**Node.js and Express.js**

**Node.js:**

**Backend:**

Backend: Consists of server (provide html/css/JS), database (stores user data) and applications.

Division of front end and back end is useful to keep your code private and prevent theft. Front end is built for the common user and is easy to understand. With frontend only, all the code (HTML/CSS/JS) is sent from the server to the user’s web browser. By using backend, the calculations can be done server-side and only the results are sent back to the user’s browser. This helps conceal our code.

Back end languages have frameworks which allow for faster process of coding (Node.js > Express.js)

**Node.js**

With regular JS, we used javascript to interact with the users webpage. With Node.JS, we can now use javascript to interact with the user’s files and documents. Node.JS also lets us run the code on a server to allow for faster execution of code. The processed information is then transferred to the user to allow for increased speeds.

**REPL (read evaluation print loop)**

Allows us to run JS in CLI line by line.

Type in node in CLI to enter REPL mode.

You can exit node by hitting Ctrl+C twice or typing in .exit

To clear the console, type: clear

**Node methods in CLI:**

node JSFileName.js

runs JS file JSFileName.js in your CLI using Node.js

You can type in part of a method name and hit tab to autocomplete it or hit tab twice for a full list of all the methods that include that partial name.

**Node.js internal modules:** <https://nodejs.org/api/>

You can use internal Node modules to perform various JS functions.

One of the modules is the file system module. To gain access, you must first run this JS code:

const fs = require('fs');

This requests the ‘fs’ module and stores it in the constant fs

These modules are useful for accessing paths, opening, editing, creating, deleting or copying files.

**Node.js external modules:**

To create a package, type: npm init :into your CLI in the directory you want

Fill in the rest of the information to create your package.json file in the directory specified.

External modules can be installed from <https://www.npmjs.com/> (node package modules)

To request these files, you must run the file request line in your CLI in the directory you want.

Ex: npm install superheroes

These external modules will show up on the package.json file as dependencies.

You can now run the external module commands by first requiring the module and then using the module’s methods.

**Express.js:**

Express.js is a framework designed for Node.js to speed up coding and reduce repeated code.

To install Express.js, it is the same as installing an external module. Just initialize a NPM on the directory you want (npm init) using your CLI. Specify a JS entry point file. Then type: npm install express :in your CLI while in the same directory you want to save Express.js in.

To use Express.js, in your JS entry point, you need to write:

const express = require(‘express’); To request the express module

const app = express(); To save the express() function to app()

We can use Express.js to run our own locally hosted server.

App.listen(serverFrequency, function(){anonymous function code;});

This stars our server on <http://localhost:serverFrequency/>

This page is the root of our server.

Any code placed in the parameter is displayed on your CLI.

We can also change what is displayed on our web page

app.get(“location”, function(request, response){code;});

This Express method allows us to specify what is displayed on our webpage at the specified location (“/” is the root page).

The request parameter (req) lists a bunch of information on the user’s request

The response parameter (res) allows us to change what is displayed on our page

response.send(“HTMLCode”);

This sends HTML code to the webpage as a response. You can also send plain text.

response.write(“HTMLCode”);

cumulatively adds HTML lines to your webpage

response.end();

or can be substituted for response.send(); //empty parameter

response.sendFile(“filePath”)

Sends full files. You can send entire html files this way.

\_\_dirname

Returns the directory of the current file. 2 underscores!!

You can have multiple app.get methods with different route parameters to set a response to all of your different webpages.

You can use the nodemon npm to automatically refresh your servers

npm install -g nodemon

location it is saved does not matter

run server with nodemon using:

nodemon fileName.js

**How to access local images and css files:**

You can add local images and css files to your web page by using:

app.use(express.static("fileName"));

This allows you to access images and css files in the root folder fileName

**How to handle user input on server:**

Forms are used to contain user inputs (text, radio, submit)

<form action=”location” method=”post”> Input HTML code </form>

action=”/”

sends information to root page of server.

If no action attribute is specified, the form by default will send the info to the current webpage it is on.

The action attribute defines the location the data is submitted to.

The method attribute defines how the information is sent by the form

“get” sends data by the form although this data is visible in the page’s address field.

“post” sends the data by the form as well although the address field is not displayed.

After form data is submitted, your webpage must be able to handle a response.

app.post(“route”, function(req, res){code});

This is executed on submission of form data.

In order to utilize user data, you need to install the npm body-parser. This package parses incoming request bodies in a middleware available under the req.body property.

npm install body-parser

const bodyParser = require(“body-parser”);

app.use(bodyParser.parseType());

bodyParser.text()

parses code as plain text

bodyParser.json()

parses code as json file (similar to JS objects)

bodyParser.urlencoded({extended: true})

parses code from html form.

{extended: true} allows us to post nested objects.

req.body

creates the object body with properties and values from the form

Values are stored as strings. To convert to int, use Number(string);