SOFE2800 – Lab 3

Bootstrap, and Node.js

Activity 1: Bootstrap

1. Create a new file and save it as **index.html**.
2. Go to http://getbootstrap.com/getting-started/ and scroll to the **"Quick Start"** section.
3. Copy the provided HTML starter code and paste it into your **index.html** file.
4. Remove the <h1> tag included in the starter template.

Activity 2: Responsive Navbar with Dropdown Menus

1. **Create the Navigation Bar**
   - Inside the `<body>` tag, create a navigation bar using the `<nav>` tag with the following classes:

     ```
     <nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
     ```

     - The class `navbar-light bg-light` sets the navbar color to white.
     - To use a dark theme, you can replace it with: `navbar-dark bg-dark`
2. **Add the University Name**
   - Inside the navbar, create a link for the university name:

     ```
     <a class="navbar-brand" href="#">Ontario Tech University</a>
     ```
3. **Add the Toggle Button (for smaller screens)**
   - The toggle button is what appears when your screen size gets too small to display the full navigation bar. This ensures your navbar is **responsive**.

- Insert this button inside the navbar:

```
<button type="button" class="navbar-toggler" data-bs-toggle="collapse" 
data-bs-target=".navbar-collapse">
      <span class="navbar-toggler-icon"></span>
</button>
```

  - The `data-bs-toggle="collapse"` and `data-bs-target=".navbar-collapse"` attributes tell Bootstrap to collapse or expand the **navbar** content when the button is clicked.
  - Inside the button, the `<span class="navbar-toggler-icon"></span>` creates the three-line icon.
- Without this button, your navigation links would not adapt properly to mobile devices.

4. **Align Navigation Items to the Right**
   - By default, navigation links appear on the left side of the navbar. To improve readability and design, we align all links to the **right side**, while keeping the **university name on the left**.
   - We achieve this using:

```
<div class="navbar-collapse collapse justify-content-end" 
id="navbarNavDropdown">
        <ul class="nav nav-pills">


      </ul>
</div>
```

     - `navbar-collapse collapse` makes the menu collapsible (works with the toggle button).
     - `justify-content-end` is a Bootstrap utility class that pushes all links to the right side of the navbar.
     - Inside this container, we use an `<ul>` list with class `nav nav-pills` to hold all navigation items (**next section**). This makes them look like clickable tabs.

5. **Add Menu Items**
   - Now we start populating the navbar with actual links. Each menu item is created as a `<li>` (list item) with an `<a>` (anchor/link) inside it.
   - Example for **Home**:

```
<li class="nav-item"><a class="nav-link active" href="#">Home</a></li>
```

     - `nav-item` tells Bootstrap this is a navigation element.
     - `nav-link` active highlights the "Home" link as the current page.
   - For **About**, we remove the active class so it doesn't look selected:

```
<li class="nav-item"><a class="nav-link" href="#">About</a></li>
```

- This shows how Bootstrap styling can easily distinguish between the current page and other links.

6. **Add a Dropdown Menu**
   - Dropdown menus allow you to group related links under one menu option (e.g., Countries → Canada, USA).
   - The parent item uses `nav-item` dropdown and the dropdown-toggle class to show that it has a submenu.
   - The `data-bs-toggle="dropdown"` attribute enables Bootstrap's dropdown functionality (without extra JavaScript).
   - Inside, we create another `<ul>` with class dropdown-menu, which holds the submenu items.
   - Example structure:

```
<li class="nav-item dropdown">
    <a href="#" class="nav-link dropdown-toggle"
id="navbarDropdownMenuLink" data-bs-toggle="dropdown">
      Country <b class="caret"></b>
    </a>
    <ul class="dropdown-menu">
      <li class="dropdown-item">Canada</li>
      <li class="dropdown-item"><a href="#">Toronto</a></li>
      <li class="dropdown-item"><a href="#">Montreal</a></li>
      <li class="dropdown-divider"></li>
      <li class="dropdown-item">USA</li>
      <li class="dropdown-item"><a href="#">New York</a></li>
      <li class="dropdown-item"><a href="#">Las Vegas</a></li>
    </ul>
</li>
```

   - Here:
     - `dropdown-item` styles each submenu entry.
     - `dropdown-divider` creates a horizontal line between sections (used to separate Canada from USA).

7. **Add a Contact Link**
   - At the end of the navigation bar, we add a **Contact** link:

```
<li class="nav-item"><a class="nav-link" data-bs-toggle="modal" data-
bs-target="#contact">Contact</a></li>
```

     - `data-bs-toggle="modal"` makes it possible to open a Bootstrap modal window when clicked (useful for forms or pop-ups).
     - `data-bs-target="#contact"` points to a modal with the ID "contact".
   - This shows how Bootstrap navbars can connect to **other components** like modals, giving a professional and interactive feel.

At this stage, your code should resemble the following:

```
<body>
  <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js"
  integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r"
  crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.8/dist/js/bootstrap.min.js"
  integrity="sha384-G/EV+4j2dNv+tEPo3++6LCgdCROaejBqfUeNjuKAiuXbjrxilcCdDz6ZAVfHWe1Y"
  crossorigin="anonymous"></script>
  <nav class="navbar fixed-top navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="#">Ontario Tech University</a>
    <button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="
    .navbar-collapse">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="navbar-collapse collapse justify-content-end" id="navbarNavDropdown">
      <ul class="nav nav-pills">
        <li class="nav-item">
          <a class="nav-link active" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">About</a>
        </li>
        <li class="nav-item dropdown">
          <a href="#" class="nav-link dropdown-toggle" id="navbarDropdownMenuLink"
          data-bs-toggle="dropdown">
            Country <b class="caret"></b></a>
            <ul class="dropdown-menu">
                <li class="dropdown-item">Canada</li>
                <li class="dropdown-item"><a href="#">Toronto</a></li>
                <li class="dropdown-item"><a href="#">Montreal</a></li>
                <li class="dropdown-divider"></li>
                <li class="dropdown-item">USA</li>
                <li class="dropdown-item"><a href="#">New York</a></li>
                <li class="dropdown-item"><a href="#">Las Vegas</a></li>
            </ul>
        </li>
        <li class="nav-item"><a class="nav-link" data-bs-toggle= "modal" data-bs-target="
        #contact">Contact</a></li>
      </ul>
    </div>
  </nav>
```

Activity 3: Layout

- To draw extra attention to important information, it appears as a **blue box with rounded corners** and **larger text**, making the content inside it stand out.
- To create one:
    1. Start with a `<div>` that has the class `"container pt-5"`.
        - The container class is used to create a responsive fixed-width layout. It ensures your content is properly aligned and spaced on the page, automatically adjusting margins depending on the screen size.
        - Adding pt-5 (padding-top level 5) creates extra vertical space above the container, pushing the content downward from the top of the page. This is especially useful when working with fixed navigation bars (navbar fixed-top), since without padding, the jumbotron content could overlap with the navbar.

2. Inside it, add another `<div>` with the class `"mt-4 p-5 bg-secondary text-white rounded text-center"`.
    - `mt-4`: Adds a top margin (`margin-top`) of 1.5rem.
    - `p-5`: Adds padding of 3rem on all sides.
    - `bg-primary`: Sets the background color to Bootstrap's primary theme color (usually blue).
    - `text-white`: Sets the text color to white.
    - `rounded`: Applies slightly rounded corners to the element.
    - `text-center`: Centers the text horizontally.
3. Next, add a **header**: `<h1>Ontario Tech Trips</h1>`
4. Then add a paragraph below it: `<p>Start Your Journey Here!</p>`

- **Column Layout**: Bootstrap divides a webpage into 12 columns, allowing you to control how content is arranged on the page.
    - Create a `<div>` with the class `"row"`.
    - Inside it, add another `<div>` with the class `"col-sm-4"`.
        - This means the content will span **4 columns** (out of 12) on small screens and above and will collapse responsively.
- **Add Content About Toronto:**
    1. Insert an image of Toronto:

       ```
       <img src="img/toronto.jpg" class="img-thumbnail">
       ```

        - The class `img-thumbnail` adds borders and padding, giving the image a framed look.
    2. Add a heading and description:

       ```
       <h3>Toronto</h3>
       <p>Write a short description about Toronto here.</p>
       ```

    3. Add a button link styled by Bootstrap:

       ```
       <a href="#" class="btn btn-danger">Learn More</a>
       ```

        - The class `btn btn-danger` creates a **red button** that stands out below the paragraph.

```
<div class="container pt-5">
  <div class="mt-4 p-5 bg-primary text-white rounded text-center">
    <h1>Ontario Tech Trips</h1>
    <p>Start Your Journey Here!</p>
  </div>
  <div class="row">
    <div class="col-sm-4">
      <a href="#">
        <img src="img/toronto.jpg" class="img-thumbnail">
      </a>
      <h3> Toronto </h3>
      <p> Toronto, the capital of the province of Ontario, is a major Canadian city along Lake Ontario's northwestern shore. It's a dynamic metropolis with a core of soaring skyscrapers, all dwarfed by the iconic, free-standing CN Tower. Toronto also has many green spaces, from the orderly oval of Queen's Park to 400-acre High Park and its trails, sports facilities and zoo. </p>
      <a href="#" class="btn btn-danger">Learn More</a>
    </div>

  </div>
</div>
```

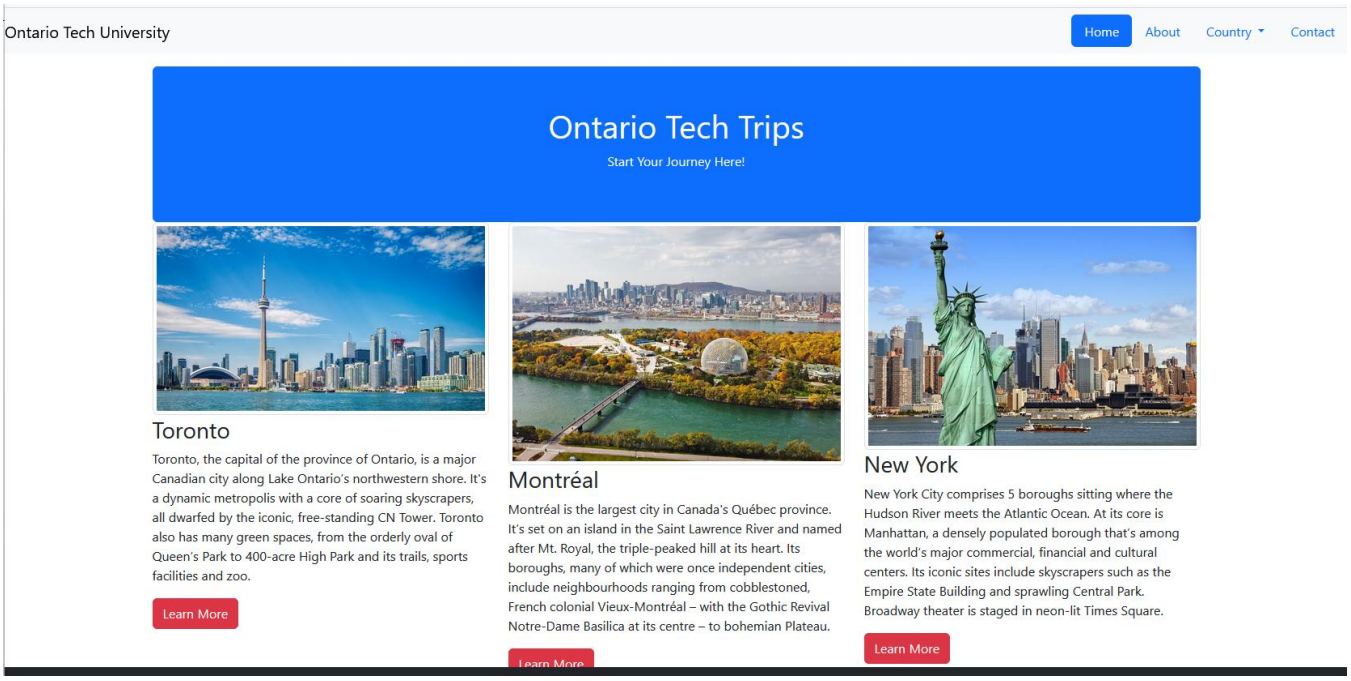
- **Apply the same steps to add sections for Montréal and New York**

```
    <div class="col-sm-4">
      <a href="#">
        <img src="img/montreal.jpg" class="img-thumbnail">
      </a>
      <h3> Montréal </h3>
      <p> Montréal is the largest city in Canada's Québec province. It's set on an island in the Saint Lawrence River and named after Mt. Royal, the triple-peaked hill at its heart. Its boroughs, many of which were once independent cities, include neighbourhoods ranging from cobblestoned, French colonial Vieux-Montréal – with the Gothic Revival Notre-Dame Basilica at its centre – to bohemian Plateau.</p>
      <a href="#" class="btn btn-danger">Learn More</a>
    </div>
    <div class="col-sm-4">
      <a href="#">
        <img src="img/york.jpg" class="img-thumbnail">
      </a>
      <h3> New York </h3>
      <p> New York City comprises 5 boroughs sitting where the Hudson River meets the Atlantic Ocean. At its core is Manhattan, a densely populated borough that's among the world's major commercial, financial and cultural centers. Its iconic sites include skyscrapers such as the Empire State Building and sprawling Central Park. Broadway theater is staged in neon-lit Times Square. </p> <a href="#" class="btn btn-danger">Learn More</a>
    </div>
```

Activity 4: Footer

- Create a `<nav>` element with the class: **`navbar fixed-bottom navbar-dark bg-dark`**
    - This will add a dark-colored navigation bar that stays fixed at the bottom of the page.

Congratulations — you're done! At this point, your webpage should look similar to the example shown below.



Activity 5: Creating a Basic HTTP Server in Node.js

In this activity, you will learn how to create a simple HTTP server that displays **"Hello World."**

1. **Create a new file** and save it as **server.js** in your workspace directory.
2. At the top of your file, **import the HTTP module** by declaring a variable named `http` and assigning it as follows:

   ```
   const http = require('http');
   ```

   This line loads Node.js's built-in HTTP module and makes it accessible through the **`http`** variable.

3. **Create the server** using the **`createServer()`** method and store it in a variable called `server`. The **`createServer()`** method takes a callback function named **`onRequest`** as its argument.
4. **Define the `onRequest` function**, which accepts two parameters — **`req`** (request) and `res` (response).
5. Inside this function, use **`res.writeHead()`** to set the HTTP header:

   ```
   res.writeHead(200, {'Content-Type': 'text/plain'});
   ```

6. Next, use **`res.write()`** to send the message **"Hello World"** to the browser, and then call **`res.end()`** to finish the response:

   ```
   res.write('Hello World');
   res.end();
   ```

7. The `createServer()` function returns an object with a `listen()` method, which specifies the port and IP address your server will use. For example:

   ```
   server.listen(1337, '127.0.0.1');
   ```

8. To confirm that your server is running, add a console message:

   ```
   console.log('Server running at http://127.0.0.1:1337/');
   ```

9. **Save your file.**

```
const http = require('http');

var server = http.createServer(onRequest);

function onRequest(req, res){
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.write('Hello World');
    res.end();
}

server.listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

10. **Run your server:**
    - Open the Command Prompt (search for `cmd.exe` in the Start menu).
    - Navigate to your workspace directory:

      ```
      cd yourWorkplaceDirectory
      ```

    - Start your server by running:

      ```
      node server.js
      ```

11. **Test your server:**
    Open your web browser and go to **http://127.0.0.1:1337/**. You should see **"Hello World"** displayed on the page.

Activity 6: Hosting a Web Page with Node.js

In the previous activity, you created a simple server that displayed plain text. Now, you will modify that code to host actual web pages using Node.js.

1. **Open your existing `server.js` file.**
   Go to **File → Save As**, and save a new copy as **ServerFile.js**.
   Keep the same code from `server.js` for now.

2. **Import the File System (fs) module.**
   At the top of your file, initialize a variable named `fs` and load the `fs` module:

   ```
   const fs = require('fs');
   ```

3. **Update the `onRequest` function.**
   Delete the old code inside the `onRequest` function.
   We'll now create logic that checks whether the incoming HTTP request is a **GET** request for the correct URL, and then serves the desired web page.

```
if (req.method === 'GET' && req.url === '/') {
    res.writeHead(200, { "Content-Type": "text/html" });
    fs.createReadStream("./index.html").pipe(res);
}
```

4. **Ensure all necessary files are in the same directory.**
   Place your `ServerFile.js` and `index.html` files together so that the server can locate the webpage when requested.

5. **Handle missing or incorrect pages.**
   Create a new function called `send404Response()` to handle cases where the requested page is not found. This function will send an error message to the client. For example:

```
function send404Response(res) {
    res.writeHead(404, { "Content-Type": "text/plain" });
    res.write("Error 404: Page Not Found!");
    res.end();
}
```

Your updated code should look like this:

```
const http = require('http');
var fs = require('fs');

var server = http.createServer(onRequest);

function onRequest(req, res){
    if( req.method=='GET' && req.url=="/" ){
        res.writeHead(200, {'Content-Type': 'text/html'});
        fs.createReadStream("./index.html").pipe(res);
    }
    else{
        send404Response(res);
    }
}

function send404Response(res) {
    res.writeHead(404, { "Content-Type": "text/plain" });
    res.write("Error 404: Page Not Found!");
    res.end();
}

server.listen(1337, '127.0.0.1');

console.log('Server running at http://127.0.0.1:1337/');
```

Exercises:

In this Bootstrap activity, you will enhance your webpage layout and implement a modal dialog box.

1. **Adjust the container spacing:**
   Locate the following line in your HTML file:

   ```
   <div class="container pt-5">
   ```

   Update it by adding a Bootstrap class that adds padding to the bottom of the division. This ensures the content will not be hidden behind the footer.

2. **Implement a Bootstrap modal:**
   A **modal** is a Bootstrap JavaScript component used to display dialogs such as lightboxes, alerts, or custom content.

   In your navigation bar, the **Contact** item is already configured to trigger a modal with the ID **contact**, as shown below:

   ```
   <a class="nav-link" data-bs-toggle="modal" data-bs-target="#contact">
   Contact</a>
   ```

   However, the modal itself has not yet been created.

   Using the example provided in the Bootstrap documentation at https://getbootstrap.com/docs/5.3/components/modal/#modal-components,
   add a modal component to your page so that when the **Contact** item is clicked, a modal appears similar to the one shown in the figure.