Meanstack

Angular tuts

——

Install NodeJS
Install MongoDB
Install Express Generator

```
npm install -g express-generator
```

Install Express

NodeJS Express Installation overview

USERLIST app with JADE - STEPS TO BE MADE

——

TO-DO App with ANGULAR

```
- public            <!-- holds all our files for our frontend angular applicatior
    ----- core.js       <!-- all angular code for our app -->
    ----- index.html    <!-- main view -->
    - package.json      <!-- npm configuration to install dependencies/modules --
    - server.js         <!-- Node configuration -->
```

cd to mean-examples
express todo

package.json

```
{
  "name"         : "node-todo",
  "version"      : "0.0.0",
  "description"  : "Simple todo application.",
  "main"         : "server.js",
  "author"       : "Scotch",
  "dependencies" : {
    "express"    : "~4.7.2",
    "mongoose"   : "~3.6.2",
    "morgan" : "~1.2.2",
"body-parser": "~1.5.2",
"method-override": "~2.1.2"
  }
}
```

npm install

server.js

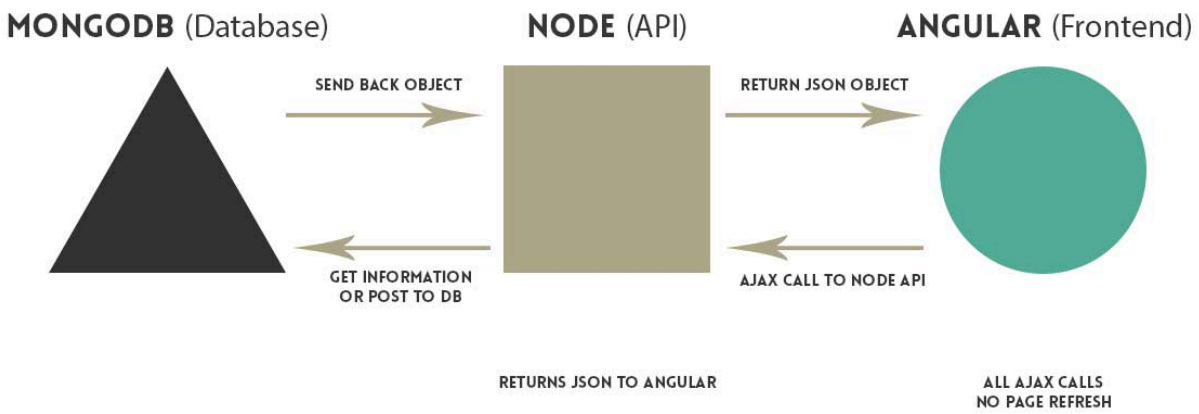
This is the file where we will:

- Configure our application
- Connect to our database
- Create our Mongoose models
- Define routes for our RESTful API
- Define routes for our frontend Angular application

- Set the app to listen on a port so we can view it in our browser

```
// set up ========================
var express = require('express');
var app = express(); // create our app w/ express
var mongoose = require('mongoose'); // mongoose for mongodb
var morgan = require('morgan'); // log requests to the console (express4)
var bodyParser = require('body-parser'); // pull information from HTML POST (express4)
var methodOverride = require('method-override'); // simulate DELETE and PUT (express4)

// configuration =================

mongoose.connect('mongodb://node:nodeuser@mongo.onmodulus.net:27017/uwO3mypu'); // connect to
mongoDB database on modulus.io

app.use(express.static(__dirname + '/public')); // set the static files location /public/img will be /img for users
app.use(morgan('dev')); // log every request to the console
app.use(bodyParser.urlencoded({
    'extended': 'true'
})); // parse application/x-www-form-urlencoded
app.use(bodyParser.json()); // parse application/json
app.use(bodyParser.json({
    type: 'application/vnd.api+json'
})); // parse application/vnd.api+json as json
app.use(methodOverride());

// listen (start app with node server.js) ======================================
app.listen(8080);
console.log("App listening on port 8080");
```

Automatically restart server when files change: By default, node will not monitor for file changes after your server has been started. This means you'd have to shut down and start the server every time you made a file change. This can be fixed withnodemon. To use: install nodemon globally `npm install -g nodemon`. Start your server with `nodemon server.js` now. Smooth sailing from there.




```
// define model =================
var Todo = mongoose.model('Todo', {
    text: String
});


// routes ======================================================================

// api ---------------------------------------------------------------------
```

```
// get all todos
app.get('/api/todos', function (req, res) {

    // use mongoose to get all todos in the database
    Todo.find(function (err, todos) {

        // if there is an error retrieving, send the error. nothing after res.send(err) will execute
        if (err)
            res.send(err)

        res.json(todos); // return all todos in JSON format
    });
});

// create todo and send back all todos after creation
app.post('/api/todos', function (req, res) {

    // create a todo, information comes from AJAX request from Angular
    Todo.create({
        text: req.body.text,
        done: false
    }, function (err, todo) {
        if (err)
            res.send(err);

        // get and return all the todos after you create another
        Todo.find(function (err, todos) {
            if (err)
                res.send(err)
            res.json(todos);
        });
    });

});

// delete a todo
app.delete('/api/todos/:todo_id', function (req, res) {
    Todo.remove({
        _id: req.params.todo_id
    }, function (err, todo) {
        if (err)
            res.send(err);

        // get and return all the todos after you create another
        Todo.find(function (err, todos) {
            if (err)
                res.send(err)
            res.json(todos);
        });
    });
});
```

Now for HTML and javascript

set up Angular

```
// public/core.js
var scotchTodo = angular.module('scotchTodo', []);

function mainController($scope, $http) {
    $scope.formData = {};

    // when landing on the page, get all todos and show them
    $http.get('/api/todos')
        .success(function(data) {
            $scope.todos = data;
```

```
            console.log(data);
        })
        .error(function(data) {
            console.log('Error: ' + data);
        });

    // when submitting the add form, send the text to the node API
    $scope.createTodo = function() {
        $http.post('/api/todos', $scope.formData)
            .success(function(data) {
                $scope.formData = {}; // clear the form so our user is ready to enter another
                $scope.todos = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
            });
    };

    // delete a todo after checking it
    $scope.deleteTodo = function(id) {
        $http.delete('/api/todos/' + id)
            .success(function(data) {
                $scope.todos = data;
                console.log(data);
            })
            .error(function(data) {
                console.log('Error: ' + data);
            });
    };

}
```

index.html

```
<!doctype html>

<!-- ASSIGN OUR ANGULAR MODULE -->
<html ng-app="scotchTodo">

<head>
    <!-- META -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Optimize mobile viewport -->

    <title>Node/Angular Todo App</title>

    <!-- SCROLLS -->
    <link rel="stylesheet" href="//netdna.bootstrapcdn.com/bootstrap/3.0.0/css/bootstrap.min.css">
    <!-- load bootstrap -->
    <style>
        html {
            overflow-y: scroll;
        }

        body {
            padding-top: 50px;
        }

        #todo-list {
            margin-bottom: 30px;
        }

        #todo-form {
            margin-bottom: 50px;
```

```
        }
    </style>

    <!-- SPELLS -->
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.16/angular.min.js"></script>
    <!-- load angular -->
    <script src="core.js"></script>

</head>
<!-- SET THE CONTROLLER AND GET ALL TODOS WITH INITIALIZE FUNCTION -->

<body ng-controller="mainController">
    <div class="container">

        <!-- HEADER AND TODO COUNT -->
        <div class="jumbotron text-center">
            <h1>I'm a Todo-aholic <span class="label label-info">{{ todos.length }}</span></h1>
        </div>

        <!-- TODO LIST -->
        <div id="todo-list" class="row">
            <div class="col-sm-4 col-sm-offset-4">

                <!-- LOOP OVER THE TODOS IN $scope.todos -->
                <div class="checkbox" ng-repeat="todo in todos">
                    <label>
                        <input type="checkbox" ng-click="deleteTodo(todo._id)"> {{ todo.text }}
                    </label>
                </div>

            </div>
        </div>

        <!-- FORM TO CREATE TODOS -->
        <div id="todo-form" class="row">
            <div class="col-sm-8 col-sm-offset-2 text-center">
                <form>
                    <div class="form-group">

                        <!-- BIND THIS VALUE TO formData.text IN ANGULAR -->
                        <input type="text" class="form-control input-lg text-center" placeholder="I want to buy a puppy
that will love me forever" ng-model="formData.text">
                    </div>

                    <!-- createToDo() WILL CREATE NEW TODOS -->
                    <button type="submit" class="btn btn-primary btn-lg" ng-click="createTodo()">Add</button>
                </form>
            </div>
        </div>

        <div class="text-center text-muted">
            <p>A demo by <a href="http://scotch.io">Scotch</a>.</p>
            <p>Read the <a href="http://scotch.io/tutorials/javascript/creating-a-single-page-todo-app-with-node-
and-angular">tutorial</a>.</p>
        </div>

    </div>

</body>

</html>
```

NOW OUR OWN MONGODB

kill mongod in Activity Monitor

mkdir data

/Applications/mongodb/mongod —dbpath /usr/local/bin/mean-examples/todo/data
/Applications/mongodb/mongo

server.js
mongoose.connect('mongodb://localhost/myapp');

——
Web Sockets - Chat App

cd examples
express chat4

package.json
```
{
  "name": "socket-chat-example",
  "version": "0.0.1",
  "description": "my first socket.io app",
  "dependencies": {
    "express": "^4.10.2"
  }
}
```

npm install

app.js
```
var app = require('express')();
var http = require('http').Server(app);

app.get('/', function (req, res) {
   res.send('<h1>Hello world</h1>');
});

http.listen(3000, function () {
   console.log('listening on *:3000');
});
```

browse to localhost:3000 to see <h1>Hello world</h1>

modify index.js
```
app.get('/', function (req, res) {
   res.sendFile(__dirname + '/index.html');
});
```

create index.html
```
<!doctype html>
<html>
  <head>
    <title>Socket.IO chat</title>
    <style>
      * { margin: 0; padding: 0; box-sizing: border-box; }
      body { font: 13px Helvetica, Arial; }
```

```
      form { background: #000; padding: 3px; position: fixed; bottom: 0; width: 100%; }
      form input { border: 0; padding: 10px; width: 90%; margin-right: .5%; }
      form button { width: 9%; background: rgb(130, 224, 255); border: none; padding: 10px; }
      #messages { list-style-type: none; margin: 0; padding: 0; }
      #messages li { padding: 5px 10px; }
      #messages li:nth-child(odd) { background: #eee; }
    </style>
  </head>
  <body>
    <ul id="messages"></ul>
    <form action="">
      <input id="m" autocomplete="off" /><button>Send</button>
    </form>
  </body>
</html>
```

ctrl+c node app to browse changes

ctrl+c AGAIN

```
npm install --save socket.io
```

That will install the module and add the dependency

app.js

```
var io = require('socket.io')(http);

io.on('connection', function(socket){
  console.log('a user connected');
});
```

index.html

```
<script src="https://cdn.socket.io/socket.io-1.2.0.js"></script>
    <script src="http://code.jquery.com/jquery-1.11.1.js"></script>
    <script>
      var socket = io();
      $('form').submit(function(){
        socket.emit('chat message', $('#m').val());
        $('#m').val('');
        return false;
      });
      socket.on('chat message', function(msg){
        $('#messages').append($('<li>').text(msg));
      });
    </script>
```

app.js

```
var app = require('express')();
var http = require('http').Server(app);
var io = require('socket.io')(http);

app.get('/', function (req, res) {
    res.sendFile(__dirname + '/index.html');
});
```

```
io.on('connection', function (socket) {
    socket.on('chat message', function (msg) {
        io.emit('chat message', msg);
    });
});

http.listen(3000, function () {
    console.log('listening on *:3000');
});
```

——

Advanced javascript protoype??

——NEARLY WORKING

```
npm install express jade socket.io
```

server.js

```
var express = require('express'),
    app = express.createServer();


app.set('views', __dirname + '/views');
app.set('view engine', 'jade');
app.set("view options", {
    layout: false
});
app.configure(function () {
    app.use(express.static(__dirname + '/public'));
});
```

create two folders inside our project folder named `"public"` and`"views"`.

configure Express to serve a `"home.jade"` file

```
app.get('/', function (req, res) {
    res.render('home.jade');
});
app.listen(3000);
```

JADE time

at top of **server.js**

```
var jade = require('jade');
```

SOCKET TIME

at top of server.js

```
var io = require('socket.io').listen(app);
```

public/script.js

```
var socket = io.connect();

function addMessage(msg, pseudo) {
    $("#chatEntries").append('<div class="message"><p>' + pseudo + ' : ' + msg + '</p></div>');
}

function sentMessage() {
    if ($('#messageInput').val() != "") {
        socket.emit('message', $('#messageInput').val());
        addMessage($('#messageInput').val(), "Me", new Date().toISOString(), true);
        $('#messageInput').val('');
    }
}

function setPseudo() {
    if ($("#pseudoInput").val() != "") {
        socket.emit('setPseudo', $("#pseudoInput").val());
        $('#chatControls').show();
        $('#pseudoInput').hide();
        $('#pseudoSet').hide();
    }
}
```

```
socket.on('message', function (data) {
    addMessage(data['message'], data['pseudo']);
});

$(function () {
    $("#chatControls").hide();
    $("#pseudoSet").click(function () {
        setPseudo()
    });
    $("#submit").click(function () {
        sentMessage();
    });
});
```