**Ch. 22 – EJS (Embedded JS templating):**

EJS allows us to update an HTML file we plan to send to our server based on given inputs. For example, we can display the day of the week before sending out HTML file instead of having to create 7 different HTML files.

**How to add values to your EJS file which replaces your HTML file:**

With EJS, you must have a folder called ‘views’ in your directory containing the ejs file you want to change. The EJS can be called whatever you want (list.ejs for example), although the extension will be .ejs and not .html.

After npm installing EJS, you must use EJS in your root JS file to implement it.

app.set("view engine", "ejs");

<h1>It's a <%= theDay %>!</h1>

Use <%= variableName %> to set up which variables you want changed in your EJS file

res.render('list', {theDay: currentDayString});

This updates the ejs file with the value currentDayString for the variable theDay

**How to add JS to your HTML file:**

<% basic JS code here %>

Allows us to run basic JS code within the <% %> in your HTML file

Must include bracket and % on every line there’s JS code.

You cannot include any HTML in the brackets.

**How to reuse HTML code:**

In your ‘views’ folder, create a file containing the chunk of html code you want to reuse (ex: header.ejs). Write all of the html code you want inside of this file. In the html file you are rendering, include:

<%- include(“fileName”) -%>

The ejs extension should not be included.

You can use this to add headers and footers to keep your style between the webpages you render constant. This reduces the amount of reused code you must write.

**How to reuse JS code:**

In another JS file, write the code you wish to reuse (for example lets say we define 2 functions: getDate() and getDay()). Using module.exports (or just exports), we can now reuse the JS code in this file.

exports.getDate = function(){code goes here; return value;}

exports.getDay = function(){code goes here; return value;}

In the JS file we wish to reuse this code, you must first require this local JS file.

const JSobjName = require(\_\_dirname + “/JSfileName.js”);

const object = JSobjName.getDate();

**Scope** – The region of a computer program where the binding is valid.

A local variable (created in a function) will only be valid in that same function (it has local scope) and cannot be accessed outside of that function or in another function.

You can create var variables in for/while/if/else statements and they will still be accessible in the function that those initial statements are a part of.

Global variables are created on the outermost layer of a program (outside of any functions) and are accessible anywhere in the program, including any functions.

const values cannot be changes

var and let values can be changes

all 3 declared in a function are local variables.

All 3 declared outside of a function are global variables

var declared in a code block is global up to its direct parent (if code block is in a function, its global to its function, not globally)

\*\*\*Regarding const, you can still push values into arrays and change the values of object keys, although you cannot assign new arrays or new objects to constants.