WEB422 Assignment 2

Submission Deadline:

Friday, May 31 at 11:59pm

Assessment Weight:

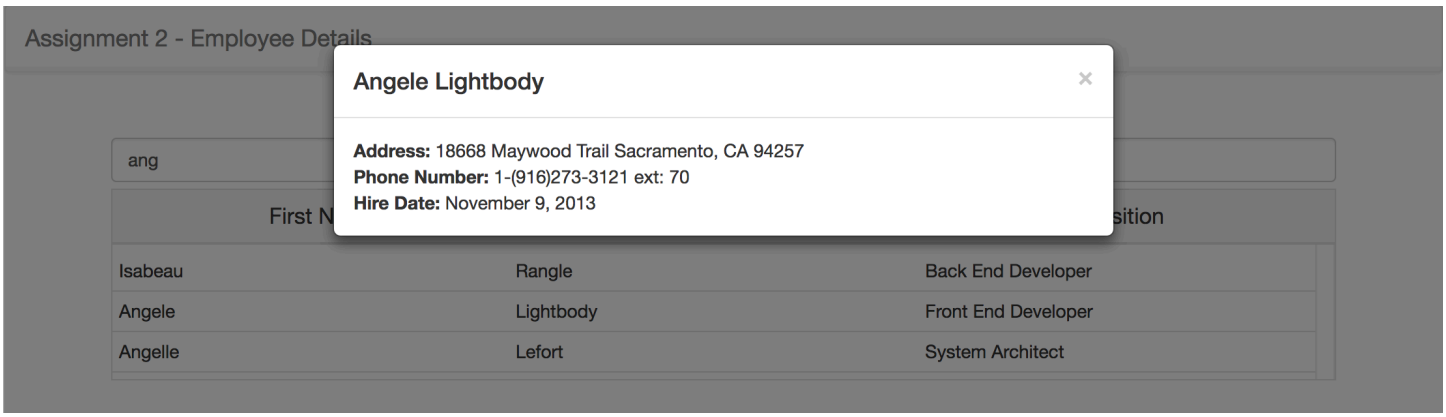
9% of your final course Grade

Objective:

To work with our Teams API on the client-side to produce a rich user interface for accessing data. We will practice using well-known libraries including Lodash, Moment, jQuery, and Bootstrap, and practice working with ES Modules and bundling.

Specification:

For this assignment, we will be focusing specifically on **Employees** of the system. We will create a single, **searchable table** that shows a subset of the employee data (ie: columns: **First Name**, **Last Name** & **Position**). When the user clicks on a specific employee (row) in the table, they will be shown a modal dialog window that contains additional detail about the employee. We will be making use of the Bootstrap framework to render our HTML, jQuery to work with the DOM and Lodash / Moment.js to format the data.



The src/ Directory

The assignment2.zip file contains a number of files and folders you will need. Please read the README.md for instructions on installing dependencies, and running scripts.

After you have run `npm install`, and installed all the necessary dependencies, you'll be able to build your code and start a web server using `npm start` or `npm run parcel` (they do the same thing). This will build your code in dist/ and start a web server on http://localhost:1234/ Try it!

If you change your code while the server is running, once you save a file, it will rebuild the code and refresh your browser. Try changing something in src/index.html or src/index.js.

Dependencies: jQuery, Bootstrap, Moment.js, and Lodash:

We're going to use a number of dependencies, and instead of putting all their CSS and JS files in our HTML, we'll create a bundle that "imports" them all as ES6 Modules.

Before we can do that we need to install a few more. I've already installed Bootstrap and jQuery for you, and set them up. Take a look at the following files:

- package.json: see the "dependencies" section of the file
- src/jquery-es.js: This will "import" the jQuery library as an ES Module, and then add it to the browser's window Object. Bootstrap expects to find it there.
- src/index.js: I've already taken care of import jQuery and Bootstrap. You can use $ anywhere in index.js, and it will already have jQuery loaded.

Now it's your turn. Let's install two more dependencies, and import them into our code: moment.js and lodash

To install these, we use `npm install`:

```
npm install --save lodash-es moment
```

By including the --save flag, we're telling npm to also update our package.json file with two new dependencies. That is, we want to install these packages, but also record that we need them in our package.json dependencies list.

After you run this command, take a look at your package.json file and make sure the "dependencies" section includes both package names.

Importing Moment and Lodash Modules:

After installing these packages to your node_modules/ folder, you can now **import** them in your code. Change to your src/index.js file, and just below line 6, you can import the modules like this:

```
// Place your imports for Moment.js and Lodash here...
import moment from 'moment';
import _ from 'lodash';
```

Now you can use the moment and _ Objects to access these modules. Another option for importing lodash, is to only import the specific functions you need, instead of the entire Object. This reduces overall code size, and improves website load times:

```
import { escape, uniq, chunk } from 'lodash';
```

In this case, only 3 functions have been imported from lodash, and we can use them by the same name we imported. For example:

```
const uniqueNumbers = uniq([1, 1, 1, 2, 3, 3]);
```

This is only an example, and you can import any functions you want, or include the entire module if you're not sure what you'll need.

Creating the Static HTML:

The first step in developing the solution is to create some Static HTML as a framework for our dynamic content.

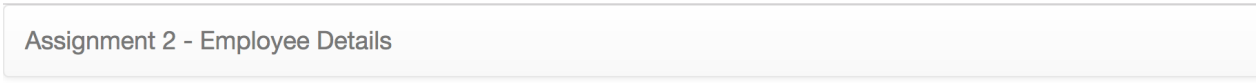
You may use your "index.html" file from Assignment 1 as a starting point, and place all your HTML in src/index.html:

Navbar

Assignment 2 will use a similar navbar, only without the dropdown menu on the right-hand side.

- To fully remove the dropdown, we will **remove** the "navbar-toggle-collapsed" button, all elements with the class "icon-bar" and the element with the class "navbar-right"
- Change your "navbar-brand" link to read: **Assignment 2 - Employee Details**

When completed, your navbar should look like the following (no right menu):

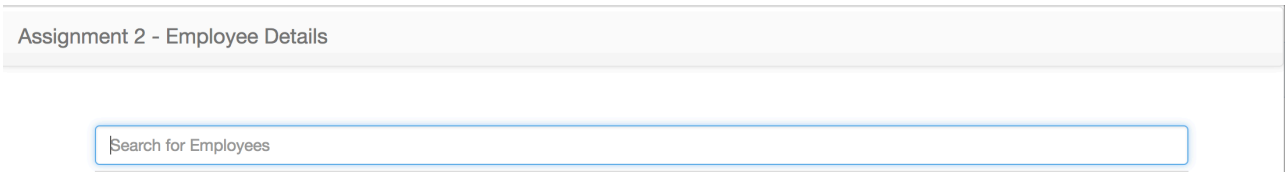


Search Field

Where you previously had your "well" in Assignment 1, we will instead place a "search field" that will take up the full width of the "col-md-12" container. This can be accomplished by creating an **input** element of type "**text**" and giving it the class "form-control", your text field must have the following attributes:

- "placeholder" - the value of the "placeholder" should read "Search for Employees"
- "id" - the search field must have a unique "id", ie: "employee-search" so that we can correctly watch it for events such as "keyup"

When complete, your page so far should look like:



Fixed-Header Table

Since we will be dealing with all Employees (300) on this page in a table, we must allow our table rows to **scroll vertically**, while the headers **remain in place.** This will make it easier for users to keep track of the data as they scroll through the list of Employees.

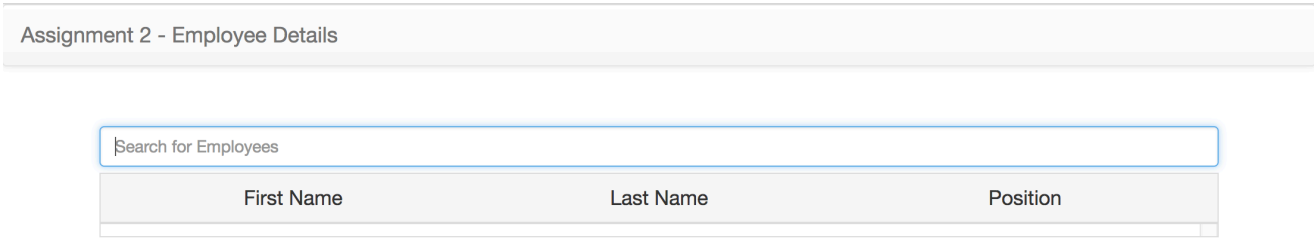
This can be accomplished in many ways, however, since we're using the **Bootstrap Grid System** we can leverage the responsive column classes to create a table-like structure that will behave in the required manner (ie, fixed header & scrolling body).

In the **Code Examples** directory, you can find sample .html & .css (also the .scss source) to create this type of structure under the "bootstrap-header-table" folder. You can find the specific folder with the files online here.

The "bootstrap-header-table" CSS is already included in index.html and src/styles. To use it:

- Include the 3-column example html code (from bootstrap-header-table.html) in a new "row" **beneath the Search Field**
- Change the 3 **header-column** elements to read: **First Name**, **Last Name**, and **Position** respectfully
- Remove the element with class "**body-row**" and all of it's contents - we will be creating a new **"body-row"** element for every Employee in our database
- Give the element with the class **"body-rows"** a unique **id**, ie: "**employees-table**". We will need to reference this element when we wish to **append** new "**body-row**" elements

When complete, your page so far should look like:



"Generic" Modal Window Container

We will be showing all of our detailed Employee information in a Bootstrap modal window. Since every time we show the modal window, it will have different content (Specific to the Employee that was clicked), we must add an empty, **generic** modal window to the bottom of our page.

To get the correct HTML to use for your Bootstrap modal window, use the following example from the documentation as a starting point.

Once you have copied and pasted the "modal" html into the bottom of your index.html page (ie, before all of your <script></script> tags), make the following changes:

- Give your with the class "**modal fade**" a unique **id**, ie: "**genericModal**". We will need to reference this element every time we wish to show a **modal window**
- Remove the "Modal Title" text from the <h4> element with class "**modal-title**". We will be using jQuery to populate this with the selected Employee's **First & Last Names**
- Remove the <p> element with the text "One fine body..." from the element with class "**modal-body**". We will be using jQuery to populate this with the selected Employee's **Address / Telephone details** as well as their **Hire Date**
- Finally, remove the element with the class "**modal-footer**". We will be using our modal window simply for displaying data, so we will not need the user to perform any additional actions via button elements in the footer.

JavaScript File (src/index.js):

Now that we have all of our static HTML / CSS in place, we can start dynamically adding content and responding to user events using JavaScript. In your src/index.js file add the following variable / functions and document ready (ie: **$(function(){ … });**) code

Variable: employeesModel

Assign a value of an empty array, ie [] to a variable called "employeesModel" at the top of your file using "let". This will be the "view model" for our current "Employees" view (global to this file). A number of the functions declared within index.js will reference this array.

Function: initializeEmployeesModel()

This function will populate the "employeesModel" array, by issuing an AJAX call to your Teams API **hosted on Heroku** and making a **GET** request for **/employees**. Once the AJAX call completes successfully, perform the following actions:

- Assign the results to the "employeesModel" variable, causing it to be populated with all 300 Employees returned from your API
- Invoke the "refreshEmployeeRows" function (see below for specification) with the employeesModel as the parameter, ie "refreshEmployeeRows(employeesModel);"

If the AJAX call fails, perform the following action:

- Invoke the "showGenericModal" function (see below for specification) with "Error" as the "title" parameter and an appropriate error message as the "message" parameter, ie: "showGenericModal('Error', 'Unable to get Employees');"

Function: showGenericModal(title,message)

When invoked, this function will perform the following actions: on your generic modal, ie: "**genericModal**"

- Set the content of the "modal-title" to whatever html was passed in the "title" parameter
- Set the content of the "modal-body" to whatever html was passed in the "message" parameter
- Lastly, programmatically show your generic modal using it's **id**, ie: "**genericModal**"

Function: refreshEmployeeRows(employees)

When invoked, this function will perform the following actions, pertaining to the "**employees**" parameter:

- Defines a Lodash template using "escape" and "evaluate" delimiters to produce the following html structure for **every employee** in the local "**employees**" array. **NOTE:** The ***[…]*** represents a placeholder for a specific value:

```
<div class="row body-row" data-id="[the employee's _id]">
        <div class="col-xs-4 body-column">[the employee's first name]</div>
        <div class="col-xs-4 body-column">[the employee's last name]</div>
        <div class="col-xs-4 body-column">[the employee's position name]</div>
</div>
```

- Next, Invoke the template function and provide the "**employees**" array as part of the parameter.
- Add the results from invoking the template function (this should be a string containing 300 "body-row" elements - one per employee) as a child of the "**employees-table**" element.
- **NOTE:** It is vital that the "**employees-table**" is **cleared** of any existing "body-row" elements, before adding any new ones.

Function: getFilteredEmployeesModel(filterString)

When invoked, this function will perform the following actions, pertaining to the global "**employeesModel**" array:

- Returns a filtered version of the "**employeesModel**" array using the following rule:
    - Any employee object in the "**employeesModel**" array whose **.FirstName**, **.LastName**, or **.Position.PositionName** properties contain the local "filterString" (provided to the function) will be added to

the filtered array. This will allow the user to filter all 3 columns of the table with a single string.

- **NOTE:** This operation is **not case sensitive**

- **Hint:** The Lodash **_.filter()** method is perfect for this type of operation

Function: getEmployeeModelById(id)

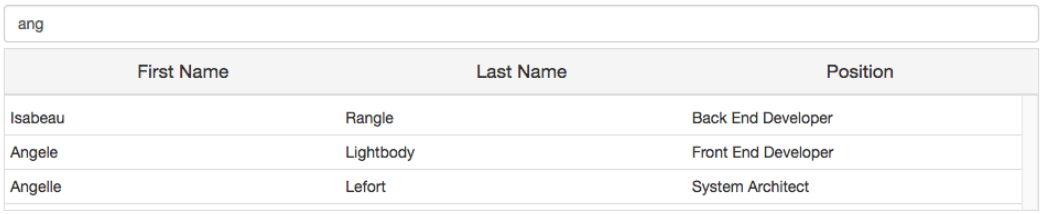
When invoked, this function will **search** the global "**employeesModel**" array for an **Employee** whose **_id** matches the local "id" (provided to the function).

- If the employee is found, a **deep copy** of the employee object is returned
- If the employee is not found, **null** is returned

jQuery DOM "ready" function: $(function() { ... });

When the DOM is ready, we need to perform some initial tasks including, **wiring up events** and **populating the page with data**. This includes:

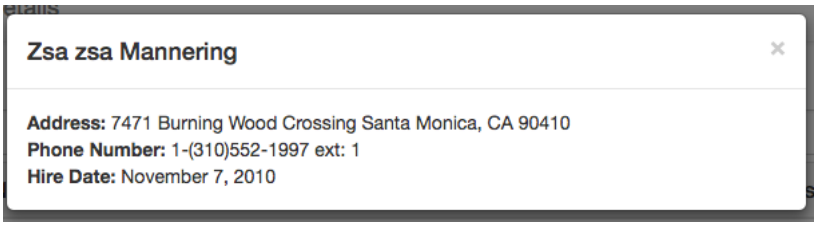
- Invoking the **initializeEmployeesModel()** function to fetch the data and populate our employees table
- Wiring up the "**keyup**" event for the Search Field ("**employee-search**") that performs the following actions when triggered:
  - Invokes the getFilteredEmployeesModel() function with the current value of the "Search Field" (this will return a filtered array using the "Search Field" string)
  - Next, invokes the refreshEmployeeRows() function with the filtered array - this will show only "filtered" rows to the user, ie:



- Wiring up the "**click**" event for every single (**current and future**) element with the class "**body-row**" that performs the following actions when triggered:
  - Gets a **copy** of the **clicked employee** by invoking the **getEmployeeModelById** function with the value of the "data-id" attribute of the clicked element (recall, every "body-row" element has a "data-id" attribute that stores that particular employee's _id value)
  - Convert the **HireDate** property of the **clicked employee** (returned from getEmployeeModelById) into a more readable format using Moment.js (**for example**: "November 7, 2010")
  - Define a Lodash template using "escape" delimiters to produce the following html structure for a **clicked employee**. **NOTE:** The ***[...]*** represents a placeholder for a specific value:

    Address: ***[all "Address" fields for the employee]***
    Phone Number: ***[the phone number and extension for the employee]***
    Hire Date: ***[The hire date formatted in a readable way]***

- Invoke the template function and provide the **clicked employee** as part of the parameter. Be sure to store the **result** (html string) for the next step.
- Invoke the **showGenericModal** function using the clicked employee's **First** & **Last Name** for the "**title**" parameter and the **result** from **invoking the template function**, for the "**message**" parameter. This will show the following modal window to the user (for Zsa zsa Mannering, for example):



Other (Optional) Improvements:

Once you have the basics working, here are some other optional things you could try to do, which would improve this code and UI:

- Use some glyphs (icons) in your UI to make it more interesting, see https://getbootstrap.com/docs/3.3/components/#glyphicons. For example, in your modal dialog for address, phone, etc.
- Split your index.js file into multiple files/modules, and use import/export to include things in your index.js. For example, split the code that works on your employees model array into its own module, and have your UI code go in another module.
- Add sorting to your table, so that users can click on a column and sort all the employees by any of the columns (e.g., last name)

Assignment Submission:

- Add the following declaration at the top of your main.js file

```
/*********************************************************************************
* WEB422 – Assignment 2
* I declare that this assignment is my own work in accordance with Seneca Academic Policy.
* No part of this assignment has been copied manually or electronically from any other source
* (including web sites) or distributed to other students.
*
* Name: _____ Student ID: _____ Date: _____
*
*
*********************************************************************************/
```

- Run the `npm run prepare-submission` script and upload the resulting **assignment2-submission.zip** to Blackboard.

Important Note:

- **NO LATE SUBMISSIONS** for assignments. Late assignment submissions will not be accepted and will receive a **grade of zero (0)**.
- After the end (11:59PM) of the due date, the assignment submission link on My.Seneca will no longer be available.
- Submitted assignments must run locally, ie: start up errors causing the assignment/app to fail on startup will result in a **grade of zero (0)** for the assignment.