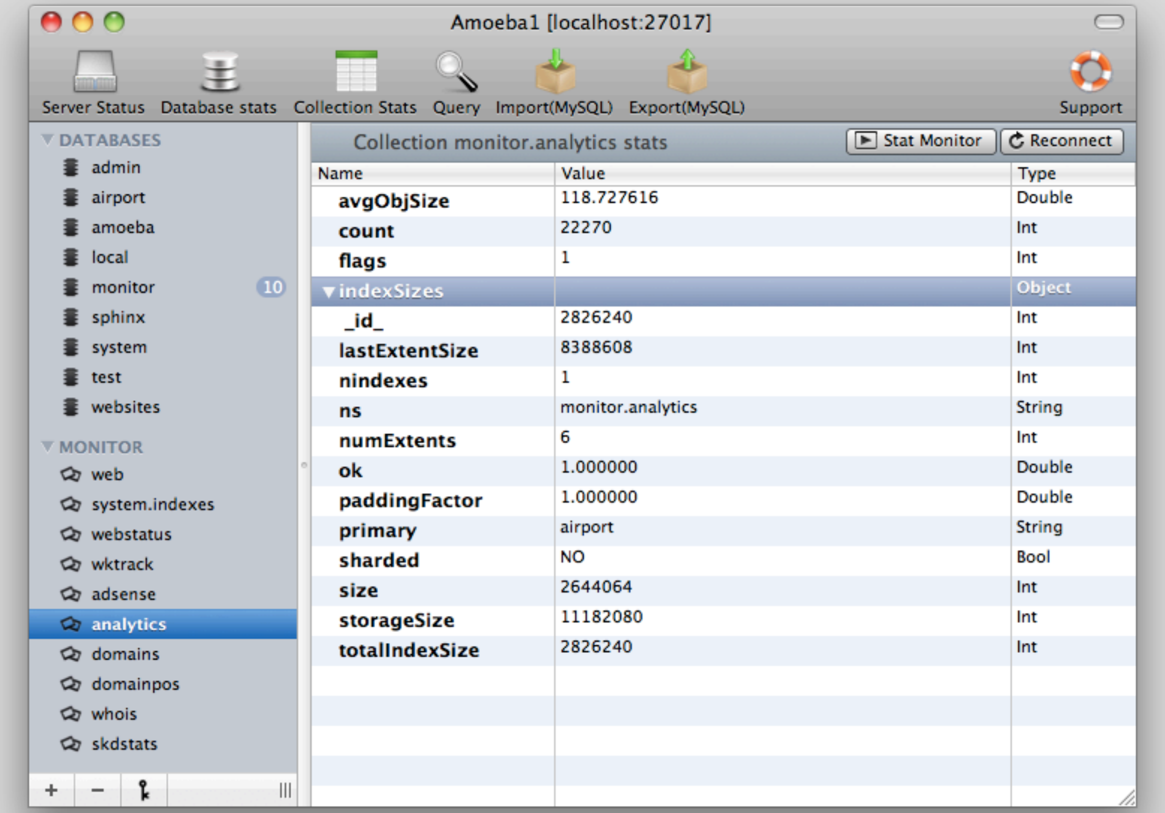
This will start the mongo server.

–dbpath=data instructs the **daemon** to use a custom data directory.
–port 27017 instructs the **daemon** to use a custom port

```
~/mongodb/bin/mongod --dbpath=data --port 27017
```

MongoHub

is a native mac GUI application for mongodb, designed for mongodb administration.



Name	Value	Type
avgObjSize	118.727616	Double
count	22270	Int
flags	1	Int
▼indexSizes		Object
_id_	2826240	Int
lastExtentSize	8388608	Int
nindexes	1	Int
ns	monitor.analytics	String
numExtents	6	Int
ok	1.000000	Double
paddingFactor	1.000000	Double
primary	airport	String
sharded	NO	Bool
size	2644064	Int
storageSize	11182080	Int
totalIndexSize	2826240	Int

https://github.com/jeromelebel/MongoHub-Mac

Run the Mongo shell (if you don’t like MongoHub)

This will start the mongo shell where you can interact directly with your MongoDB.

For a list of commands check out the following link:
https://docs.mongodb.com/v3.2/reference/mongo-shell/

```
~/mongodb/bin/mongo
```

Assignment #1

You are going to use your knowledge of Node.js, Express, Pug and MongoDB to create an interactive ToDo List. The ToDo list will enable a user to VIEW all ToDo list items, DELETE an item and ADD a new item to the list. All ToDo list items are to be stored in a MongoDB. You need to create your ToDo list app using Node.js with Express and Pug templates.

See BlackBoard for more details.

DUE: FEBRUARY 10, 2017 @ 5:00 pm (at the end of class)

In-Class Challenge

1. Start working on Assignment #1. Based on what we have covered so far, you should be able to create all of the functionality (minus MongoDB).
2. Come to class next Friday with your ToDo list app working (minus MongoDB). Save to a local JSON file.
3. We will add MongoDB to your site next week.
4. I have attached my notes for MongoDB from last semester. Please review before next week's class.

How Are We Doing?

Are your Brains fried yet?

Some Code
To Get You Going

These are the basics - CRUD.

The Following Pages are from my Last Semester's MongoDB lecture.

Please Review Before next week.

Install the MongoDB module

Use NPM to install the module.

```
> npm install mongodb
```

Connect to your MongoDB

Open the corresponding file and add the following code (shown in blue).

```
var mongodb = require('mongodb');
//We need to work with "MongoClient" interface in order to connect to a mongodb server.
var MongoClient = mongodb.MongoClient;
// Connection URL. This is where your mongodb server is running.
var url = 'mongodb://localhost:27017/presidents';
// Use connect method to connect to the Server
MongoClient.connect(url, function (err, db) {
        if (err) {
                console.log('Unable to connect to the mongoDB server. Error:', err);
        }     else {
                console.log('Connection established to', url);
                // do some work here with the database.
                 //Close connection
                db.close();
        }
});
```

wk4-07.js

Important connection notes

Make sure your ports match.

```
var mongodb = require('mongodb');
//We need to work with "MongoClient" interface in order to connect to a mongodb server.
var MongoClient = mongodb.MongoClient;
// Connection URL. This is where your mongodb server is running.
var url = 'mongodb://localhost:27017/presidents';
// Use connect method to connect to the Server
MongoClient.connect(url, function (err, db) {
        if (err) {
                console.log('Unable to connect to the mongoDB server. Error:', err);
        }    else {
                console.log('Connection established to', url);
                // do some work here with the database.
                 //Close connection
                db.close();
        }
});
```

wk4-07.js

Important connection notes

The name of your database appears as the last part of your url.

```
var mongodb = require('mongodb');
//We need to work with "MongoClient" interface in order to connect to a mongodb server.
var MongoClient = mongodb.MongoClient;
// Connection URL. This is where your mongodb server is running.
var url = 'mongodb://localhost:27017/presidents';
// Use connect method to connect to the Server
MongoClient.connect(url, function (err, db) {
        if (err) {
                console.log('Unable to connect to the mongoDB server. Error:', err);
        }    else {
                console.log('Connection established to', url);
                // do some work here with the database.
                 //Close connection
                db.close();
        }
});
```

wk4-07.js

A MongoDB Has Collections & Documents

A single database can have multiple collections, each containing different document structures.

Let’s Create a Collection

We’re going to use our voters from the mini-challenges

Add a collection to your MongoDB

The name of your database appears as the last part of your url.

```
var mongodb = require('mongodb');
//We need to work with "MongoClient" interface in order to connect to a mongodb server.
var MongoClient = mongodb.MongoClient;
// Connection URL. This is where your mongodb server is running.
var url = 'mongodb://localhost:27017/presidents';
// Use connect method to connect to the Server
MongoClient.connect(url, function (err, db) {
        if (err) {
                console.log('Unable to connect to the mongoDB server. Error:', err);
        }    else {
                console.log('Connection established to', url);
                // Data is stored in collections; we need to create one
                var collection = db.collection('voters');
                 //Close connection
                db.close();
        }
});
```

wk4-08.js

Let’s Add a Document to Our Collection

We’re going to use our voters from the mini-challenges

Add documents to your MongoDB – Part I

Insert the code into the Else of the successful connection to the DB

```
…

console.log('Connection established to', url);
// Data is stored in collections; we need to create one
var collection = db.collection('voters');
//Create some users
var user1 = {name: 'Kimmy Kardashian', occupation: 'Self Appointed Awesome
Person', politicalAffiliation:'Shopping'};     var user2 = {name: 'David
Hasselhoff', occupation: 'German Demi-God', politicalAffiliation:'Church of
Baywatch'};
var user3 = {name: 'Joey Chestnut', occupation: 'Professional Eater',
politicalAffiliation:'Oscar Mayer'};

…
```

wk4-09.js

Add documents to your MongoDB – Part II

Insert the code into the Else of the successful connection to the DB

```
…

// Insert some users
collection.insert([user1, user2, user3], function (err, result) {
       if (err) {
              console.log(err);
       } else {
              console.log('The documents inserted with "_id" are:',
result.length, result);
       }
       //Close connection
       db.close();
});

…
```

wk4-09.js

Some notes about the data

MongoDB stores data as BSON; Binary JSON.

```
…

var user2 = { name: 'David Hasselhoff',
              occupation: 'German Demi-God',
              politicalAffiliation: 'Church of Baywatch’
};

…
```

wk4-09.js

Let’s Edit a Document in Our Collection

We’re going to use our voters from the mini-challenges

Edit a document in your MongoDB

Insert the code into the Else of the successful connection to the DB

```
…
console.log('Connection established to', url);
// Data is stored in collections; we need to create one
var collection = db.collection('voters');
// Edit one of the users
collection.update({
        name: 'David Hasselhoff'}, {$set: {occupation: 'German/American Demi-God'}
}, function (err, numUpdated) {
        if (err) {
                console.log(err);
        }
        else if (numUpdated) {
                console.log('Updated Successfully.');
        }
        else {
                console.log('No document found with defined "find" criteria!');
        }
        //Close connection
        db.close();
});
…
```

wk4-10.js

More about collection.update()

First part identifies which record you are looking for

```
…

// Edit one of the users
collection.update({
      name: 'David Hasselhoff’
}, {  $set: {
            occupation: 'German/American Demi-God’
      }
}

…
```

wk4-10.js

More about collection.update()

$set identifies what you want to update

```
…

// Edit one of the users
collection.update({
      name: 'David Hasselhoff’
}, {  $set: {
            occupation: 'German/American Demi-God’
      }
}

…
```

wk4-10.js

Let’s Query a Collection

We’re going to use our voters from the mini-challenges

Query a collection in your MongoDB

Insert the code into the Else of the successful connection to the DB

```
…
console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.find({
        occupation: 'Professional Eater'
}).toArray(function (err, result) {
        if (err) {
                console.log(err);
        }
        else if (result.length) {
                console.log('Found:', result);
        }
        else {
                console.log('No document(s) found with defined "find" criteria!');
        }
        //Close connection
        db.close();
});
…
```

wk4-11.js

More notes about collection.find()

You can search for any key/value pair

```
…
console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.find({
        name: 'Joey Chestnut'
}).toArray(function (err, result) {
        if (err) {
                console.log(err);
        }
        else if (result.length) {
                console.log('Found:', result);
        }
        else {
                console.log('No document(s) found with defined "find" criteria!');
        }
        //Close connection
        db.close();
});
…
```

wk4-11.js

More notes about collection.find()

You can search for any key/value pair

```
…
console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.find({
        occupation: 'Professional Eater'
}).toArray(function (err, result) {
        if (err) {
                console.log(err);
        }
        else if (result.length) {
                console.log('Found:', result);
        }
        else {
                console.log('No document(s) found with defined "find" criteria!');
        }
        //Close connection
        db.close();
});
…
```

wk4-11.js

More notes about collection.find()

.toArray returns the entire record associated with the key/value pair

```
…
console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.find({
         occupation: 'Professional Eater'
}).toArray(function (err, result) {
         if (err) {
                  console.log(err);
         }
         else if (result.length) {
                  console.log('Found:', result);
         }
         else {
                  console.log('No document(s) found with defined "find" criteria!');
         }
         //Close connection
         db.close();
});
…
```

wk4-11.js

Let’s Delete a Document from our Collection

We’re going to use our voters from the mini-challenges

Delete a document from your MongoDB

This will delete the first document that contains the key/value pair

```
…

console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.deleteOne({
        occupation: "German Demi-God"
}, function (err, result) {
        if (err) {
                console.log(err);
        }
        console.log('Deleted the record.');
        //Close connection
        db.close();
});

…
```

wk4-12.js

Let’s Delete Multiple Documents from our Collection

We’re going to use our voters from the mini-challenges

Delete a document from your MongoDB

This will delete all of the documents that contains the **key/value** pair

```
...

console.log('Connection established to', url);
// Get the voters collection
var collection = db.collection('voters');
// Insert some users
collection.deleteMany({
        occupation: "German Demi-God"
}, function (err, result) {
        if (err) {
                console.log(err);
        }
        console.log('Deleted the record.');
        //Close connection
        db.close();
});

...
```

wk4-13.js

All of the CRUD Functionality You Need!

The Create, Read, Update and Delete functions of MongoDB