**What I learned making this app**

Goal – get the day of the week and create a unique response based on the date

Res.send vs. res.write –

Res.send is the last message the browser reads, whereas res.write allows you to add multiple lines. Res.write is like writing an email, while res.send is like typing in a messenger app.

Res.sendfile – sends entire html file. (\_\_dirname + ‘/index.html’)

Date() object – native Javascript object. Prototype.getDay() method, returns date as 0-6.

Templating – can change certain parts of HTML depending on logic in server.

Using regular HTML, if you wanted to have a different response for each day of the week, you’d need seven HTML documents. EJS templates allows you to “inject” variable into HTML so that you can update the content from your app.js file.

**Setting up EJS file**

EJS – Embedded Javascript Templating - <https://ejs.co/>

Specifically using EJS with Express page - <https://github.com/mde/ejs/wiki/Using-EJS-with-Express>

Tell app to use EJS:

App.set(‘view engine’, ‘ejs’)

Res.render – uses view engine to “render” particular page, such as index.ejs page. Assumes a views directory containing index.ejs file. Essentially writing html code

To pass variable from app to ejs = res.render(‘index’, {foo: ‘FOO’}); key/value pair. Passes foo variable and inserts into where foo is found in ejs. Add **marker** (<%= %>) in ejs to tell file where to place variable.

Key = variable in EJS file

Value = variable declared in app.js file

(could be same name, i.e. day: day)

**Running code inside HTML file**

<% %> scriplet tag – inject Javascript control flow into HTML (if, switch, while)

-Add around any part of code that is not HTML (don’t forget to add around closing bracket).

-Work on line-by-line basis – have to add open and close scriplet tag for every single line that contains JS

In most cases, you want to keep your logic inside your server – only use if it’s much quicker to do inside template (control flow)

**Passing Data from Webpage to Server**

-Javascript date object:

toLocaleDateString() method – format JS using parameters

var options = { weekday: ‘long’, year: ‘numeric’, month: ‘long’, day: ‘numeric’ }

returns weekday, day, month, and year, don’t have to add switch statement to choose dates from 0-6

create unordered list with to do items.

To enter new items to list, need to pass data entered on webpage form to server via a POST request (send data to server)

Add form with text input and button which POSTS user input to home (/) route

Naming input as “newItem” allows you to access it through body-parser.

App.post(‘/’, function(req, res) {

Var item = req.body.newItem

})

Routing:

When submit button pressed, form makes a POST request to home route and posts value of text input with “newItem” name.

When request is received, it is “caught” in app.post, tap into request looking through body of post request for value “newItem

Pass data back to web page into list item

-add newListItem variable to app.get / res.render

-res.redirect in app.post – when a POST is triggered, value of item saved, redirect to home route which triggers app.get

Create array items – every time new post request, append new item to array. This way variable can be rendered on app.get and updated for each POST request

<% for (let i = 0; i < newListItems.length; i++) { %>

<li> <%= newListItems[i] %> </li>

<% } %>

Iterate through each item in newListItems array and for reach create a new list element using index number

Diff. between var and let –

In conditional loops, var is global, but let is local

**Styling ToDoList**

-put css file in css folder

-link to css from index.ejs, but nothing shows up. Have to be served by server

Node only serves up app.js (access point) and views folder

Create **public** folder with css, js, etc.

Tell express to serve up public folder as static resource

-webkit-linear-gradient (65deg, purple 50%, white 50%)

Creates gradient at 65 degrees with half per color

**Pushing to git**

-make repository on github

-git init -b main in command line

Add node modules to git ignore – don’t need to add specific library files because package.json defines dependencies, package.lock defines versions

Git add . – adds current directory, all files

Git commit -m “message”

Need to push branch

Git remote add origin {URL} (get from “cloning” repository)

Git push origin mains

**Templating vs. Layouts**

-can change route by adding app.get(‘/work’) and app.post(‘/work’) for example.

-Issue – res.redirect(‘/work’) still goes to home route, because form has “/” listed as action

-change value of button to <%= listTitle%> so that it can respond dynamically to route

-can tap into list (name of button) button using req.body

-add if/else statement to app.post to route based on value of submit button

How to add separate pages with similar styles?

EJS allows for creation of “layouts” / partials, inject same header or footer into every page while changing main content. Recycle portions of HTML

Code to include another ejs file:

<%- include("header") -%>

Create separate partials folder (make sure to update file path: patials/header

Makes it easy to create large website with many different pages

**Node Modules = Passing Functions and Data between files**

-modules contain export.()

Create module of own from scratch – function to generate current day

Functions don’t really belong in app.js – save this file for routing and requiring

Separate get date function to new .js file

Node has a module object, which references current module.

Since module is installed locally rather than using npm, to acces it, have to add \_\_dirname +

Parent module – module which launched current module

Module.exports = getDate

Export getDate function

NB don’t add parentheses or you will call function

Add function to app.get = let day = date()

If you want module to do more than one thing:

Instead of binding entire module object to date function, can do module.exports.getDate = getDate, then module.exports.{whatever} = {whatever}

Call in app.get with date.getDate (or date.getDay if you only want day of week)

Exports shortcut =

Can use exports variable to refer to module.exports

Export.getDate = function() {

**TO ADD –**

**Urgent (background color), Normal, Unimportant (opacity)**

**-on button click = dropdown**

1. Change button to trigger dropdown absolute element (don’t POST)
2. Three elements which each have their own value which will POST and add class to most recent item (last-child)
3. Pass variable class (set as ‘ ‘)

Old version:

<%- include("partials/header") -%>

<body>

<!-- display current day of week -->

<div id="heading" class="box">

<h1> <%= listTitle %> </h1>

</div>

<div class="box">

<!-- To-Do List items -->

<% for (let i = 0; i < newListItems.length; i++) { %>

<div class="item">

<input type="checkbox">

<p class="item"> <%= newListItems[i] %> </p>

</div>

<% } %>

<!-- / = home route -->

<form class="item" action="/" method="post">

<input type="text" name="newItem" placeholder="New Task" autocomplete="off">

<button type="submit" name="list" value=<%= listTitle %>>+</button>

</form>

</div>

<%- include('partials/footer') -%>

**WHAT I ADDED**

Add dropdown button (script tag in footer) absolutely positioned

Depending on button clicked, add class name to array that coincides with list items