TEMPLATING DATABASE DRIVEN APPLICATIONS

EJS

AIM

The aim of this lab is to introduce you to the process of creating dynamic pages through templates.

The first section of this lab will walk you through the process of setting up a template, using partials and inserting data into a template.

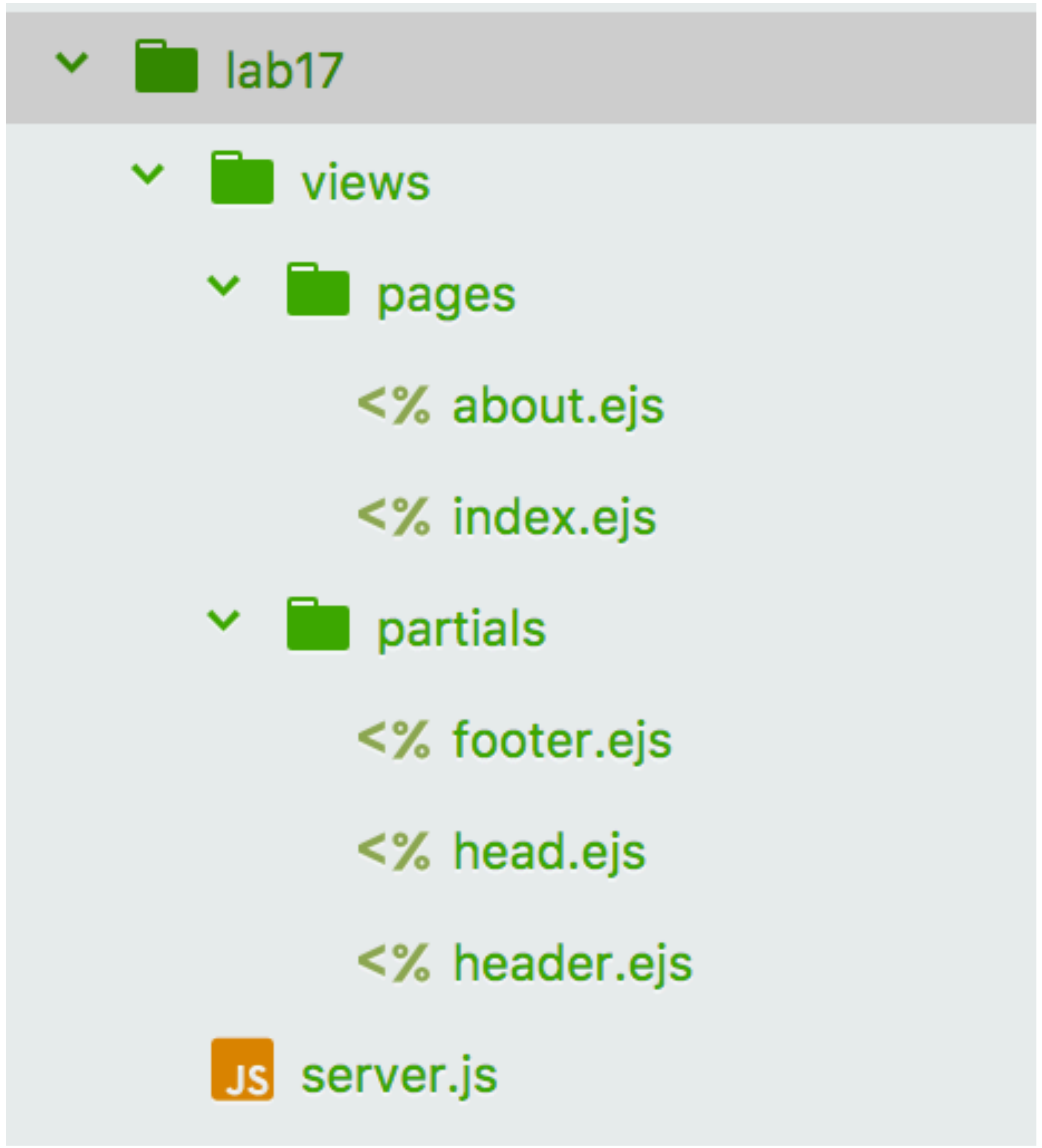
In the second section of the lab you will adapt your Star Wars Quotes app from last week to use templates for its pages.

SETTING UP THE PROJECT

PROJECT STRUCTURE

PROJECT STRUCTURE

- When using Express with page templates, Express will alway look for the templates in a folder called **views.**
- It is usually best practice to structure your folders so that it makes sense to anyone else who wants to look at your code.
- In your **lab_14 folder**
- **Create the files opposite replicating the folder structure** (don't put any code in them yet)



```
lab17
  views
    pages
      <% about.ejs
      <% index.ejs
    partials
      <% footer.ejs
      <% head.ejs
      <% header.ejs
  server.js
```

server.js should be at the top level

the views folder should have a partials folder and pages folder

CREATING PARTIAL PAGES

PARTIALS

HEAD.EJS

- head.ejs is going to be our partial page that contains all the head information that all our pages are going to use.
- This is where we would put stylesheets, any links to client side javascript, links to things like jquery
- the head HTML below is pretty simple, really just importing bootstrap and setting the page title

```
<!-- views/partials/head.ejs -->
<meta charset="UTF-8">
<title>Super Awesome</title>

<!-- CSS (load bootstrap from a CDN) -->
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/bootstrap.min.css">
<style>
    body    { padding-top:50px; }
</style>
```

HEADER.EJS

- header.ejs is going to be our partial page that contains our navigation system that we would like to use on all our pages.

```
<!-- views/partials/header.ejs -->

<nav class="navbar navbar-default" role="navigation">
<div class="container-fluid">

    <div class="navbar-header">
        <a class="navbar-brand" href="#">
            <span class="glyphicon glyphicon glyphicon-tree-deciduous"></span>
            EJS Is Fun
        </a>

        <ul class="nav navbar-nav">
            <li><a href="/">Home</a></li>
            <li><a href="/about">About</a></li>
        </ul>
    </div>
</div>
</nav>
```

these are all bootstrap classes

links in our navigation

FOOTER.EJS

- footer.ejs is going to be our partial page that contains our page footer.
- Usually this is simply a copyright or some other general information.
- The code for this page is below.

```
<!-- views/partials/footer.ejs -->
<p class="text-center text-muted">© Copyright 2018 Some People</p>
```

PARTIALS

- We are not going to use these partial pages directly , but are going to import them into any page which we want to use them.
- This means that all our pages will have the same style, navigation and footer….even though we have only written them once

CREATING PAGE TEMPLATES

PAGES

INDEX.EJS

- index.ejs is going to be our main page.
- We are going to import the partials we have just created into it.
- Remember from the lecture anywhere you want to use a partial you just use the **<%- include('path-to-partial') %>** command.
- Our partials are stored in our partials folder so our path would be something like **../partials/head**
- because our index file is in the views folder we need to use the .. in the path to go up a directory
- You can see opposite where the head, header and footer are imported into the correct positions in the page.

```
<!-- views/pages/index.ejs -->

<!DOCTYPE html>
<html lang="en">
<head>
    <%- include('../partials/head') %>
</head>
<body class="container">

<header>
    <%- include('../partials/header') %>
</header>

<main>
    <div class="jumbotron">
        <h1>This is great</h1>
        <p>Welcome to templating using EJS</p>
    </div>
</main>

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

</body>
</html>
```

ABOUT.EJS

- about.ejs is just another page that is going to demonstrate how multiple pages can use the partials.
- Again we are simply going to import the partials we have just created into it.
- But this time I have included some different bootstrap classes, just to highlight that the CSS styles really are imported.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <%- include('../partials/head') %>
</head>
<body class="container">

<header>
    <%- include('../partials/header') %>
</header>

<main>
<div class="row">
    <div class="col-sm-8">
        <div class="jumbotron">
            <h1>This is great</h1>
            <p>Welcome to templating using EJS</p>
        </div>
    </div>

    <div class="col-sm-4">
        <div class="well">
            <h3>Look I'm A Sidebar!</h3>
        </div>
    </div>

</div>
</main>

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

</body>
</html>
```

CREATING THE SERVER

SERVER.JS

SERVER.JS

- None of what we have done yet will really do anything.
- A browser doesn’t know how to render an EJS file until it has been processed by our server.
- Our server is very simple and just contains two routes, one for "/" (the index) and one for "about"
- The main difference is the 'view engine' setting. This tells Express what templating framework we are using and how do deal with the EJS files
- when we want to render a template (ejs file) we just call **res.render('path-to-template')** all our templates are in the pages folder, so we use that in the path.

```
// server.js
// load the things we need
var express = require('express');
var app = express();

// set the view engine to ejs
app.set('view engine', 'ejs');

// use res.render to load up an ejs view file

// index page
app.get('/', function(req, res) {
    res.render('pages/index');
});

// about page
app.get('/about', function(req, res) {
    res.render('pages/about');
});

app.listen(8080);
console.log('8080 is the magic port');
```

TESTING SO FAR

- Save everything you have done so far
- Commit and push up to your codio box. (remember to pull down the code onto codio)
- now in a terminal window **enter your lab_14 directory** (something like `cd CM2104DWD/lab_14` depending on how you have named your directories)
- run the commands opposite to install and set up our server project.

```
npm init
```

initialise the project, the default values should be fine. if you get an update warning you can ignore it.

```
npm install express
```

installs express!!

```
npm install ejs
```

installs ejs!!

Note if you do want to update npm you can do so by typing

```
sudo npm i -g npm
```

STARTING THE SERVER

- start your server by typing **`npm start`**
- Express acts as it own server running on port 8080 so we must use the correct url to access the site navigating through the codio static view will not work!
- **try visiting http://your-site-8080.codio.io**
- **and http://your-site-8080.codio.io/about**
- remember to substitute in your own site name, you should see that the two pages are using the same styles and the navigation works on both.

INSERTING DATA INTO A TEMPLATE

<%= %>

INSERTING DATA

- Ok now we are going to see how we can insert server side data into a page.
- Comment out your current '/' route in server.js and replace it with this new one.
- the code here create a simple array called **DRINKS**
- and a string called **TAGLINE**
- these values are then passed into the template

current '/' route

```
// // index page
// app.get('/', function(req, res) {
//     res.render('pages/index');
// });
```

new '/' route

```
// index page
app.get('/', function(req, res) {
    var drinks = [
        { name: 'Bloody Mary', drunkness: 3 },
        { name: 'Martini', drunkness: 5 },
        { name: 'Scotch', drunkness: 10 }
    ];
    var tagline = "Any code of your own that
you haven't looked at for six or more months
might as well have been written by someone
else.";

    res.render('pages/index', {
        drinks: drinks,
        tagline: tagline
    });
});
```

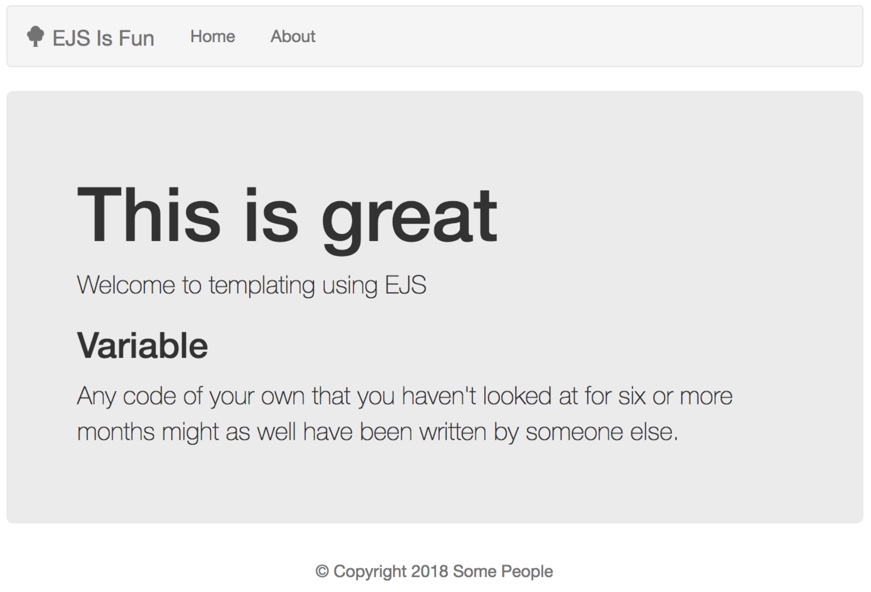
these values are passed to the template

USING THE VALUES

- To use the values in the template we have to use the **<% %>** tags
- Add the code opposite into your index.ejs, just after the `<p>welcome to templating using ejs</p>` line is fine.
- We are using the <%= tag here as we want the value to be written to the final page
- Save and commit all your code, push to github and pull on codio
- restart your server, (npm start) and see what happens.

```
<h2>Variable</h2>

<p><%= tagline %></p>
```

the tags define that we are asking for values based to the template from express



This is great

Welcome to templating using EJS

Variable

Any code of your own that you haven't looked at for six or more months might as well have been written by someone else.

© Copyright 2018 Some People

USING THE MORE COMPLEX VALUES

- the drinks value we passed through was an array, we can loop through this in the template, very useful for things like displaying lists.
- Note the use of some different tags for flow control. items within the <% tags are not added to the page but just used to control the loop.

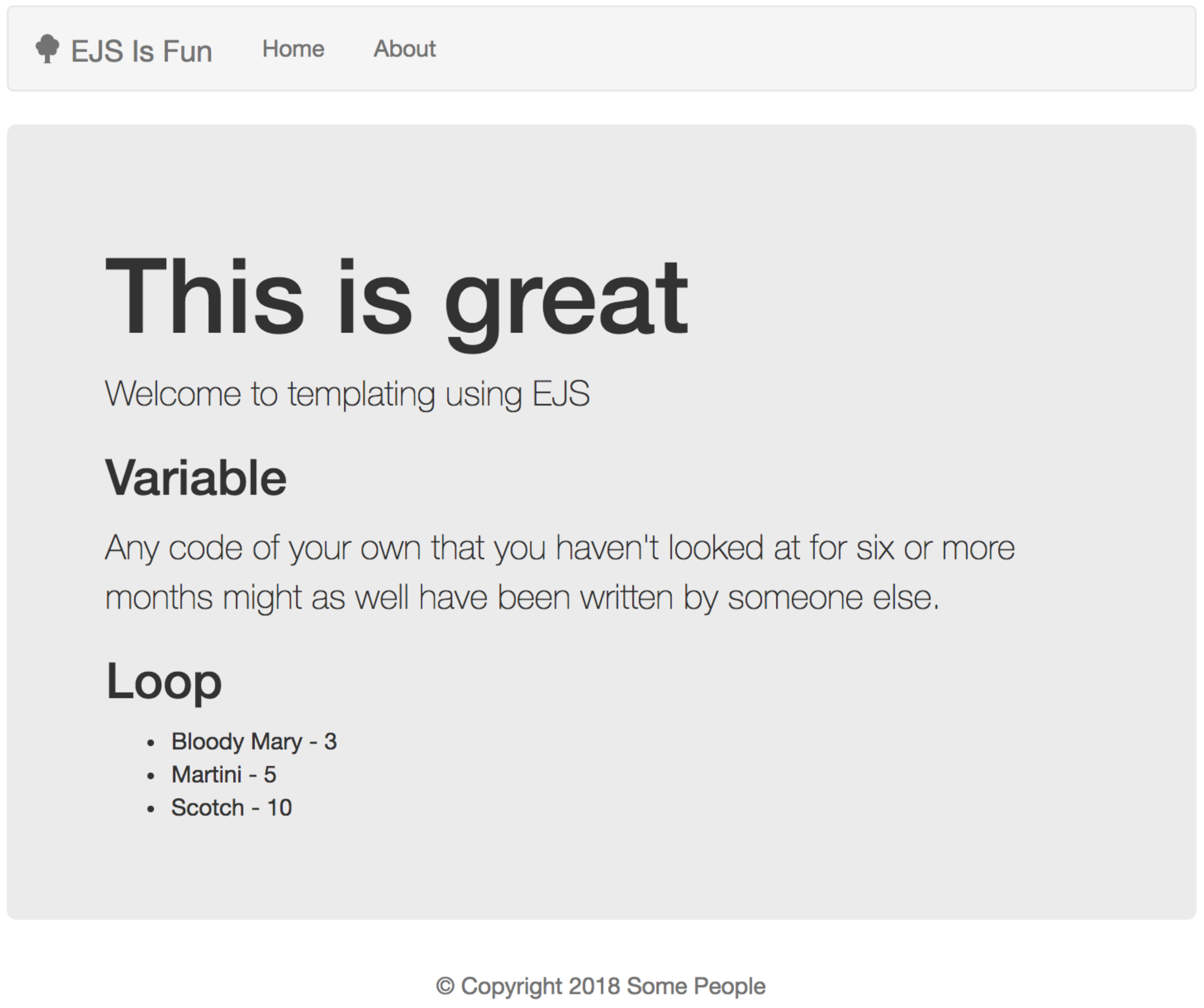
```
<h2>Loop</h2>

     <ul>
       <% drinks.forEach(function(drink) { %>
         <li>
           <%= drink.name %> -
           <%= drink.drunkness %>
         </li>
       <% }); %>
     </ul>
```

control the loop,
not drawn on page

values from array,
drawn on page

TESTING

- Again save, commit and push your code up to github/codio
- after restarting your server, you should see something like this



This is great

Welcome to templating using EJS

Variable

Any code of your own that you haven't looked at for six or more months might as well have been written by someone else.

Loop

- Bloody Mary - 3
- Martini - 5
- Scotch - 10

© Copyright 2018 Some People

STAR WARS QUOTES

YOUR TURN

STAR WARS QUOTES

- Last week you should have developed a simple web app that displayed star wars quotes and allowed you to enter, delete and update quotes.
- The final output was very bland and just wrote the lines to the screen. The navigation was pretty poor as well.
- I have provided a version of this app on moodle, it doesn’t work yet as there are a few tasks steps you need to do.

SETTING UP

- Download the starwarsquotes zip file from moodle.
- Add these files to your visual studio code project in a directory called lab14-quotes
- Save, Commit and push as normal and then pull the files down to your server.
- you will need to run

```
npm init
npm install express
npm install mongodb@2.2.33
npm install ejs
```

- now complete the steps outlined on the next few pages to complete the app

TASK - CONSISTENT STYLING

- Your task is to use templates to make the Star Wars Quotes app much better.
- You will see I have made some changes to the code. The index page is now much more pretty, but the other pages have no styling (you can test these locally to see what they look like).
- Use the same approach that you have just seen to perform the steps opposite to create a nicely styled app that uses as little code as possible.

Step 1: Rename all your the **.HTML** files to **.EJS**

Step 2: Look at the index.ejs file, what parts of that file do you want to replicate in the other files? Look at what you did in the previous example in the lab. What would go in a HEAD, HEADER and FOOTER partial?

Step 3:Edit the other EJS files so that the styles are consistent using the importing your new partial files just as you did in the previous example.

Step 4: Edit the server.js file to complete the get routes at the top (I have completed the first for you)

you can test your server now, you should see that the styles are consistent!

TASK: USING DATA

- Now that you have the styling consistent you need to find a way of directing the quotes to the index.ejs page
- Again using the previous example you should be able to work out how to do this

Step 1: the code in the allquotes route currently gets all the quotes from the db. Use the code in here to work out what code you need in your '/' route to get an array of all the quotes.

Step 2: Pass your array to the index.ejs file using the render function, hint: **res.render('index.ejs', {quotes:result})**

Step 3:In your index.ejs, you should be able to use the code shown for the drinks example to loop through and create the list of quotes.

Remember when you want to use the data that has been passed through you need to your the <% tags.

Step 4: Some of the other pages (filter for example) also shows a list of quotes, can you complete the functionality here as well?

FINAL THOUGHTS

WHAT HAVE WE DONE?

WHAT HAVE WE DONE?

- In this lab you have seen how EJS can be used to create templates for dynamic pages.
- We have used EJS because it is essentially more javascript, so you don’t need to learn another language.
- You now know how to share settings, styles, navigation etc across multiple pages in a site with no code repetition
- In week 7 I will show you how to use this technique for a user profile system.