**Exercise02\_05\_01 – Step 1**

index.ejs

<link rel="stylesheet" href="style.css">

<script type="text/javascript" src="main.js"></script>

------------------------------------

main.js

(function() {

var selectedUserId;

var cache = {};

});

------------------------------------

main.js

(function() {

var selectedUserId;

var cache = {};

alert("Hello from my IIFE!");

})();

------------------------------------

main.js

(function() {

var selectedUserId;

var cache = {};

function startup() {

alert('I am in startup()!')

}

document.addEventListener("DOMContentLoaded", startup, false);

})();

------------------------------------

main.js

function startup() {

var friends = document.getElementsByClassName('friend');

for (var i = 0; i < friends.length; i++) {

console.log('friend nbr: ', i);

}

}

main.js

for (var i = 0; i < friends.length; i++) {

friends[i].addEventListener("click", function() {

selectedUserId = this.getAttribute('uid');

console.log("Twitter id: ", selectedUserId);

});

}

------------------------------------

main.js

friends[i].addEventListener("click", function() {

for (var j = 0; j < friends.length; j++) {

friends[j].className = 'friend';

}

this.className += ' active';

selectedUserId = this.getAttribute('uid');

});

**Exercise02\_05\_01 – Step 2**

index.js

function ensureLoggedIn(req, res, next) {

var credentials = authenticator.getCredentials();

if (!credentials.access\_token ||   
 !credentials.access\_token\_secret ||  
 !credentials.twitter\_id) {

return res.sendStatus(401);

}

next();

}

------------------------------------

index.js

app.get('/checkcookie', ensureLoggedIn, function(req, res,   
 next){  
 console.log(req.cookies);  
 return res.sendStatus(200);

});

------------------------------------

index.js

app.get('/friends/:uid/notes', ensureLoggedIn, function(req,   
 res, next){

var credentials = authenticator.getCredentials();

storage.getNotes(credentials.twitter\_id, req.params.uid,  
 function(err, notes) {

if (err) {

return res.status(500).send(err);

}

res.send(notes);

});

});

storage.js

},

getNotes: function(ownerid, friendid, callback) {

var cursor = database.collection('notes').find({

owner\_id: ownerid,

friend\_id: friendid

});

cursor.toArray(function(err, notes) {

if (err) {

return callback(err);

}

callback(null, notes.map(function(note) {

return {

\_id: note.\_id,

content: note.content

}

}));

});

}

**Exercise02\_05\_01 – Step 3**

main.js

selectedUserId = this.getAttribute('uid');

var notes = getNotes(selectedUserId,   
 function(notes) {

console.log(notes);

});

------------------------------------

main.js

function getNotes(userId, callback) {

if (cache[userId]) {

return callback(cache[userId]);

}

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) {

var notes = JSON.parse(xhttp.responseText ||   
 []);

cache[userId] = notes;

callback(notes);

}

};

xhttp.open('GET', "/friends/" +   
 encodeURIComponent(userId) + "/notes", true);

xhttp.send();

}

**Exercise02\_05\_01 – Step 4**

main.js

var notes = getNotes(selectedUserId,   
 function(notes) {

var docFragment =   
 document.createDocumentFragment();

var notesElements =   
 createNoteElements(notes);

console.log(notes);

});

------------------------------------

main.js

function createNoteElements(notes) {

return notes.map(function(note) {

var element = document.createElement('li');

element.className = "note";

element.setAttribute('contenteditable', true);

element.textContent = note.content;

return element;

});

return notes;

}

------------------------------------

main.js

var notesElements =   
 createNoteElements(notes);

notesElements.forEach(function(element) {

docFragment.appendChild(element);

});

var newNoteButton =   
 createAddNoteButton();

docFragment.appendChild(newNoteButton);

console.log(notes);

------------------------------------

main.js

function createAddNoteButton() {

var element = document.createElement('li');

element.className = "add-note";

element.textContent = "Add a new note ...";

return element;

}

main.js

docFragment.appendChild(newNoteButton);

document.getElementById('notes').

innerHTML = "";

document.getElementById('notes').

appendChild(docFragment);

------------------------------------

main.js

element.textContent = "Add a new note ...";

element.addEventListener('click', function() {

var noteElement = createNoteElements([{}])[0];

document.getElementById('notes').

insertBefore(noteElement, this);

noteElement.focus();

});

------------------------------------

style.css

h1 {

border-bottom: 1px solid black;

}

a {

text-decoration: none;

}

.logout {

position: absolute;

}

**Exercise02\_05\_01 – Step 5**

index.js

var express = require('express');

var bodyParser = require('body-parser');

------------------------------------

index.js

app.use(require('cookie-parser')());

app.use(bodyParser.json());

------------------------------------

index.js

app.post('/friends/:uid/notes', ensureLoggedIn, function(req, res, next){

storage.insertNote(req.cookies.twitter\_id, req.params.uid,   
 req.body.content,

function(err, note) {

if (err) {

return res.status(500).send(err);

}

res.send(note);

});

});

------------------------------------

storage.js

},

insertNote: function(ownerid, friendid, content, callback) {

database.collection('notes').insert({

owner\_id: ownerid,

friend\_id: friendid,

content: content

},

function(err, result) {

if (err) {

return callback(err, result);

}

callback(null, {

\_id: result.ops[0].\_id,

content: result.ops[0].content

});

});

}

**Exercise02\_05\_01 – Step 6**

main.js

function postNewNote(userId, note, callback) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) {

var serverNote = JSON.parse(xhttp.responseText   
 || {});

cache[userId].push(serverNote);

callback(serverNote);

}

};

xhttp.open('POST', "/friends/" +   
 encodeURIComponent(userId) + "/notes", true);

xhttp.setRequestHeader("Content-Type",   
 "application/json;charset=UTF-8");

xhttp.send(JSON.stringify(note));

}

------------------------------------

main.js

element.textContent = note.content;

element.addEventListener('blur', function() {

note.content = this.textContent;

console.log('blur event');

});

return element;

------------------------------------

main.js

element.addEventListener('keydown', function(e) {

if (e.keyCode == 13) {

e.preventDefault();

if (element.nextSibling.className == 'add-  
 note') {

element.nextSibling.click();

}

else {

element.nextSibling.focus();

}

}

});

main.js

note.content = this.textContent;

if (note.content == "") {

if (note.\_id) {

}

else {

document.getElementById('notes').

removeChild(element);

}

}

------------------------------------

main.js

else if (!note.\_id) {

postNewNote(selectedUserId,

{content: this.textContent},

function(newNote) {

note.\_id = newNote.\_id;

});

}

**Exercise02\_05\_01 – Step 7**

index.js

app.put('/friends/:uid/notes/:noteid', ensureLoggedIn, function(req, res){

var noteId = req.params.noteid;

storage.updateNote(req.params.noteid,   
 req.cookies.twitter\_id, req.body.content,

function(err, note) {

if (err) {

return res.status(500).send(err);

}

res.send({

\_id: note.\_id,

content: note.content

});

});

});

------------------------------------

storage.js

var MongoClient = require('mongodb').MongoClient;

var ObjectID = require('mongodb').ObjectID;

------------------------------------

storage.js

},

updateNote: function(noteId, ownerId, content, callback) {

database.collection('notes').updateOne({

\_id: new ObjectID(noteId),

owner\_id: ownerId

},

{

$set: { content: content }

},

function(err, result) {

if (err) {

return callback(err);

}

database.collection('notes').findOne({

\_id: new ObjectID(noteId)

}, callback);

});

}

**Exercise02\_05\_01 – Step 8**

main.js

function putNote(userId, note, callback) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) {

var serverNote = JSON.parse(xhttp.responseText   
 || {});

callback(serverNote);

}

};

xhttp.open('PUT', "/friends/" +   
 encodeURIComponent(userId) + "/notes/" +   
 encodeURIComponent(note.\_id), true);

xhttp.setRequestHeader("Content-Type",   
 "application/json;charset=UTF-8");

xhttp.send(JSON.stringify(note));

}

------------------------------------

main.js

else if (!note.\_id) {

postNewNote(selectedUserId,

{content: this.textContent},

function(newNote) {

note.\_id = newNote.\_id;

});

}

else {

putNote(selectedUserId, note, function() {} );

}

**Exercise02\_05\_01 – Step 9**

index.js

app.delete('/friends/:uid/notes/:noteid', ensureLoggedIn, function(req, res){

var noteId = req.params.noteid;

storage.deleteNote(req.params.noteid,   
 req.cookies.twitter\_id,

function(err, note) {

if (err) {

return res.status(500).send(err);

}

res.sendStatus(200);

});

});

------------------------------------

storage.js

},

deleteNote: function(noteId, ownerId, callback) {

database.collection('notes').deleteOne({

\_id: new ObjectID(noteId),

owner\_id: ownerId

}, callback);

}

**Exercise02\_05\_01 – Step 10**

main.js

function deleteNote(userId, note, callback) {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (xhttp.readyState == 4 && xhttp.status == 200) {

cache[userId] =   
 cache[userId].filter(function(localNote) {

return localNote.\_id != note.\_id;

});

callback();

}

};

xhttp.open('DELETE', "/friends/" +   
 encodeURIComponent(userId) + "/notes/" +   
 encodeURIComponent(note.\_id), true);

xhttp.send();

}

------------------------------------

main.js

if (note.content == "") {

if (note.\_id) {

deleteNote(selectedUserId, note,   
 function() {

document.getElementById('notes').

removeChild(element);

});

}

**Exercise02\_05\_01 – Step 11**

authenticator.js

twitterCredentials.twitter\_id = data.id\_str;

res.cookie('access\_token', oauth\_access\_token,   
 { httponly: true });

res.cookie('access\_token\_secret',   
 oauth\_access\_token\_secret,   
 { httponly: true });

res.cookie('twitter\_id', data.id\_str,   
 { httponly: true });

------------------------------------

index.js

// var credentials = authenticator.getCredentials();

// if (!credentials.access\_token ||   
 !credentials.access\_token\_secret) {

// return res.sendStatus(401);

// }

if (!req.cookies.access\_token ||   
 !req.cookies.access\_token\_secret) {

return res.sendStatus(401);

}

var url =   
 'https://api.twitter.com/1.1/statuses/update.json';

authenticator.post(url, req.cookies.access\_token,   
 req.cookies.access\_token\_secret,

{

status: "Testing cookies on Twitter"

},

index.js

// var credentials = authenticator.getCredentials();

// if (!credentials.access\_token ||   
 !credentials.access\_token\_secret) {

// return res.sendStatus(401);

// }

if (!req.cookies.access\_token ||   
 !req.cookies.access\_token\_secret) {

return res.sendStatus(401);

}

var url = 'https://api.twitter.com/1.1/search/tweets.json';

var query = querystring.stringify({ q: 'BMW' });

url += "?" + query;

authenticator.get(url, req.cookies.access\_token,   
 req.cookies.access\_token\_secret,

------------------------------------

index.js

// var credentials = authenticator.getCredentials();

// if (!credentials.access\_token ||   
 !credentials.access\_token\_secret) {

// return res.sendStatus(401);

// }

if (!req.cookies.access\_token ||   
 !req.cookies.access\_token\_secret) {

return res.sendStatus(401);

}

var url = 'https://api.twitter.com/1.1/friends/list.json';

if (req.query.cursor) {

url += "?" + querystring.stringify({ cursor:   
 req.query.cursor });

}

authenticator.get(url, req.cookies.access\_token,   
 req.cookies.access\_token\_secret,

index.js

// var credentials = authenticator.getCredentials();

// if (!credentials.access\_token ||   
 !credentials.access\_token\_secret) {

// return res.redirect(‘/login’);

// }

if (!req.cookies.access\_token ||   
 !req.cookies.access\_token\_secret) {

return res.redirect(‘/login’);

}

. . . . .

storage.getFriends(req.cookies.twitter\_id,   
 function(err, friends) {  
------------------------------------

index.js

function renderMainPageFromTwitter(req, res) {

var credentials = authenticator.getCredentials();

------------------------------------

index.js

var url =   
 'https://api.twitter.com/1.1/friends/ids.json';

url += "?" + querystring.stringify({ user\_id:   
 req.cookies.twitter\_id, cursor: cursor});

authenticator.get(url, req.cookies.access\_token,   
 req.cookies.access\_token\_secret,

------------------------------------

index.js

var url =

'https://api.twitter.com/1.1/users/lookup.json';

url += "?" + querystring.stringify({ user\_id:   
 getHundredIds(n).join(',')});  
 authenticator.get(url, req.cookies.access\_token,   
 req.cookies.access\_token\_secret,

------------------------------------

index.js

app.get('/logout', function(req, res) {

authenticator.clearCredentials();

res.clearCookie('access\_token');

res.clearCookie('access\_token\_secret');

res.clearCookie('twitter\_id');

index.js

// var credentials = authenticator.getCredentials();

if (!req.cookies.access\_token ||   
 !req.cookies.access\_token\_secret ||   
 !req.cookies.twitter\_id) {

storage.getNotes(req.cookies.twitter\_id, req.params.uid,   
 function(err, notes) {

storage.insertNote(req.cookies.twitter\_id, req.params.uid,   
 req.body.content,  
 storage.updateNote(req.params.noteid,   
 req.cookies.twitter\_id, req.body.content,  
 storage.deleteNote(req.params.noteid,   
 req.cookies.