Node.js - Day 2

Express Framework Lab Assignments

1. Problem Statement

Make a hello world Express program that will display "Hello, world!" at the root URL: /

2. Problem Statement

Tell the year you were born

Make an express program that will display the year you were born when you report your age in a query parameter. When you type in http://localhost:3000/year?age=32 into the address bar of your browser, for example, it will display You were born in 1984.

3. Problem Statement

Create an Express.js app that outputs "Hello World!" when somebody goes to /home. The port number will be provided to you by expressworks as the first argument of the application, ie. process.argv[2].

4. Problem Statement

Make a Node.js program named express-hello-world.js that outputs "Hello World" to browsers making a GET request to the root (/) url.
Also, to browsers that make a GET request to the /time url, send the current date and time in ISO format: 2015-12-31T23:59:59.999Z.
Finally, use an environment variable named PORT for the port number if one is provided. If one is not provided use 8080.
i.e. The command below should start a server on the port 1337.

```
PORT=1337 node express-hello-world.js
```

and the command below should start a server on the port 8080.

```
node express-hello-world.js
```

5.Problem Statement :Forms

This exercise will teach you how to process the traditional (non-AJAX) web form.
Write a route ('/form') that processes HTML form input (<form><input name="str"/></form>) and responds with the value of str backwards.

To handle a POST request, use the post() method which is used the same way as get():

app.post('/path', function(req, res){...})
Express.js uses middleware to provide extra functionality to your web server.

Simply put, a middleware is a function invoked by Express.js before your own request handler.

Middleware provide a large variety of functionality such as logging, serving static files, and error handling.

A middleware is added by calling use() on the application and passing the middleware as a parameter.

To parse x-www-form-urlencoded request bodies, Express.js can use urlencoded() middleware from the body-parser module.

const bodyparser = require('body-parser')
app.use(bodyparser.urlencoded({extended: false}))

HINTS
Here is how we can print characters backwards (just one way to do it):

req.body.str.split('').reverse().join('')
Extended set to true (qs) or false (querystring) determines the parser module.

Read more about Connect middleware here:
https://github.com/senchalabs/connect#middleware
The documentation of the body-parser module can be found here:
https://github.com/expressjs/body-parser

6. Create a Express server that uses static files. Place an `index.html`, an image and a css file in a directory named `public`. Use the index.html to display the photo and some caption text in red or another color using css.

7. Shopping List

We will be building a simple application where we will store a shopping list. You should use an **array** to store your items in the shopping list.

Our application should have the following routes:

1. GET /items - this should render a list of shopping items.
2. POST /items - this route should accept form data and add it to the shopping list.
3. GET /items/:id - this route should display a single item's name and price
4. PATCH /items/:id, this route should modify a single item's name and/or price
5. DELETE /items/:id - this route should allow you to delete a specific item from the array.

Happy Coding!!!!