NodeJS Notes

API = application program interface

GUI = graphical user interface

cls clears terminal

cd.. brings you back to root folder

cd folder/ brings you to wanted folder in root folder

exit removes entire terminal

node program.js runs the program

\*\*learn more about JSON\*\*

json lowercase can be equal to true/false

Setting json = true in an object w/external request as 1st arg auto parses the external info

npm install – in command line for any new nodejs/vb .exe

In command line, in each folder, npm init -y to initialize, -y answers yes for all defaults

npm install somePackage@latestVersion

var = require(‘./localfile/program’) allows program to use internal program when they program uses module.exports

var = require(‘express’) used to create server side script (client as well?)

calling express() doesn’t take any args ie. var1 = express(). var1 usually = app

var1.get(‘route to site after .com ie. /help’, ( req, res) => { do something for route ie. res.send(‘Hello’) } )

req.query looks for a user requested query ie. req.query.address ie. /help?address=cincinnati

var1.listen(port#) to setup web server. Can add 2nd arg like console just to verify running

var1.use() customizes the server you create (add’l?)

var1.set(‘view engine’, ‘hbs’) – after installing hbs (handlebars integrated w/express), can create dynamic templates. Handlebars expects to live in folder called ‘views’ w/in project folder. Create file with .hbs extension. Then, in app, use var1.get(‘/filename ’, (req, res) => { res.render(‘filename’, {object w/values} ) } ). Inject values into html (in .hbs file) ie <p>Created by {{value}}<p>

Handlebars (for dynamic rather than static web pages; use with express; see above)

var = require(‘path’) used to serve up folder paths to a file(s)

path.join(\_\_dirname, ‘../folder’) gives current folder, then up one, folder, then file(s) in folder

var = require(‘fs’) does some good stuff like .find

fs.writeFile(‘filename.ext’) self explanatory

fs.readFile(‘filename.ext’) self explanatory

var.filter(function(arg){return x}) filters for x

var = require(‘yarg’) allows inputs from the command line

But so does var = process.argv[2] if(var === ‘x’) {do something}

[0] = node, [1] = program to run, [2] = input after program

var = require(‘request’) allows program to get data from external ie. website

request({object}, function(error,response) object like url,

for url customizing, add ? to end and items wanted, ie. ?lang=en&units=us

var = function(){ try{} catch(x){return y}} tries some code; if not, uses catch

Not sure why this is better than if/else

var.push(info) inbed into function, pushes info into parent function/array

.parse() cuts everything except string value

var = function(x) {x.forEach((y) => {doSomething} ) performs a function on each element in an array(x)

module.exports = { name: var} makes current file available to other files

setTimeout(()=>, #mlsecs) allows to run code after secs, but everything else will run first

Node doesn’t like url2 as a variable

encodeURIComponent(input) use when adding input to a url so special chars become browser readable

A ­callback is an argument that does something in ­response to data retrieved/not ie. from url

var = (arg, callback) => { const url = ‘website’

request({url, json: true}, (error, response.body) => {

if (error) { callback (‘Can’t connect’, undef )}

else { callback(undef, url info) } } )

fetch(‘url’) gets whatever is in the url (THIS IS CLIENT SIDE)

this is also get(‘url’) ???

.then((response) => { doSomething } ) does something after the fetch

var = document.querySelector(‘htmlTag ie. form’) this will select the htmlTag ie. form (CLIENT)

var2 = document.querySelector(‘input’) this will grab whatever user input to form

var.addEventListener(‘event ie. submit’, (e) => { doSomething } ) do something when form used

e = event. e.preventDefault() stops page from default refreshing so user input still visible to user

var3 = var2.value extracts input value

var4 = d.qS(‘tag’) var4.textContent will put content on page

in html file, <script src=”/folder/js.js”></script> needs to be placed after all input