CSCI 4131 – Internet Programming
Homework Assignment 5

Posted 3/5/2018

Due: 3/23/2018 at 11:55 PM

Submissions Accepted (with Penalty) through 3/24/2018 at 11:55AM

1 Description

The objective of this assignment is to provide an introduction to web-development with Node.js. We will provide most of the code for the assignment to you, and you are required to complete certain functions to complete the assignment. Node.js is basically JavaScript running a Web-server. It uses an event-driven, non-blocking I/O model. So far, in this course we have used JavaScript for client-side scripting. For this assignment, we will use JavaScript for server-side scripting. Essentially, instead of writing the server code in Python, we will develop a basic web-server using JavaScript.

In this assignment, you will also use minimal amount of jQuery for AJAX and DOM manipulation. AJAX is used on client-side to create asynchronous web applications. It is an efficient means of requesting data from the server, receiving data from the server, and updating the web page without reloading the entire web-page. This assignment has 9 pages.

2 Preparation and Provided Files

I. The first step will be to get Node.js running on CSE lab machines.This can be accomplished as follows:

1. Log into a CSE lab machine.
2. Most of the CSE lab machines run version 8.9.4 of Node.js (as of 03/02/2018) which is the most current version. Vole runs version 6.12.3 which will also be sufficient for this assignment.
3. Open the terminal and type the following command to add the Node.js module:
   ```
   module add soft/nodejs
   ```
4. The next step is to check the availability of Node.js. Type the following command to check the version of Node.js on the machine:
   ```
   node -v
   ```
5. This will display the current installed version.

II. The second step is to create a Node.js project for this assignment. This can be accomplished as follows:

1. Open the terminal on a CSE lab machine.
2. Create a directory named <x500id_hw05> by typing the following command:
```
mkdir yourx500id_hw05
```
3. Go inside the directory by typing the following command:
```
cd  yourx500id_hw05
```
4. Having a file named *package.json* in Node.js project makes it easy to manage module dependencies and makes the build process easier. To create *package.json* file, type the following command:
```
npm init
```
5. This will prompt you to enter the information. Use the following guideline to enter the information (The things that you need to enter are in bold. Some fields can be left blank.):
```
name: (yourx500id_hw08) yourx500id_hw05

version: (1.0.0) <Leave blank>

description: Assignment 5

entry point: (index.js) <Leave blank> (We will provide an index.js
file for your use)

test command: <Leave blank>

git repository: <Leave blank>

keywords: <Leave blank>

author: yourx500id

license: (ISC) <Leave blank>
```
6. After filling in the above information, you will be prompted to answer the question: "`Is this ok? (yes)`". Type **yes** and hit enter.
7. Now copy all the files present that are provided for this assignment to this directory: *yourx500id_hw05*
8. Change the permissions of all files and folders inside *yourx500id_hw05 directory to 777.*

9. Listing all the available files in the directory should display the following:
```
drwx------  4 nayan003 CSEL-student    7 Mar  3 03:50 ./

drwx------ 12 nayan003 CSEL-student   17 Mar  3 03:50 ../

drwxrwxrwx  4 nayan003 CSEL-student    7 Mar  3 03:50 client/

-rwxrwxrwx  1 nayan003 CSEL-student 2957 Mar  3  2018 index.js*

drwxrwxrwx 50 nayan003 CSEL-student   50 Mar  3 03:37 node_modules/

-rwxrwxrwx  1 nayan003 CSEL-student  309 Mar  3 03:37 package.json*
```

```
-rwxrwxrwx  1 nayan003 CSEL-student 1297 Mar  3  2018 places.json*
```

10. The project setup is now complete and we are ready to start the server.

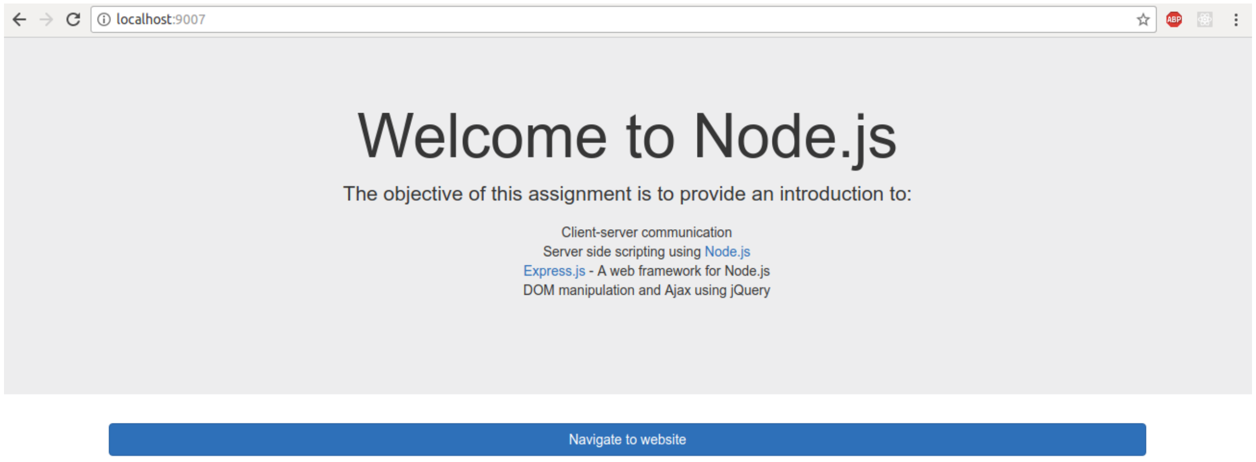
To start the server, type the following command:

```
node index.js
```

This starts the server and binds it to port 9007.

Now, using the local browser open: **localhost:9007**

The following page should be displayed:



Welcome to Node.js

The objective of this assignment is to provide an introduction to:

Client-server communication
Server side scripting using Node.js
Express.js - A web framework for Node.js
DOM manipulation and Ajax using jQuery

The following files are provided for this assignment:

1. **index.js**: This file contains the partially complete code for the node.js server.
2. **client/welcome.html**: Home page for this application.
3. **client/favorites.html**: Page which displays the list of favorite places.
4. **client/addPlace.html**: Form to add details about a new place.
5. **client/404.html**: Page used for 404 NOT FOUND error.
6. **client/405.html**: Page used for 405 METHOD NOT ALLOWED error.
7. **client/css/style.css**: CSS file used for styling.
8. **client/js/script.js**: JavaScript file used for AJAX call and for dynamic table population.
9. **places.json**: This file contains the list of favorite places in JSON format.

You will need to add code to the files highlighted in red only.

3 Functionality

Note: It is advisable to complete the code changes for server before changing the code for client. All the server endpoints (APIs) can be tested using POSTMAN or curl.

Client

All the resources related to client have been provided in the client folder. The client folder has five HTML files (welcome.html, favorites.html, addPlace.html, 404,html, 405.html). **You do not need to change any of these files.**

You only need to change client/js/script.js file.

favorites.html has a table (id = "myFavTable") whose body is empty. You need to add code in *client/js/script.js* file to dynamically populate the contents of the table after getting the list of favorite places from the server. Add code in client/js/script.js file to:

1. Hit getListOfFavPlaces endpoint of server using get method of AJAX.
2. Upon successful completion of API call, server will return the list of places.
3. Use the response returned to dynamically add rows to 'myFavTable' present in favorites.html page.
4. You can make use of jQuery or JavaScript to achieve this.

Server

When the server starts, it listens for incoming connections on port 9007. This server is designed to handle only GET and POST requests.

GET requests:

1. The server has been designed to serve three different HTML pages to clients: *welcome.html, favourites.html,* and *addPlace.html*
2. The server can also return the list of favorite places (in JSON format) by reading *places.json* file.
3. GET request for the welcome.html: The code for this has already been provided to you in *index.js* file. **You do not need to add any code for this.**
4. GET request for the *addPlace.html* page:
    a. You need to complete the *getAddPlacePage* function in index.js file to return addPlace.html page to the client.
5. GET request for the *favorites.html* page:
    a. You need to complete the *getFavouritesPage* function in index.js file to return your favorites.html page to the client.
6. GET request to return the list of favorite places:

    a. The function *getListOfFavPlaces* reads the *places.json* file and returns the list of favorite places in JSON format. **You do not need to add any code for this.**

7. GET request for any other resource: If the client requests any resource other than those listed above, the server should return 404.html file along with 404 code. For this, complete the *get404* function in index.js file.

POST requests:

8. The server should process the form data posted by client. The form present in *addPlace.html* allows user to enter details about a new place and update the list of favorite places.
9. Details for few places are pre-populated in *places.json* file. Your job is to add the details of the new place to this file and redirect the user to the *favorites.html* page after successful addition of the new course.
10. For this, you need to complete the *addPlaceFunction* function present in index.js file. This function should read the form data, add the new information to places.json file, and redirect the user to *favorites.html*. The code for redirection is 302. Ensure that the newly added data does not change the format of the course.json file. *Hint: You can use querystring module for parsing form data.*

Requests other than GET, POST:

11. If the server receives any requests other than GET or POST, the server should return 405.html file along with 405 code. For this, complete the *get405* function in index.js file.

4 Submission Instructions

1. Include the following two files in one zipped folder for your submission:
    a. index.js
    b. script.js
2. You do not need to provide any extra files.
3. The name of the zipped folder should be yourx500id_nodejs.
   PLEASE ENSURE TO TEST YOUR CODE ON CSE LAB MACHINES.

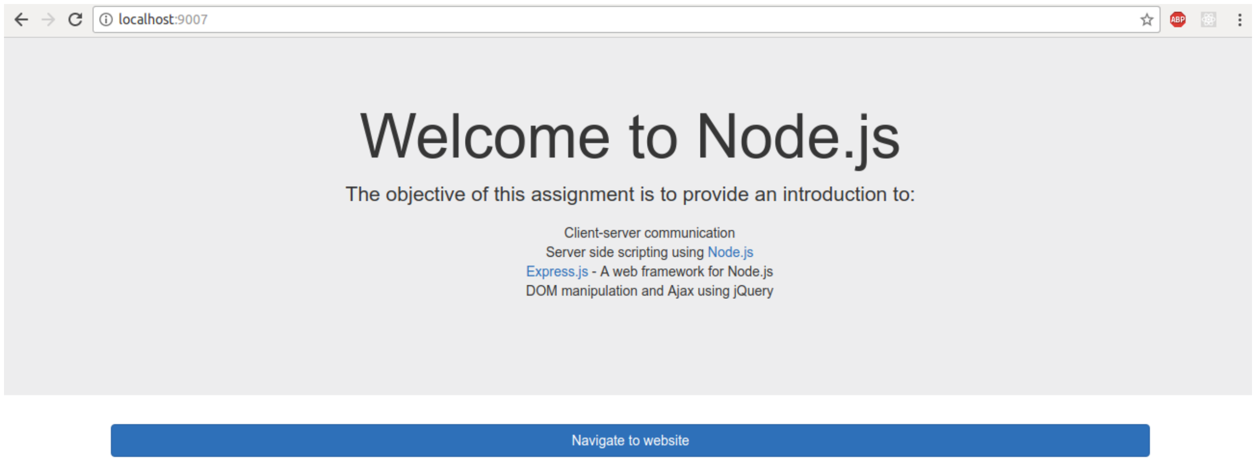
5 Evaluation

Your submission will be graded out of 100 points on the following items:

1. *POST endpoint* successfully adds the details of the new place to places.json file (**20 points**).
2. User is redirected to *favorites.html* page after successful addition of a new place (**10 points**).
3. Client successfully gets the list of favorite places by using getListOfFavPlaces API of server. The favorite places are dynamically added to the table present in favorites.html page. (**20 points**)
4. *addPlace.html* can be successfully returned by the server (**15 points**).
5. *favorites.html* can be successfully returned by the server (**15 points**).
6. Appropriate error message and code is returned for 404 error condition (**10 points**).
7. Appropriate error message and code is returned for 405 error condition (**10 points**).

6 Screenshots

welcome.html



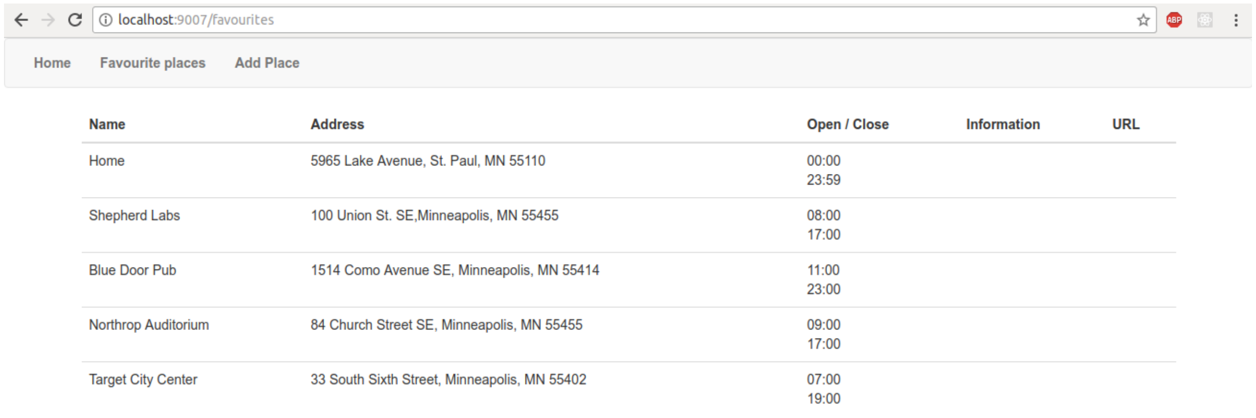
localhost:9007

Welcome to Node.js

The objective of this assignment is to provide an introduction to:

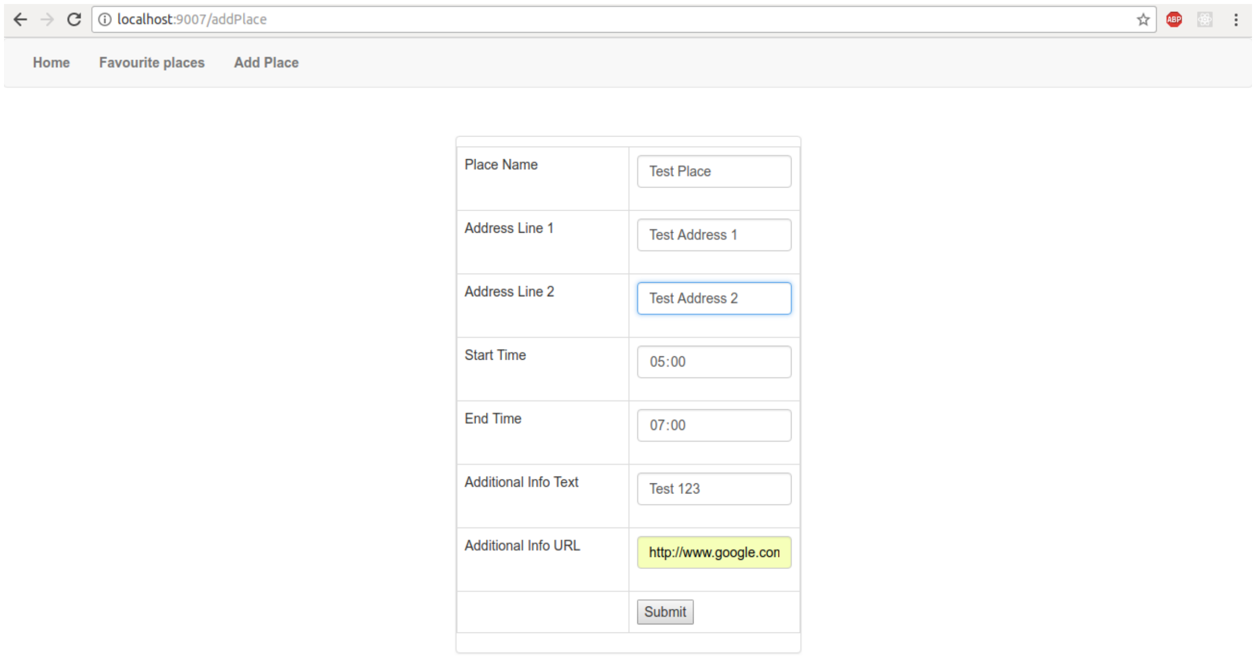
Client-server communication
Server side scripting using Node.js
Express.js - A web framework for Node.js
DOM manipulation and Ajax using jQuery

Initial display for favorites.html



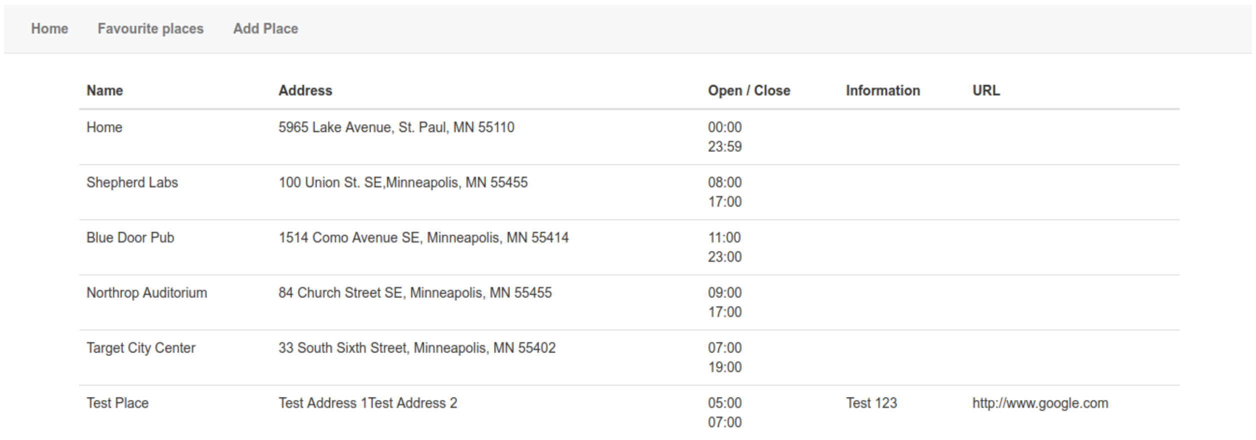
Name	Address	Open / Close	Information	URL
Home	5965 Lake Avenue, St. Paul, MN 55110	00:00 23:59		
Shepherd Labs	100 Union St. SE,Minneapolis, MN 55455	08:00 17:00		
Blue Door Pub	1514 Como Avenue SE, Minneapolis, MN 55414	11:00 23:00		
Northrop Auditorium	84 Church Street SE, Minneapolis, MN 55455	09:00 17:00		
Target City Center	33 South Sixth Street, Minneapolis, MN 55402	07:00 19:00		

Add details for a new place



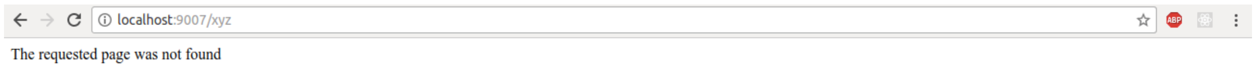
Place Name	Test Place
Address Line 1	Test Address 1
Address Line 2	Test Address 2
Start Time	05:00
End Time	07:00
Additional Info Text	Test 123
Additional Info URL	http://www.google.con

favorites.html page after addition of a new place



Name	Address	Open / Close	Information	URL
Home	5965 Lake Avenue, St. Paul, MN 55110	00:00 23:59		
Shepherd Labs	100 Union St. SE,Minneapolis, MN 55455	08:00 17:00		
Blue Door Pub	1514 Como Avenue SE, Minneapolis, MN 55414	11:00 23:00		
Northrop Auditorium	84 Church Street SE, Minneapolis, MN 55455	09:00 17:00		
Target City Center	33 South Sixth Street, Minneapolis, MN 55402	07:00 19:00		
Test Place	Test Address 1Test Address 2	05:00 07:00	Test 123	http://www.google.com

404 error



The requested page was not found