BTI225 Assignment 4 [60 marks]

Objective: Practice writing DOM and CSS.
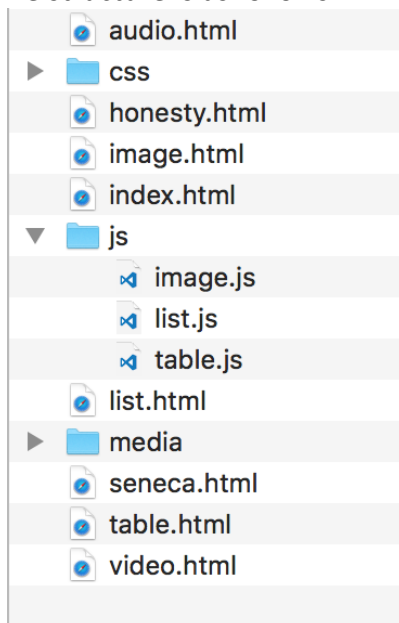
Specification:
This assignment is built upon Assignment 3. If you are unable to start this assignment because Assignment 3 was incomplete - email me for a clean version of the Assignment 3 files to start with.
This assignment includes 2 parts, DOM and CSS.

Specification (Part A):

You will add DOM features to the existing HTML files in your A3. See details below:

1. Download *a4.zip* from blackboard and unzip it.
2. Backup your A3 work. A4 will be built on A3. After unzip a4.zip, copy the directory *"PartA/js"* with its files to your A3 working directory where stores A3 html files and two directories *"media"* and *"css"*. When you transfer your work to matrix, make sure you upload all the files and directories in the same structure to *"public_html"* on matrix. The file structure is as follows:

```
    audio.html
▶   css
    honesty.html
    image.html
    index.html
▼   js
        image.js
        list.js
        table.js
    list.html
▶   media
    seneca.html
    table.html
    video.html
```

"HTML Lists" - list.html: (20 marks)

Update the "HTML Lists" page as follows:
In the end of the file, before closing </section>, after the list definition.

1. Include the *list.js* file to *list.html (<script>)*.   **[1 mark]**
2. Provide a comment <!-- A4 -->
    Draw a horizontal line (<hr>). Add a **header "Fruit List" (<h2>)** as a title for the next

section (step 4)

3. create a container (i.e.: <div>) with unique id value (e.g., "list1"). Be prepared for the following step 4. **[1 mark]**

4. Use the **fruits** array inside the **lists.js** file (js/list.js) to write (using JavaScript) an ordered list containing all the fruit inside the array to the above container (id "list1", which was created in step 3) in the list.html page. Be sure to write to your container (using JavaScript) **after** the window has completely loaded (window.onload). The ordered list should look like the image below when complete: **[5 marks]**

1. Apples
2. Oranges
3. Pears
4. Grapes
5. Pineaples
6. Mangos

5. Draw a horizontal line (<hr>). Add a button **"Add Lists of Directories and Files"**. When user clicks this button, it will add a nested list to the following container (*list2, see next step*), which will be generated using JavaScript as described below. **[1 mark]**

(<button onclick="addList2();"> …. </button>)

6. Create a container (i.e.: <div>) with unique id value (e.g., "list2"). Be prepared for the following step 7.

7. Use the **directory** array inside the **list.js** file (js/list.js) to write (using JavaScript) a <u>nested unordered list </u>containing all the files and directories (with their files) inside the array to the above container (id "list2", which was created in step 6) in the list.html page. (refer to the following screenshot). **[12 mark]**

In js/list.js, you will notice that files have the type "**file**" and consist of a "**name**" property, whereas directories have the type "**directory**" and consist of a "**name**" property in addition to an **array of files**. Use these properties to correctly construct your nested unordered list (using JavaScript).

The nested unordered list should look like the image below when complete:

- file1.txt
- file2.txt
- HTML Files
  - file1.html
  - file2.html
- file3.txt
- JavaScript Files
  - file1.js
  - file2.js
  - file3.js

"HTML Tables" - table.html: (15 marks)

Update the "HTML Tables " page as follows:

1. Include the *table.js* file to *table.html (<script>)*.   **[1 mark]**
2. Provide a comment <!-- A4 -->   **[1 mark]**
   Draw a horizontal line (<hr>). Add a **header "User Information" (<h2>)** as a title for the next section (step 4)
3. **Update the table.html page to include an additional container (i.e.: <div>) with a unique id value.**   **[1 mark]**

  4. Use the **users** array inside the **table.js** file (js/table.js) to write a complete **table** containing all the users inside the array (and an appropriate header row), to your container in the table.html page. **[4 (header) +6(rows) + 2 (email link) = 12 marks]**

You will notice that users consist of the properties: **first_name**, **last_name**, **age**, and **email**.

hint: use *for/in* loop to loop through the object properties, then use notation *obj[prop]* to access the value for each property.

Use the following to set the border for *td*. You can do the similar for *table* element

    td.style.border = "2px solid";

Use these properties to correctly construct your table (using JavaScript) with the following headers: **First Name**, **Last Name**, **Age** and **Email**. You must also ensure that all email addresses are rendered as a valid "mailto" link and will open as a new message in the user's default mail client when clicked.

Be sure to write to your container **after** the window has completely loaded. The table should look like the image below when complete:

First Name	Last Name	Age	Email
Kaitlin	Burns	23	kburns99753@usermail.com
Joshua	Feir	31	josh319726@usermail.com
Stephen	Shaw	28	steve.shaw47628@usermail.com
Timothy	McAlpine	37	Timbo72469@usermail.com
Sarah	Connor	19	SarahC6320@usermail.com

"HTML Images" - image.html: (10 marks)

Update the "HTML Images " page as follows:

1. Include the *image.js* file to *image.html (<script>)*.   **[1 mark]**
2. Provide a comment <!-- A4 -->
   Draw a horizontal line (<hr>).
3. **Update the image.html page to include an additional container (i.e.: <div>) with a unique id value.**   **[1 mark]**
4. Use the **images** array inside the **image.js** file (js/image.js) to write 5 new **<figure>** elements to your container in the image.html page (using JavaScript). You will notice that each image in the array has the following properties: **caption**, **alt** and **url**. Use these properties to correctly construct (using JavaScript) your **<figure>** elements such that each element contains a valid **image** (using **url** and **alt)** as well as a **caption** underneath the image with the message contained within the **caption** property   **[8 marks]**

   Hint: HTML tag <figure> <figcaption> in lecture note, bti225-wk6.ppt, page 13

<figure> specifies self-contained content, frequently with a caption (<figcaption>), and is typically referenced as a single unit. Example HTML code snippet:

```
<div class="picture">
  <figure>
    <figcaption>
       This is a figure caption
    </figcaption>
    <img src="image-01.jpg" alt="landscape 1"
         title="landscape 1" />
  </figure>
</div>
```

In this assignment, you are going to use JavaScript to generate the <figure> and <figcaption> elements. The following screenshot shows the first 2 images as example. Your work should display all the images with captions.

Be sure to write to your container **after** the window has completely loaded (using JavaScript).

Red Slate Mountain

Indonesian Jungle

Specification (Part B): (15 marks)

For Part B, we will create a new CSS file to be used in the *"CanadianPT.html"* file located within the PartB folder.

Open the uncompressed folder (a4) and locate the directory "PartB". Copy the file *"CanadianPT.html"* and the directory *"images"* to A3 working directory, which will later be transferred to the directory *"public_html"* on your matrix. The file structure is as follows

Getting Started

- locate the "**css**" directory and add/create the file "**site.css**".
- Open file "canadianPT.html" for editing and add the <link> tag to include your new site.css file.
- Open both "site.css" and "canadianPT.html" for editing/viewing.
- Open "index.html", in the <nav> <div> section, in the end before <li> …. Honesty , add a new list item <li>. Its content/text is "Canadian Provinces and Territories", click this text, it will open the "canadianPT.html" page in a new tab. The following is part of "index.html" screenshot. **[1 mark]**

BTI225 - Assignments

| Home | List | Table | Image | Audio | Video | Seneca | **Canadian Provinces and Territories** | Honesty |

Welcome

- Open "canadianPT.html" in a web browser or click the tab/menu item from the above "index.html". It looks like the following (the screenshot is not complete, you should see more content):

Canadian Provinces and Territories

Canada

- Alberta
- British Columbia
- Manitoba
- New Brunswick
- Newfoundland and Labrador
- Nova Scotia
- Northwest Territories
- Nunavut
- Ontario
- Prince Edward Island
- Quebec
- Saskatchewan
- Yukon

Alberta is Canada's fourth-most populous province and most populous of Canada's three prairie provinces. Alberta and its neighbour, Saskatchewan, were established...
Read more

British Columbia is also a component of the Pacific Northwest, along with the U.S. states of Oregon and Washington. The province's name was chosen by Queen...
Read more

Manitoba is a Canadian prairie province. The province, with an area of 649,950 square kilometres (250,900 sq mi), has a largely continental climate, with...
Read more

New Brunswick is one of Canada's three Maritime provinces and is the only province in the Canadian federation that is constitutionally bilingual...
Read more

Step 1 – Styling the main containers (6 marks)

- Notice how the page "**canadianPT.html**" is broken up into major sections: **<div class="main-container">…</div>**, **<header>…</header>, <nav>…</nav>, <section class="main">…</section>** and **<footer>…</footer>**

- We need to apply styles to these sections, so add the following CSS Selectors (in the file **site.css**) to your page. For each selector, use the suggested CSS Properties to achieve each design requirement. See the completed sample for reference

CSS Selector	Design Requirement	Suggested CSS Properties
.main-container	• Set the maximum width to 960 pixels • No top or bottom margins • Left and right margins must be "auto"	• max-width • margin
.main-container nav	• Set the width to 30% • Element is positioned on the left side of the page	• width • float
.main-container .main	• Set the width to 70% • Element is Positioned on the right side of the page	• width • float
.main-container header	• Padding around the content must be 10 pixels • Background colour must be #eeeeee • There must be a solid bottom border 1 pixel thick, coloured #cccccc • Bottom margin must be 30 pixels	• padding • background-color • border-bottom • margin-bottom
.main-container header h1	• Set the width to 70% • The size of the font must be 24 pixels • Element is Positioned on the right side • The text is right-aligned • Top margin must be 10 pixels	• width • font-size • float • text-align • margin-top
.main-container footer	• position the element beneath all floating elements ("clear" the element) • Padding around the top and bottom must be 20 pixels	• clear • padding • background-color • border-top • font-size

	• Padding around the left and right must be 10 pixels • Background colour must be #eeeeee • There must be a solid top border 1 pixel thick, coloured #cccccc • The size of the font must be 13 pixels	

After you finish the above CSS, the *canadianPT.html* page looks like the following (the screenshot is not complete, you should see more content):

Step 2 – Styling the navigation and logo (4 marks)

- Now that the page is looking a little more organized, we want to update the "Canada" logo (flag & name) as well as the side navigation to look a little cleaner. For each selector, use the suggested CSS Properties to achieve each design requirement. See the completed sample for reference

CSS Selector	Design Requirement	Suggested CSS Properties
#logo	• Set the width to 30% • The size of the font must be 25 pixels • The font must be bold	• width • font-size • font-weight
#logo img	• The image must be 94 pixels wide • The image must be vertically positioned in the middle, relative to the "Canada" text	• width • vertical-align
nav ul	• Do not show any bullets or numbers for the list	• list-style-type
nav ul a	• render the link as a "block-level" element • remove the underline • Padding around the link must be 5 pixels	• display • text-decoration • padding

After you finish the above CSS, the *canadianPT.html* page looks like the following (the screenshot is not complete, you should see more content):

Step 3 – Styling the province containers (complete CSS) (4 marks)

- Everything is looking pretty good, but the page is very vertical. For a more modern design, we will render all of the provinces as "cards" and allow them to sit side-by-side horizontally. We also want to allow them to move onto the next line and render correctly if the user decides to resize the browser window. For each selector, use the suggested CSS Properties to achieve each design requirement. See the completed samples for reference

CSS Selector	Design Requirement	Suggested CSS Properties
div.province	• Set the width to 30% • Each new element is positioned to the right of the previous element, horizontally (**hint**: float:**left**) • The margin around the element must be 8 pixels • The height of the element must be 310 pixels	• width • float • margin • height
div.province .description	• The element must not be any higher than 165 pixels • Only If the content goes beyond 165 pixels high, show a scroll bar • If the content does not fit in the container horizontally, hide the scrollbar • The top and bottom margins must be 8 pixels	• max-height • overflow-y • overflow-x • margin
div.province a	• render the link as a "block-level" element • position the text in the middle of the element (horizontally) • set the colour of the background to #555555 • Set the colour of the text to #eeeeee • remove the underline • Padding around the link must be 5 pixels	• display • text-align • background-color • color • text-decoration • padding
img.flag	• set the width to be 100% of the parent	• width

After you finish the above CSS, the *canadianPT.html* page looks like the following (the screenshot is not complete, you should see more content):

If you resize to a smaller screen, you will see something like the following screenshot:

Other Requirements

- All tags/attributes must be in **lower case.**
- Validate all your html files using the W3C Markup Validation Service: https://validator.w3.org/ .
- Validate all your CSS using the **W3C CSS Validation**: https://jigsaw.w3.org/css-validator
- Include the timestamp in each page. You can simply include the following code to your HTML file at a place that is easy to spot.

```
<script type="text/javascript">
          var dt=new Date(document.lastModified);
          document.write('This page was last updated on '+dt.toLocaleString())
     </script>
```

Submission:

You do not need explicitly submit the work. Leave it on matrix account, which will be checked after due date. **DO NOT need to email me or submit to blackboard**.

Late submission:

10% penalty of the assignment each day, up to 5 school days. It will not be accepted after 5 school days late and will be marked zero for this assignment.