FS II Notes - Day 6 (Oct 12, 2019)

Server-Side Javascript Programming - NodeJS

Server Development Template Setup

- Verify NodeJS installation and NPM installation
- Project set up
  - NPM
  - Project Initialization

npm init

External Information Link:

https://www.w3schools.com/nodejs/nodejs_http.asp

TASK:

Demonstrate management of development environment for client/server testing setup. Demonstrate file management for project testing and demonstration.

- Verify that NodeJS and NPM are installed and functional within your working environment
- Create a development folder that will be dedicated to client-side code exercises named “js_server_side” - This is our “Development Exercise Root Repository” (DERR) for server-side javascript
- Create the following within the DERR:

0_app/client/index.html

0_app/client/js

0_app/client/css

0_app/server/

- Initialize server/ as a NPM project

Environment Testing

- Dates

```
d = new Date();
document.getElementById("demo").innerHTML = d;
```

- NodeJS server

You can start NodeJS from the folder where the file is located:

> node filename.js

```
// Variables are being set as constants for easier access to the app
// For production this set up may not be optimal

//  Require the HTTP module for basic server functionality
const http = require("http");

//  Set the hostname, can also be an address (e.g. 127.0.0.1)
const hostname = 'localhost';

//  Set the port, for this test we're using 999
//  Common port usage reference:
//  https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers
const port = 999;

//Create a server object and listen on the port we set for requests
const server = http.createServer((req, res) => {

  //Set the response variables
  //  HTTP status code 200 OK
  res.statusCode = 200;
  //  HTTP header as plain text
  res.setHeader('Content-Type', 'text/plain');
  //  Text to be displayed
  res.end('Hello World\n');
});

//  Now that everything is set up, the server listens for requests
//  When a request comes in to the server it will pass the port and host info
```

```
//  The values passed when the function is called can then be displayed
server.listen(port, hostname, () => {
  console.log(`http://${hostname}:${port}/` + ' ok');
});
```

External Information Link:

https://www.w3schools.com/js/js_dates.asp

TASK:

Demonstrate simple client/server testing setup. Demonstrate file management for project development.

- Create a file named “test.js” in 0_app/server/
- Create a javascript test file for testing different ports, one of the ports should be for http and others should handled using variables. Display information to the browser that provides useful, human-readable information about the server.
- JS error handling should always be used

Development Environment

- Dependencies
  - Express Install

> npm install express

```
const port = 999;
const express = require('express')
const app = express();

app.get('/', (req, res) => {
  res.send('Hello World')
});

app.listen(port, () => {
  console.log(‘ok on ‘ + port)
});
```

External Information Link:

https://docs.npmjs.com/files/package-lock.json

TASK:

Demonstrate installation of dependencies. Demonstrate file management for project testing.

- Create a project folder named “1_express”
- Create javascript test file for a project with express.js as a dependency that displays useful information to the browser and console in human-readable format

Query Strings

- Query String sources
  - Form submitting
  - User generated
- Parsing the query string

```
//  Create the server to work with the query string
http.createServer(function (req, res) {
  //  Send an OK header since everything is fine here
  res.writeHead(200, {'Content-Type': 'text/html'});
  //  Split the URL into parts
  var queryData = url.parse(req.url, true).query;

  //  Now we have an object we can work with
  //  We can execute code here for that purpose
  var returnValue = "ok";
  //  End the response and send back returnValue
  res.end(returnValue);
}).listen(port);
```

External Information Link:

https://www.w3schools.com/nodejs/nodejs_http.asp

TASK:

Demonstrate use of Date objects for displaying information to users. Demonstrate understanding of query string values.

- Make a copy of the template folder named "2_query_strings" in the DERR. The folder should be at the same directory level as the template
- Create a NodeJS web server that listens for requests on port 80 and returns human-readable information about the server including:
  - The date and time of the request
  - The URL requested
  - The names and values of query string data
  - Handling of no query string data

Serving Files

```
// Setting up modules
var http = require('http');  //  HTTP
var url = require('url');  //  URL handling
var fs = require('fs');  //  File serving

http.server(function (req, res) {
  var queryData = url.parse(req.url, true);  //  parse the URL data
  var filename = queryData.pathname + filename;  //  assign a file name for retrieval
//  Read the file. If it exists send the contents back, if not return a 404 error
 fs.readFile(filename, function(err, data) {
    if (err) {
      res.writeHead(404, {'Content-Type': 'text/html'});
      return res.end("404 Not Found");
    }
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    return res.end();
  });
}).listen(port);
```

External Information Link:

https://www.w3schools.com/nodejs/nodejs_url.asp

TASK:

Demonstrate understanding of relative file handling. Demonstrate serving pages based on user input.

- Make a copy of the template folder named “3_url_handling” in the DERR. The folder should be at the same directory level as the template
  - Create a NodeJS web server that listens for requests on port 80 and serves files based on the URL request:
    - The files should be served from the client/ directory of your DERR
    - There should be at least three page options for the user
    - The user should be presented with choices for navigation if no URL or query string data is provided (default page)

Data Formats

- JSON

  - Syntax

name/value pairs
comma separates data
curly braces hold objects
square brackets hold arrays

Values *must* be of type: string, number, JSON object, array, boolean or null

JSON files end with .json

JSON text MIME type is “application/json"

  - Parsing

```
// This will convert JSON text data into a javascript object
   var objectJSON = JSON.parse(textJSON);
```

External Information Links:

https://www.w3schools.com/js/js_json_html.asp

https://www.w3schools.com/js/js_json_parse.asp

TASK:

Demonstrate ability to work with variables and functions. Demonstrate understanding of JSON data.

- Make a copy of the template folder named “4_JSON” in the DERR. The folder should be at the same directory level as the template folder
- Create a NodeJS web server that listens for requests on port 80 and does the following:
  - Store an array of data that contains at least 6 key/value pairs using the keys: firstName, lastName, email, address, phone, verified. The variables should contain human-readable values
  - Return the value from the array based on the query string where:
    - If no key matching the query string is provided then no value is provided but the user is informed

- If there is a matching key the user is returned the value in the form of raw JSON that is readable within the browser

FINAL TASK:

- Using your Git account push your js_server_side DERR to a repo named “FS2_JS_NodeJS”