EAST TechSprint Log:

Use this document to log your work for the TechSprint period. Remember, your log should reflect adequate effort to achieve growth in the skills you are learning. This work is to be done **in addition** to any work that you are doing or skills you are developing for your EAST Project (although the work can be complementary). Each log entry should follow the following format:

- Date
- Description of learning goal or task
- Link to tutorial resource(s)
- Example of **your work** based on the tutorial and goals (screenshot preferred), embedded in this document
- Reflection on what you learned, challenges or other important elements

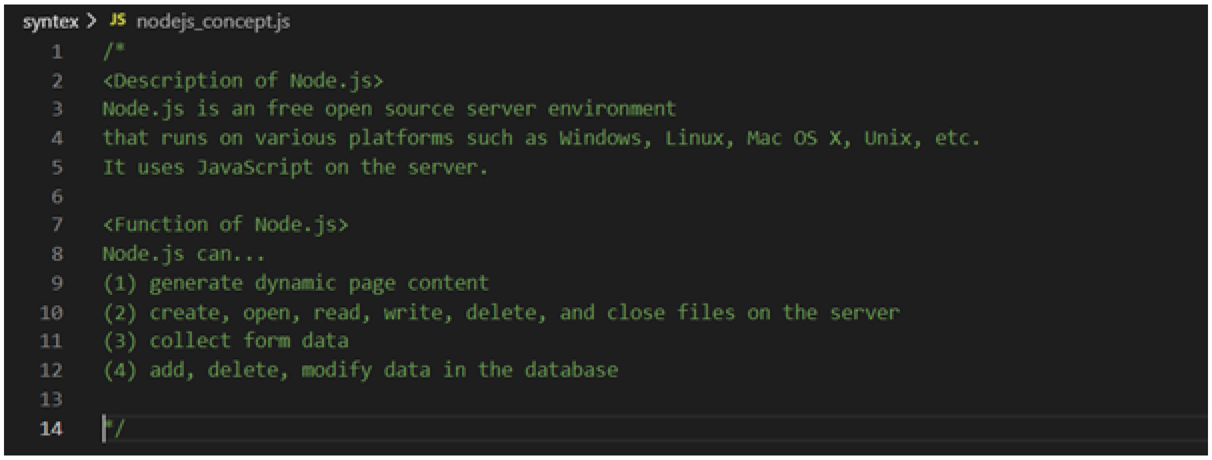
This log will be due by 11:59 PM on the Sunday following the TechSprint period.

Date: 2/1

Task: Learning about the concept of Node.js.

Link to Source:

- https://www.youtube.com/watch?v=3RS_A87IAPA&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=1
- https://www.youtube.com/watch?v=vT51SuzozLc&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=2
- https://www.w3schools.com/nodejs/nodejs_intro.asp



syntex > nodejs_concept.js

```
/*
<Description of Node.js>
Node.js is an free open source server environment
that runs on various platforms such as Windows, Linux, Mac OS X, Unix, etc.
It uses JavaScript on the server.

<Function of Node.js>
Node.js can...
(1) generate dynamic page content
(2) create, open, read, write, delete, and close files on the server
(3) collect form data
(4) add, delete, modify data in the database

*/
```

Reflection: Node.js is not only a useful server that enables easy and fast code change but also a catalyst that induces users' active participation in web applications. It has a large potential to assemble the information effectively to create active web sites.

Date: 2/1

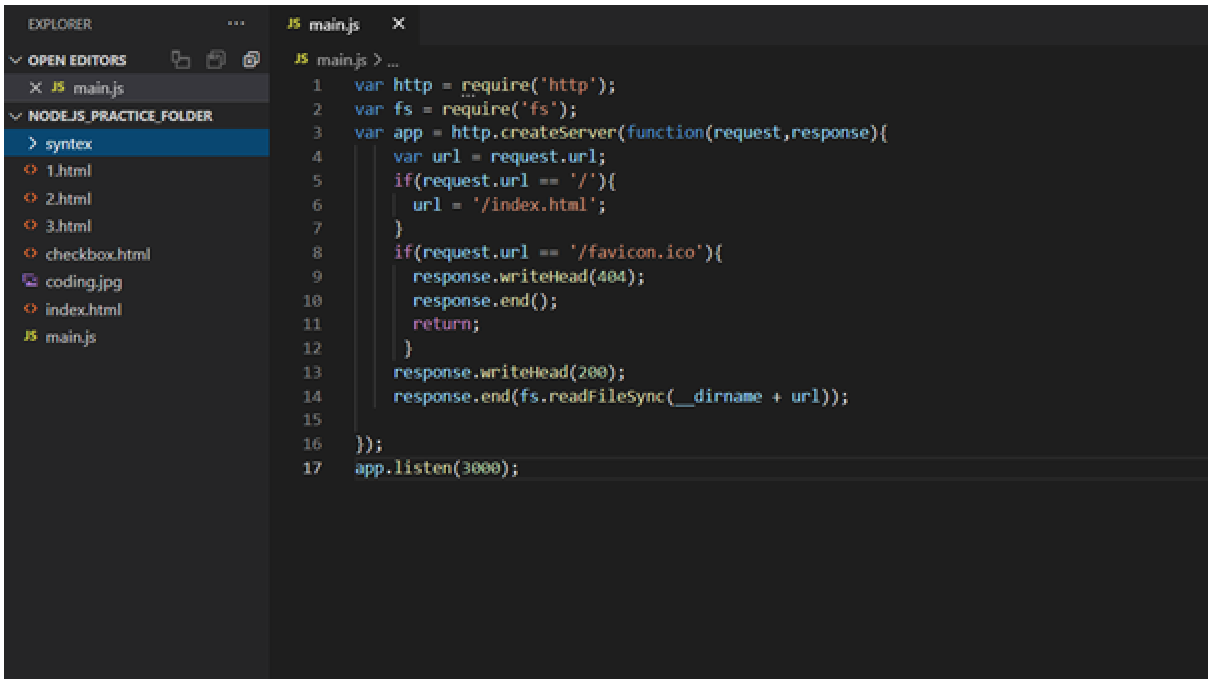
Task: Learning about how to run Node.js using JavaScript

Link to Source:

- https://www.youtube.com/watch?v=0P8r0XqYfVw&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=3
- https://www.youtube.com/watch?v=QVR5HASMdfk&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=4

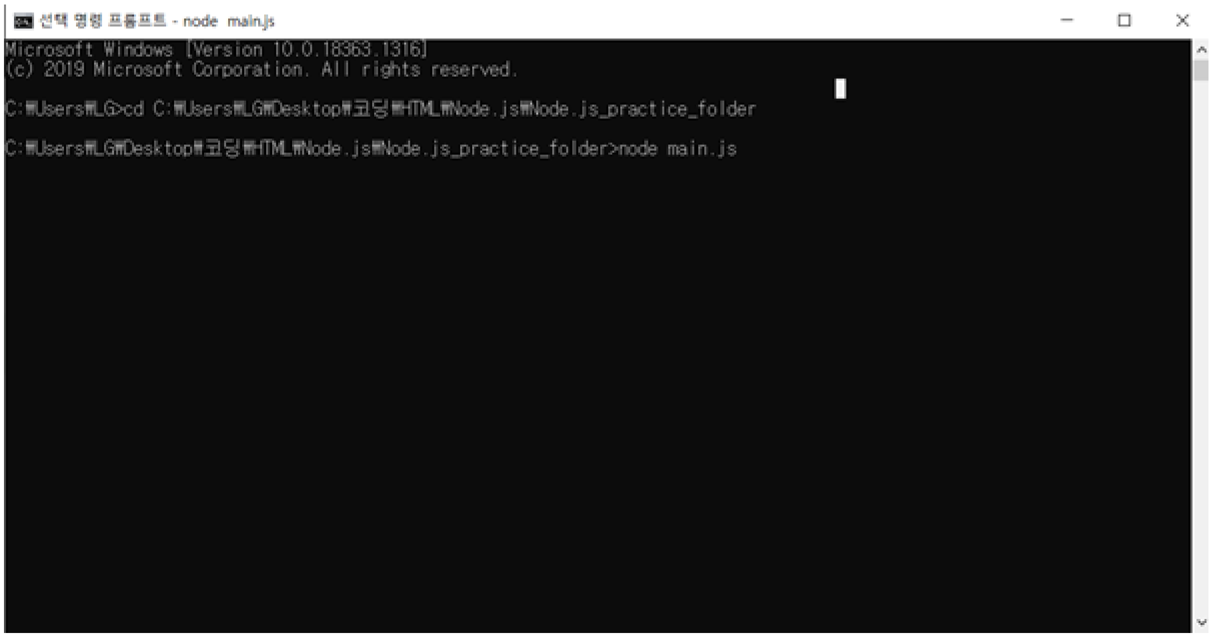
1. Install Node.js from https://nodejs.org/en/ .

2. Create an example of HTML and JS to test whether node.js works on my laptop.



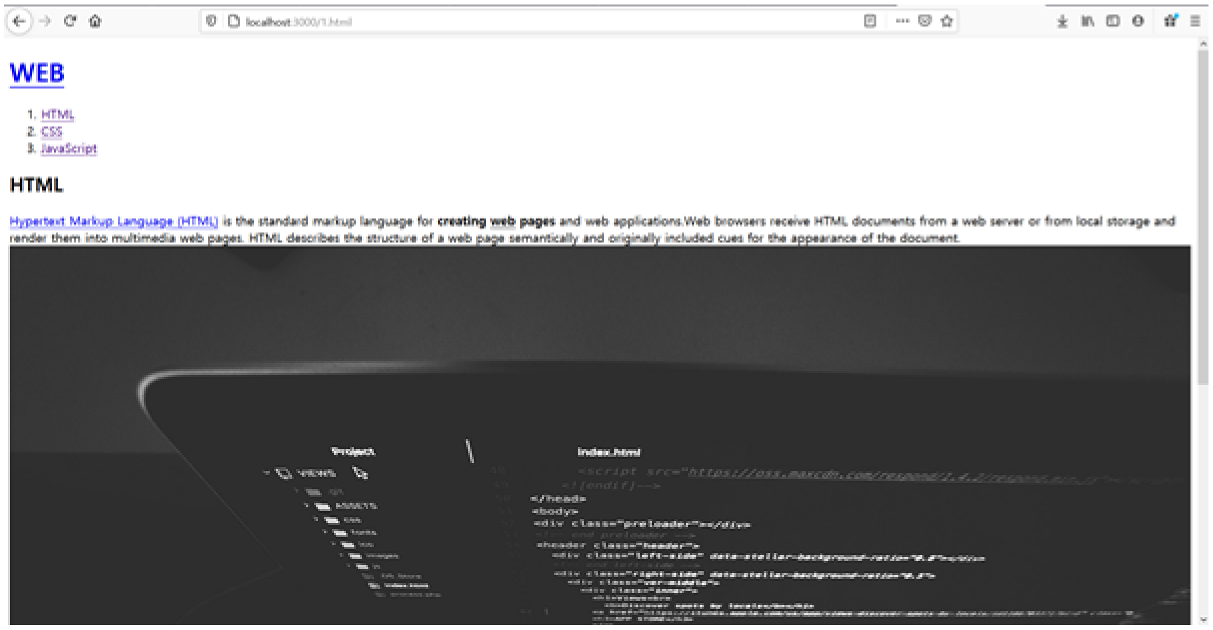
```
var http = require('http');
var fs = require('fs');
var app = http.createServer(function(request,response){
    var url = request.url;
    if(request.url == '/'){
      url = '/index.html';
    }
    if(request.url == '/favicon.ico'){
      response.writeHead(404);
      response.end();
      return;
    }
    response.writeHead(200);
    response.end(fs.readFileSync(__dirname + url));

});
app.listen(3000);
```

3. Use command prompt (cmd) to run the example files.



```
Microsoft Windows [Version 10.0.18363.1316]
(c) 2019 Microsoft Corporation. All rights reserved.

C:₩Users₩LG>cd C:₩Users₩LG₩Desktop₩코딩₩HTML₩Node.js₩Node.js_practice_folder

C:₩Users₩LG₩Desktop₩코딩₩HTML₩Node.js₩Node.js_practice_folder>node main.js
```

4. Enter into http://localhost:3000/ on the web browser and check the server worked. (put the port as 3000 in the JS code.)



WEB

1. HTML
2. CSS
3. JavaScript

HTML

Hypertext Markup Language (HTML) is the standard markup language for **creating web pages** and web applications.Web browsers receive HTML documents from a web server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

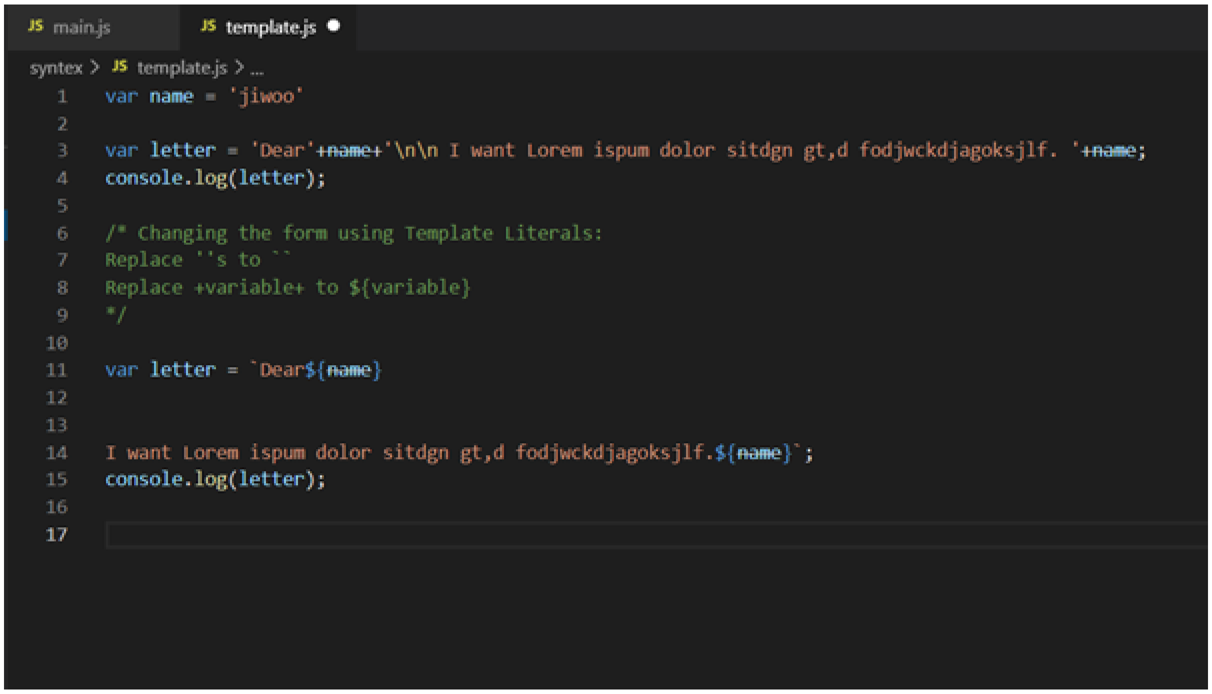
Reflection: When adopting a server system such as Node.js, recalling the stored code and information is possible. Node.js helps to run the server with javascript and change its contents.

Date: 2/2

Task: Learning about “Template Literals” (JavaScript Grammar)

Link to Source:

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals
- https://www.youtube.com/watch?v=9GWjNcV96Ws&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=13



```javascript
var name = 'jiwoo'

var letter = 'Dear'+name+'\n\n I want Lorem ispum dolor sitdgn gt,d fodjwckdjagoksjlf. '+name;
console.log(letter);

/* Changing the form using Template Literals:
Replace ''s to ``
Replace +variable+ to ${variable}
*/

var letter = `Dear${name}


I want Lorem ispum dolor sitdgn gt,d fodjwckdjagoksjlf.${name}`;
console.log(letter);

```

Template literals are string literals allowing embedded expressions.

- Multi-line strings and string interpolation is possible.
- Template literals are enclosed by the backtick (` `) instead of quotes.
- Can contain placeholders using (${expression}).

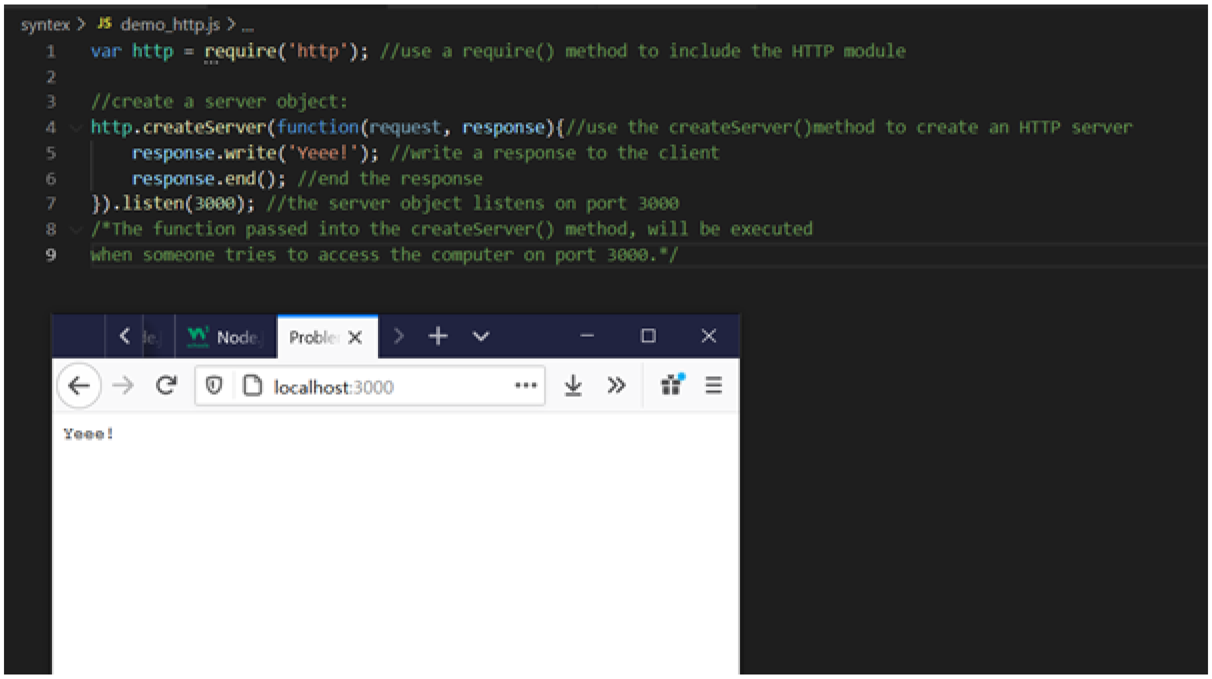
Reflection: When writing strings in JavaScript, it is useful to adopt template literals to write them decently. Template literals are used widely when expressing contents with strings.

Date: 2/2

Task: Learning Node.js HTTP Module

Link to Source:

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



```
var http = require('http'); //use a require() method to include the HTTP module

//create a server object:
http.createServer(function(request, response){//use the createServer()method to create an HTTP server
    response.write('Yeee!'); //write a response to the client
    response.end(); //end the response
}).listen(3000); //the server object listens on port 3000
/*The function passed into the createServer() method, will be executed
when someone tries to access the computer on port 3000.*/
```

```
Yeee!
```

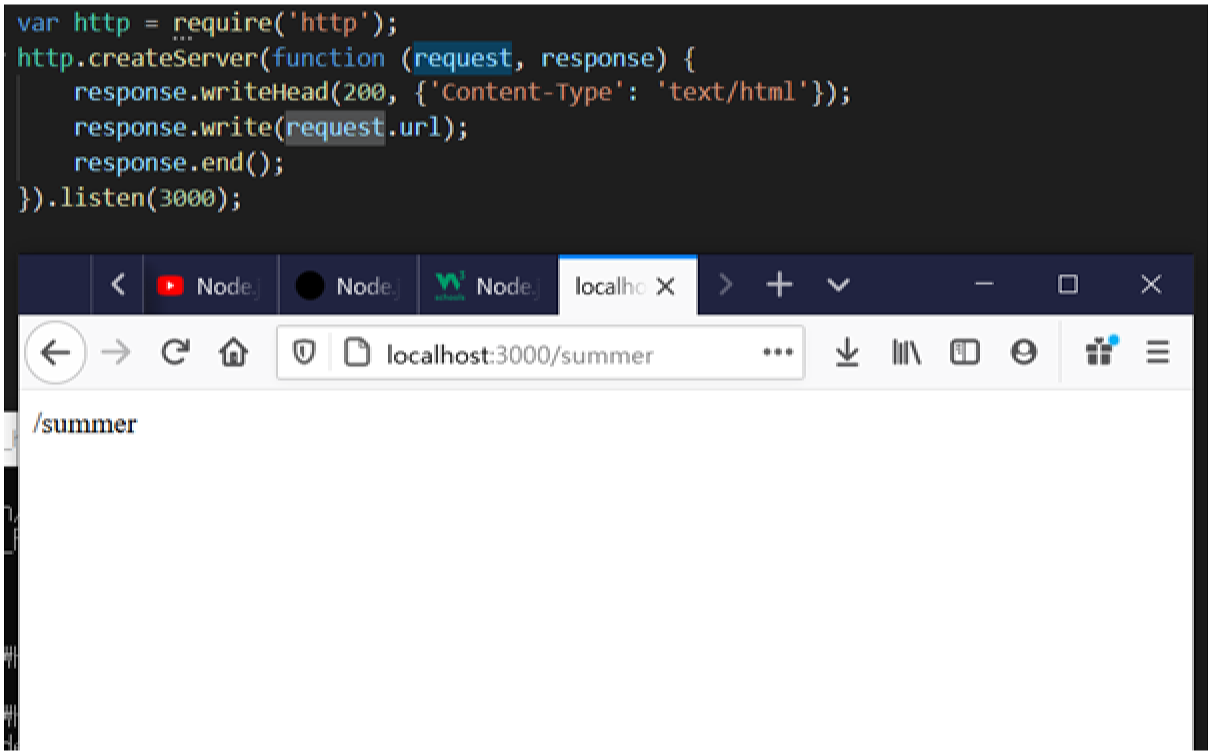
Reflection: When using modules in Node.js, it creates a server and writes responses to the client. Responses that clients get can be seen differently with the server.

Date: 2/3

Task: Showing different information by changing the query string

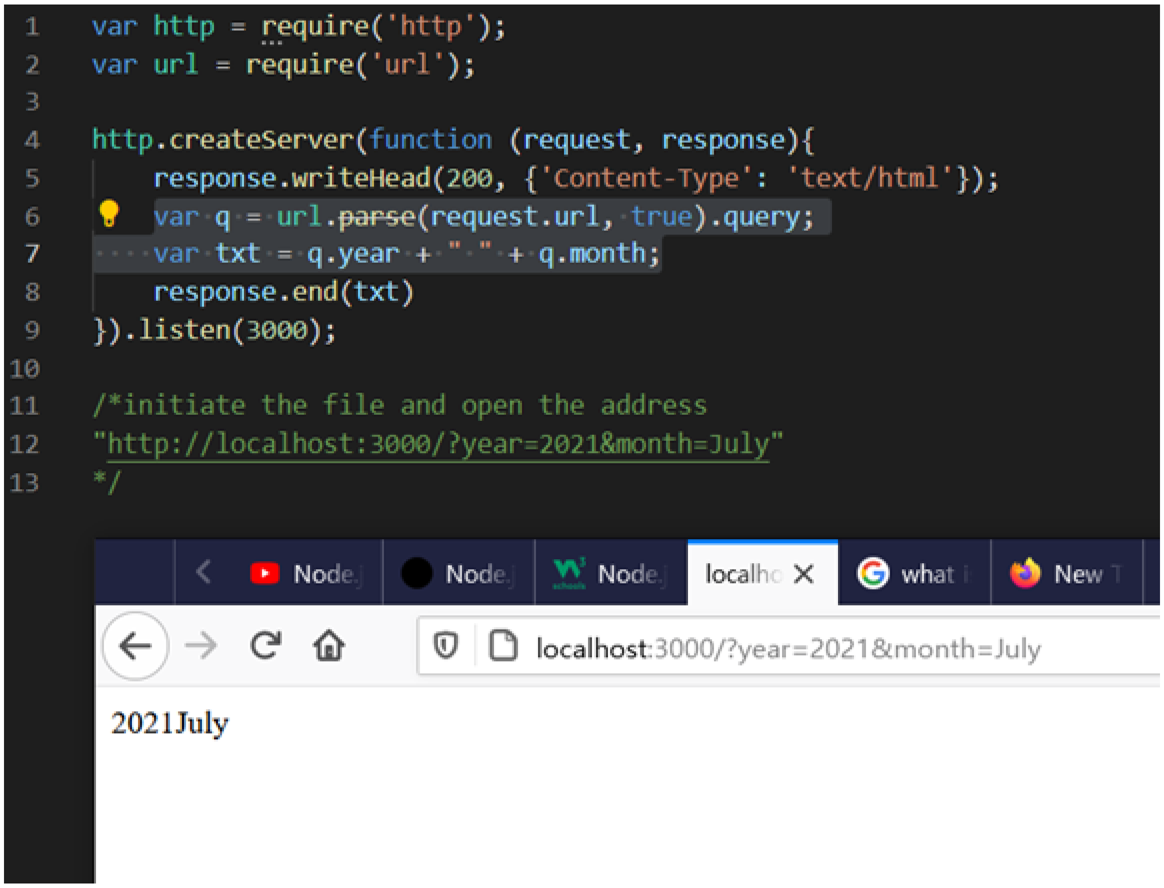
Link to Source:

- https://www.youtube.com/watch?v=Zhbvui_T9VY&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=14
- https://www.youtube.com/watch?v=_bh8ztMJIPE&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=15
- https://www.w3schools.com/nodejs/nodejs_http.asp



```
var http = require('http');
http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(request.url);
    response.end();
}).listen(3000);
```

```
/summer
```



```
var http = require('http');
var url = require('url');

http.createServer(function (request, response){
    response.writeHead(200, {'Content-Type': 'text/html'});
    var q = url.parse(request.url, true).query;
    var txt = q.year + " " + q.month;
    response.end(txt)
}).listen(3000);

/*initiate the file and open the address
"http://localhost:3000/?year=2021&month=July"
*/
```

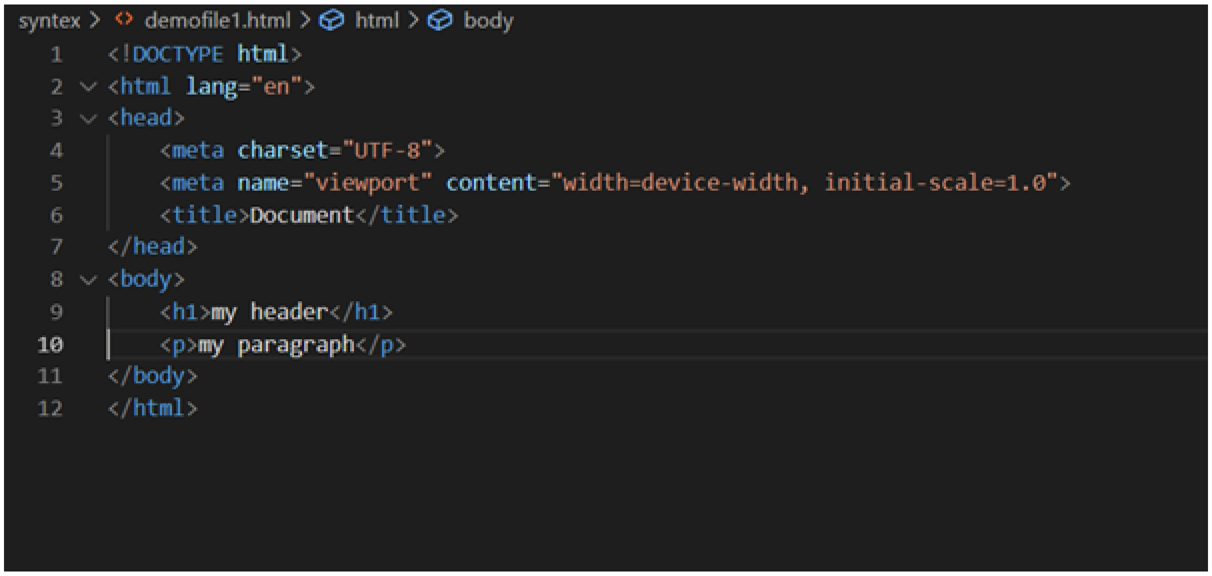
```
2021July
```

Reflection: A query string is a part of a URL that assigns values to specified parameters. Parsing and changing the query string from the URL is effective because they can make the URL contain id values, which contains information to change the contents a web page shows.

Date: 2/3

Task: Learning how to make a server read html files using Node.js.

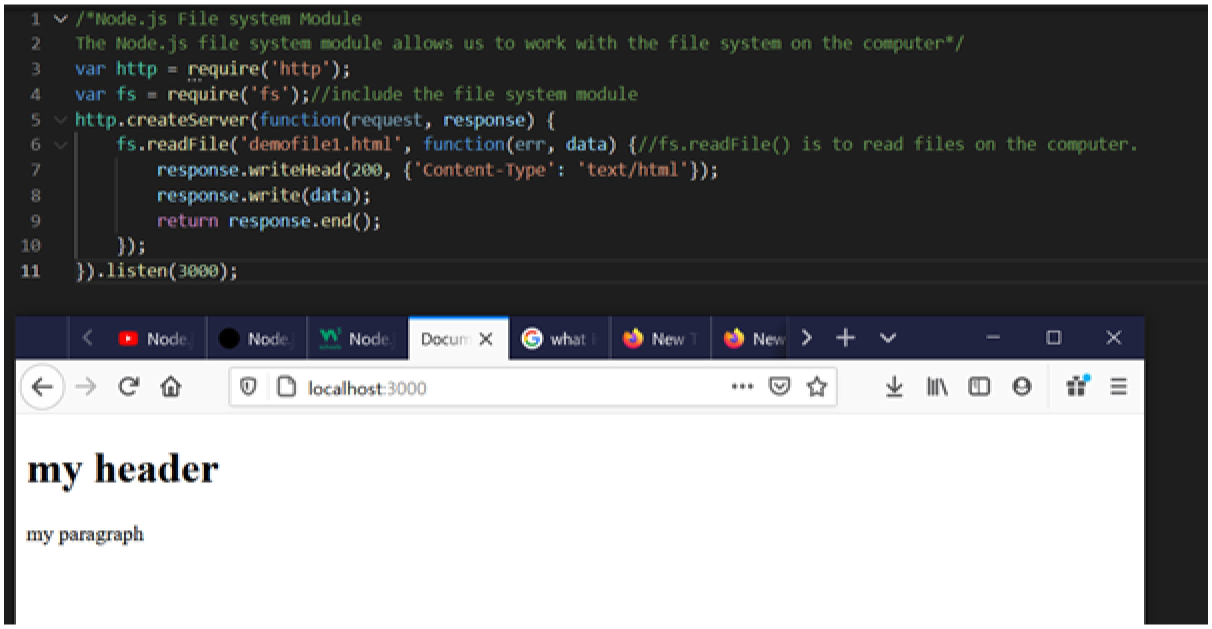
Link to Resource: https://www.w3schools.com/nodejs/nodejs_filesystem.asp



syntex > demofile1.html > html > body

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>my header</h1>
    <p>my paragraph</p>
</body>
</html>
```

(Screenshot above is an example html file in which its data was used to show on the browser.)



```
/*Node.js File system Module
The Node.js file system module allows us to work with the file system on the computer*/
var http = require('http');
var fs = require('fs');//include the file system module
http.createServer(function(request, response) {
    fs.readFile('demofile1.html', function(err, data) {//fs.readFile() is to read files on the computer.
        response.writeHead(200, {'Content-Type': 'text/html'});
        response.write(data);
        return response.end();
    });
}).listen(3000);
```

my header

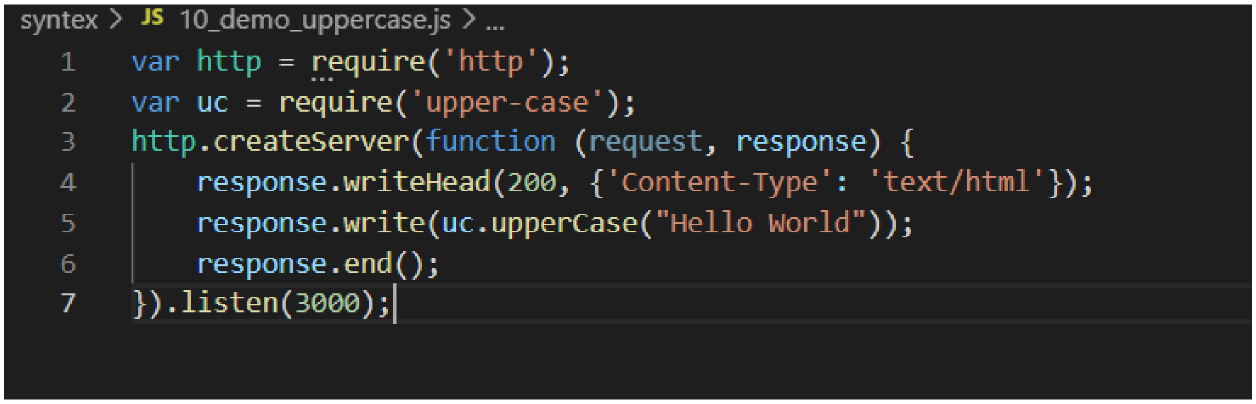
my paragraph

fs : file system module

Reflection: When using a file system module, the server can read html files and show them when it is requested. This makes it easier for developers to revise their codes or add contents on their web pages. When I built a website before, I did not know how to convert the contents in which I had to make separate html files. By adapting the file system module, improvement of development method is expected.

Date: 2/5
Task: Learning about Node Package Manager (NPM)
Link to Resource:

- https://www.w3schools.com/nodejs/nodejs_npm.asp
- https://developer.mozilla.org/en-US/docs/Glossary/Node.js#node_package_manager_npm



syntex > JS 10_demo_uppercase.js > ...

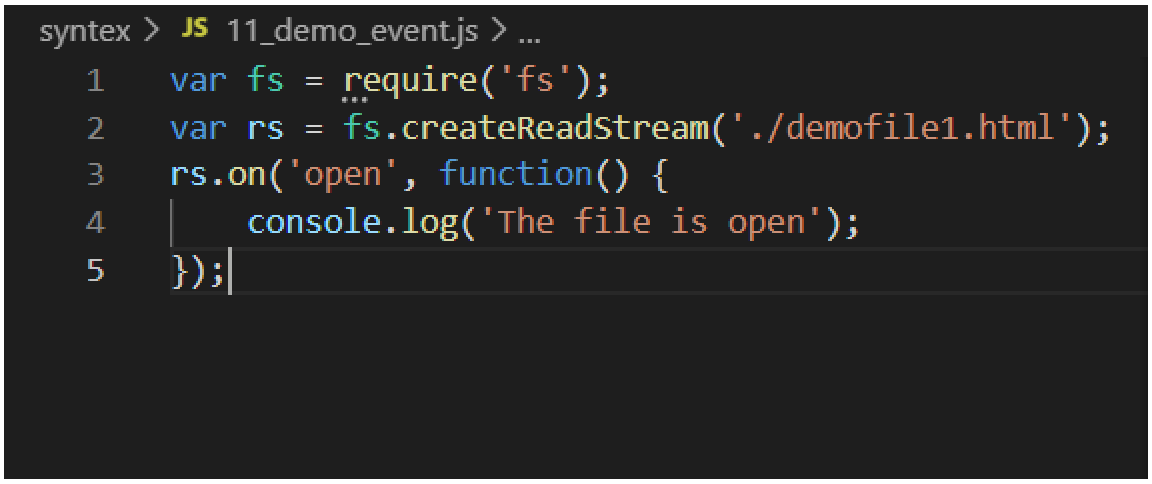
```
var http = require('http');
var uc = require('upper-case');
http.createServer(function (request, response) {
    response.writeHead(200, {'Content-Type': 'text/html'});
    response.write(uc.upperCase("Hello World"));
    response.end();
}).listen(3000);
```

NPM is a package manager that enables users to download packages into a node_modules folder.

https://www.npmjs.com/ supplies description of various packages. It also helps distinguish popular and well-known packages for better usage.
Reflection: NPM provides various packages in Node.js. It makes Javascript development easier and productive.

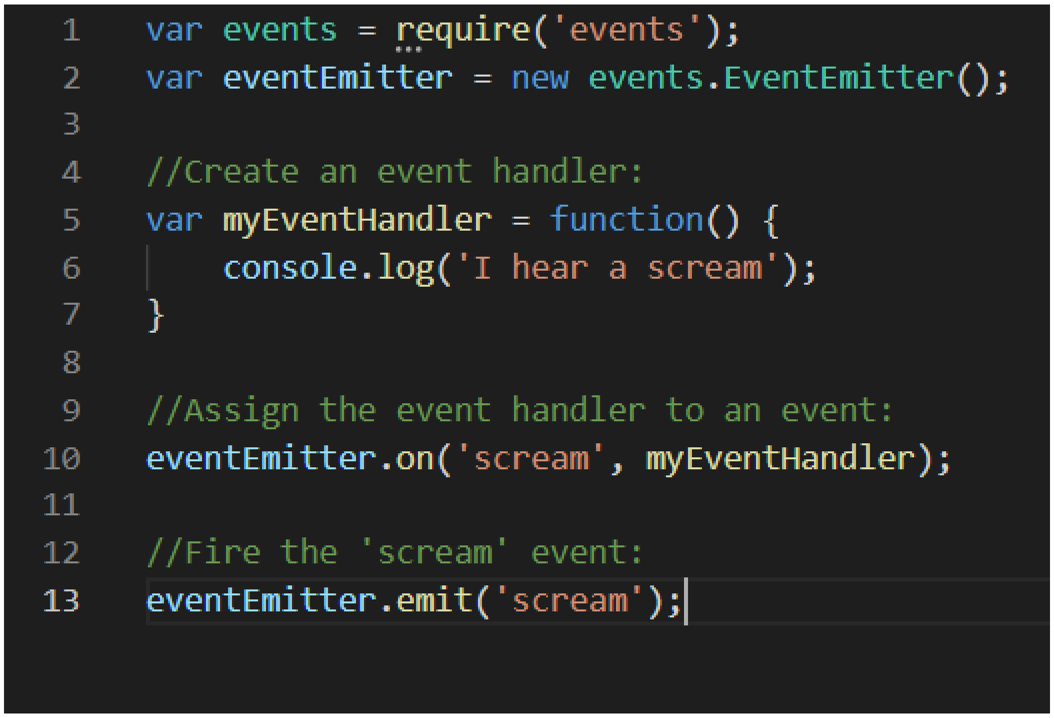
Date: 2/6-2/7
Task: Learning Node.js Events
Link to Resource:

- https://www.w3schools.com/nodejs/nodejs_events.asp
- https://www.tutorialspoint.com/nodejs/nodejs_event_emitter.htm



syntex > JS 11_demo_event.js > ...

```
var fs = require('fs');
var rs = fs.createReadStream('./demofile1.html');
rs.on('open', function() {
    console.log('The file is open');
});
```



```
var events = require('events');
var eventEmitter = new events.EventEmitter();

//Create an event handler:
var myEventHandler = function() {
    console.log('I hear a scream');
}

//Assign the event handler to an event:
eventEmitter.on('scream', myEventHandler);

//Fire the 'scream' event:
eventEmitter.emit('scream');
```

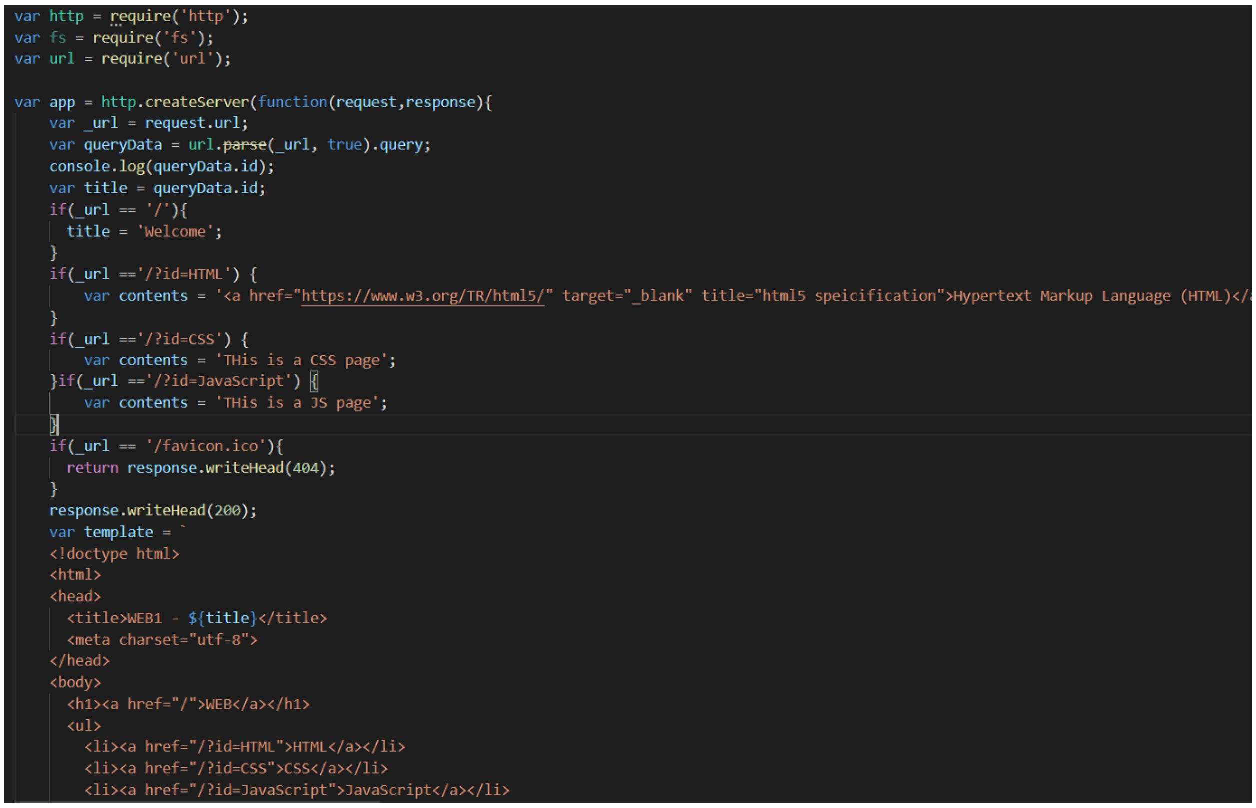
* on(event, listener) - adds a listener at the end of the listeners array for the specified event. Returns emitter, so calls can be chained.
* emit(event, [arg1], [arg2],[...]) - Execute each of the listeners in order with the supplied arguments. Returns true if the event had listeners, false otherwise.

Reflection: Node.js provides an event handler to create, assign, fire, and remove events. When using methods of events emitter, events emitted by objects in a Node can be detected. It is useful to discern errors. Further understanding about events is necessary.

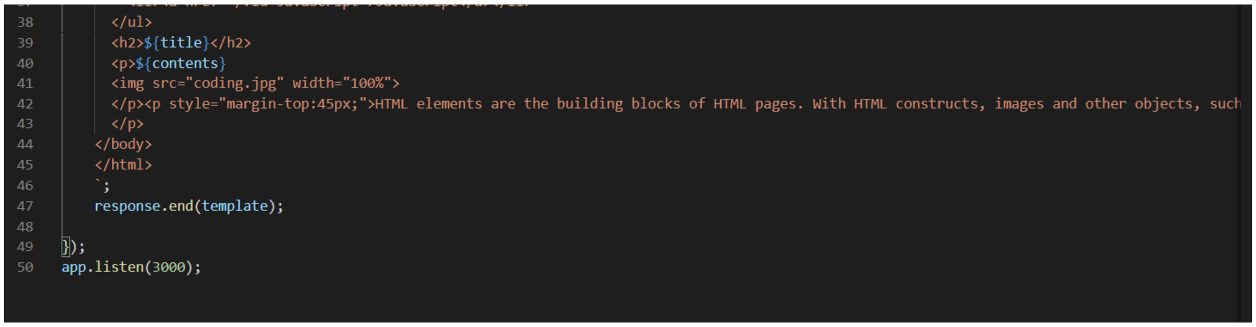
Date: 2/8
Task: Learning how to use queryData id to change contents and create a web application
Link to Resource:

- https://www.youtube.com/watch?v=s8HrD5aOXmE&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=16
- https://opentutorials.org/module/3549/21047
- https://www.youtube.com/watch?v=X9In3UA84O4&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=18

#1

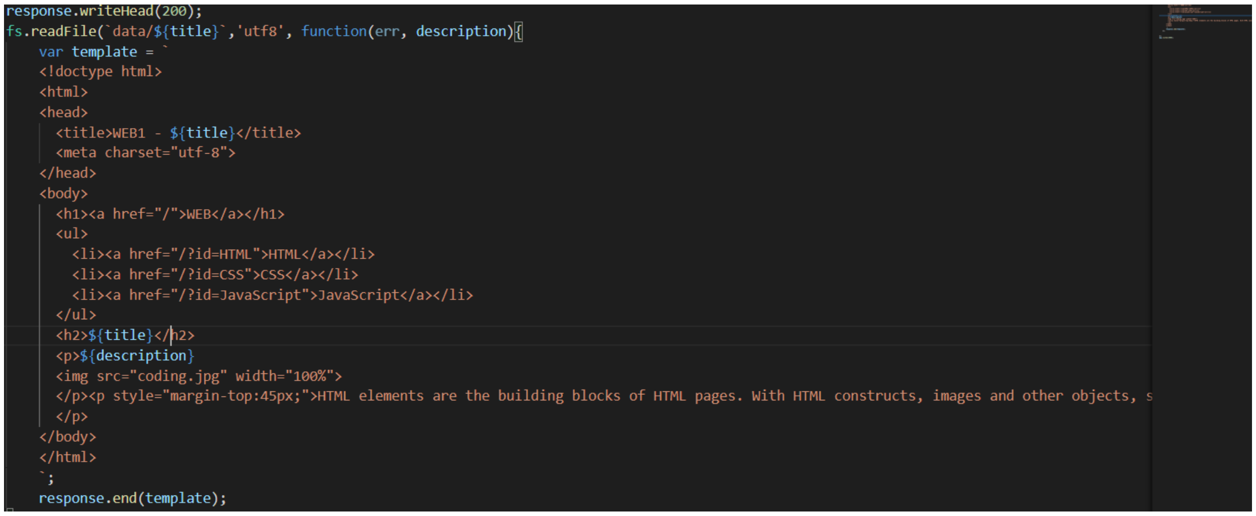


```
var http = require('http');
var fs = require('fs');
var url = require('url');

var app = http.createServer(function(request,response){
    var _url = request.url;
    var queryData = url.parse(_url, true).query;
    console.log(queryData.id);
    var title = queryData.id;
    if(_url == '/'){
      title = 'Welcome';
    }
    if(_url =='/?id=HTML') {
        var contents = '<a href="https://www.w3.org/TR/html5/" target="_blank" title="html5 speicification">Hypertext Markup Language (HTML)</
    }
    if(_url =='/?id=CSS') {
        var contents = 'THis is a CSS page';
    }if(_url =='/?id=JavaScript') {
        var contents = 'THis is a JS page';
    }
    if(_url == '/favicon.ico'){
      return response.writeHead(404);
    }
    response.writeHead(200);
    var template = `
    <!doctype html>
    <html>
    <head>
      <title>WEB1 - ${title}</title>
      <meta charset="utf-8">
    </head>
    <body>
      <h1><a href="/">WEB</a></h1>
      <ul>
        <li><a href="/?id=HTML">HTML</a></li>
        <li><a href="/?id=CSS">CSS</a></li>
        <li><a href="/?id=JavaScript">JavaScript</a></li>
```



```
            </ul>
            <h2>${title}</h2>
            <p>${contents}
            <img src="coding.jpg" width="100%">
            </p><p style="margin-top:45px;">HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such
            </p>
        </body>
        </html>
        `;
        response.end(template);

    });
    app.listen(3000);
```

#2



```
response.writeHead(200);
fs.readFile(`data/${title}`,'utf8', function(err, description){
    var template = `
    <!doctype html>
    <html>
    <head>
      <title>WEB1 - ${title}</title>
      <meta charset="utf-8">
    </head>
    <body>
      <h1><a href="/">WEB</a></h1>
      <ul>
        <li><a href="/?id=HTML">HTML</a></li>
        <li><a href="/?id=CSS">CSS</a></li>
        <li><a href="/?id=JavaScript">JavaScript</a></li>
      </ul>
      <h2>${title}</h2>
      <p>${description}
      <img src="coding.jpg" width="100%">
      </p><p style="margin-top:45px;">HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, s
      </p>
    </body>
    </html>
    `;
    response.end(template);
```

Reflection: By reading txt files in Node.js, only one js file can contain various contents. There is no need to create the same html files that have only different texts. Contents on the web can consist of text files, not individual html files. Also, if there are changes in contents of txt files, there is no need to close and rerun node because the server reads txt files every time we reload the site.

Date: 2/10-2/14
Task: Creating a simple HTML application using Node.js
Link to Resource:

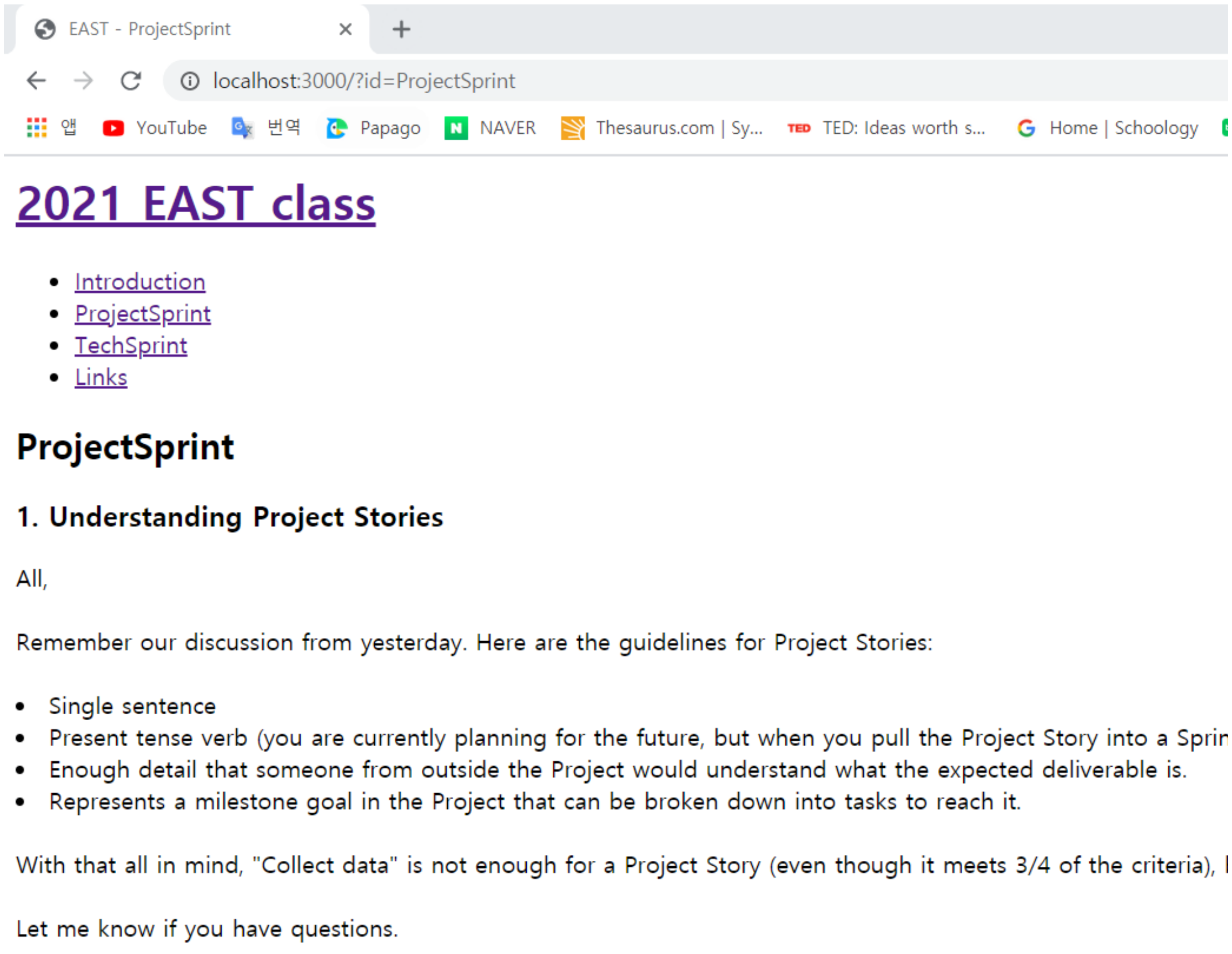
- https://www.youtube.com/watch?v=krfmrBsWqzs&list=PLuHgQVnccGMA9QQX5wqj6ThK7t2tsGxjm&index=25

```
var http = require('http');
var fs = require('fs');
var url = require('url');

var app = http.createServer(function(request, response){
    var _url = request.url;
    var queryData = url.parse(_url, true).query;
    var pathname = url.parse(_url, true).pathname;
    if(pathname === '/') {
        if(queryData.id ===undefined){
            fs.readFile(`../data/${queryData.id}.txt`, 'utf8',
function(err, description){
                var title = 'Welcome';
                var description = 'Welcome to EAST class';
                var template = `<!doctype html>
                <html>
                <head>
                  <title>EAST - ${title}</title>
                  <meta charset="utf-8">
                </head>
                <body>
                  <h1><a href="/">2021 EAST class</a></h1>
                  <ul>
                    <li><a href="/?id=Introduction">Introduction</a></li>
                    <li><a
href="/?id=ProjectSprint">ProjectSprint</a></li>
                    <li><a href="/?id=TechSprint">TechSprint</a></li>
                    <li><a href="/?id=Links">Links</a></li>
                    </ul>
                  <h2>${title}</h2>
                  <p>${description}</p>
                </body>
                </html>
                `;
                response.writeHead(200);
                response.end(template);
              });
            } else {
              fs.readFile(`../data/${queryData.id}.txt`, 'utf8',
function(err, description){
                var title = queryData.id;
                console.log(description);
                var template = `
                <html>
                <head>
                  <title>EAST - ${title}</title>
                  <meta charset="utf-8">
                </head>
                <body>
```

```
                  <h1><a href="/">2021 EAST class</a></h1>
                  <ul>
                    <li><a href="/?id=Introduction">Introduction</a></li>
                    <li><a
href="/?id=ProjectSprint">ProjectSprint</a></li>
                    <li><a href="/?id=TechSprint">TechSprint</a></li>
                    <li><a href="/?id=Links">Links</a></li>
                    </ul>
                  <h2>${title}</h2>
                  <p>${description}</p>
                </body>
                </html>
                `;
                response.writeHead(200);
                response.end(template);
              });
            }
          } else {
            response.writeHead(404);
            response.end('Not found');
          }



      });
      app.listen(3000);
```

I have created a HTML application about EAST class to apply what I learned during this sprint. The purpose of this website is to gather information about the things students need to know for the class.

Example of the webpage:



2021 EAST class

- Introduction
- ProjectSprint
- TechSprint
- Links

ProjectSprint

1. Understanding Project Stories

All,

Remember our discussion from yesterday. Here are the guidelines for Project Stories:

- Single sentence
- Present tense verb (you are currently planning for the future, but when you pull the Project Story into a Sprir
- Enough detail that someone from outside the Project would understand what the expected deliverable is.
- Represents a milestone goal in the Project that can be broken down into tasks to reach it.

With that all in mind, "Collect data" is not enough for a Project Story (even though it meets 3/4 of the criteria), I

Let me know if you have questions.

Reflection: When applying the file reading function of node.js, it is easier and faster to create a webpage using HTML. It is important to set appropriate ids to clarify and improve the quality of contents.