WEB700 Assignment 4

# Submission Deadline:

Sunday, July 6th @ 11:59 PM

# Assessment Weight:

9% of your final course Grade

# Objective:

Create and demonstrate a new "theme" for your application by creating custom CSS to be used across the site, as well as rendering common HTML elements to showcase the theme. Finally, we will extend the functionality of our "legoSets" module to add new sets.  
  
**NOTE:** If you require a *clean version* of Assignment 3 to begin this assignment, please email your professor.

# Specification:

### **Step 1:** HTML Elements Demo (about.html)

A large part of this assignment will be creating and demonstrating a new "theme" for your application. However, before we begin with the CSS for the theme, we should first add several HTML elements to showcase the various aspects of the theme and how it will appear once it has been applied to the site. This will involve creating elements such as tables, lists, paragraphs, links, images as well audio & video.

We will showcase these HTML elements beneath the <section>…</section> element on the about.html page. To begin, write a short paragraph describing yourself (ie: what you're studying at Seneca, hobbies, etc, whatever you like).

Once this is complete, create a relevant **header** and **paragraph** to describe the content of the next section - this will be the "Theme Demo".

Once this is complete, create the following sections on your page:

"Tables" Section

* This section must have a relevant **header** to distinguish it from the rest of the page
* Next, it must consist of **two** HTML tables that must contain a **minimum** of **3 rows** and **3 columns** and must make use of the following elements / properties:
  + caption
  + thead,tbody,tfoot
  + th, td, tr
  + the "rowspan" property and the "colspan" property
* The cells in the tables **cannot be empty**, but can contain any content that you wish (images, links, text, etc)
* At the bottom of the section, write a short **paragraph** describing the content in the section

"Lists" Section

* This section must have a relevant **header** to distinguish it from the rest of the page
* Next, it must consist of a single list (either unordered or ordered) with a minimum of **5** **items** – at least one of these items should contain a **list** that is a **different type** than the parent list (ie: if the parent is an ordered list, then one of the items should contain an unordered list, and vice versa.) with at least **3 items.**
* The items in the list **cannot be empty**, but can contain any content that you wish (images, links, text, etc).
* At the bottom of the section, write a short **paragraph** describing the content in the section

"Images" Section

* This section must have a relevant **header** to distinguish it from the rest of the page
* Next, it must consist of at least 3 images that include "alt" text. You're welcome to use any images you like (as long as they can be reused for "noncommercial" purposes). **NOTE:** the images **do not** have to exist on your sever – you may use complete paths to the images, ie:  
    
  <https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/Lego_Color_Bricks.jpg/440px-Lego_Color_Bricks.jpg>
* At the bottom of the section, write a short **paragraph** describing the content in the section

"Audio" Section

* This section must have a relevant **header** to distinguish it from the rest of the page
* This section will consist of an HTML 5 audio player. For the audio source, you may use the following files if you wish (not mandatory if you have found something else that you like better)
* <https://pat-crawford-sdds.netlify.app/shared/winter-2025/web700/A4/sample-audio.mp3>
* <https://pat-crawford-sdds.netlify.app/shared/winter-2025/web700/A4/sample-audio.ogg>
* At the bottom of the section, write a short **paragraph** describing the content in the section

"Video" Section

* This section must have a relevant **header** to distinguish it from the rest of the page
* This section will consist of an HTML 5 video player. For the video source, you may use the following files if you wish (not mandatory if you have found something else that you like better)
  + <https://pat-crawford-sdds.netlify.app/shared/winter-2025/web700/A4/sample-video.mp4>
  + <https://pat-crawford-sdds.netlify.app/shared/winter-2025/web700/A4/sample-video.webm>
* At the bottom of the section, write a short **paragraph** describing the content in the section.

### **Step 2:** CSS for custom "Theme" (theme.css)

Now that we have our "Theme Demo" section complete, it's time to create our custom theme using CSS:

* Within the root of your application, create a new folder called "public"
* Within the "public" folder, create a "css" folder (this is where we will place our CSS file(s)
* Finally, within the "css" folder, create a "theme.css" file.
* Now that we actually have a "theme.css" file (albeit an empty one), we must include it in all of our "views" (.html files), including:
  + home.html
  + about.html
  + 404.html

This will involve using the appropriate <link> element in the <head> of your files (**after** the bootstrap CSS).

* With this complete, we must use the built in express "static" middleware within our server.js file to identify our newly created "public" folder as a source for static files.

Updating "theme.css" to provide a unique look / feel for your application

Now that we have a "theme.css" file and its correctly linked in our html files, we can start to personalize our web app by adding some CSS. There are plenty of resources online to help you pick colours and find (as well as generate) interesting styles to apply to selected elements. Some quality resources to get your started include:

* This "Colour Wheel", used to pick complimentary colours and get their "hex" values: <https://www.canva.com/colors/color-wheel/>
* A "Box Shadow" Generator, used to provide the complete CSS for adding a "box shadow" to an element: <https://www.cssmatic.com/box-shadow>
* Sample CSS for generating a "full page gradient" (**Note:** for this example to work for us, we must change the selector from "html" to "body, html" and instead of "#red" and "#blue", simply use "red" and "blue" – or whatever other colours you like): <https://coderwall.com/p/ape0jg/full-page-gradient-background>

Additionally, to ensure that your gradient covers the entire page, regardless of how far you scroll, you must add "**overflow: auto;"** as well.

* Google Fonts, used to generate an @import statement to add to our "theme.css" file that will enable us to use 3rd party "web fonts" (**Note:** Add the @import statement at the top of your theme.css file and add the "font-family" style to whatever element you wish to style the font): <https://fonts.google.com/>

Regarding our own app, we must style the following elements to have an impact on our layout (see the table below).

**Note:** This is the **minimum** amount of styling required. Please feel free to add **additional styles and html** if you have a specific vision for your app that goes beyond the requirements.

Elements and Styles

The below table can be used as a guide to help you style your application. To get a sense of what your application could look like after applying you own styles to the following elements, please see the following "before" and "after" images (**NOTE**: Consider adding class="table" to your <table> tags – this was done in the sample):

|  |  |
| --- | --- |
| Before | After |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Element** | **Style Suggestions** |
| Navigation Bar & Hero   * Element with class "bg-light" | * Background colour or gradient * Box shadow (outline or inset) |
| Navigation "Brand", ie: "Student Name"   * Element with class "navbar-brand" inside an element with class "navbar-light" | * Cool font from "Google Fonts" (do not forget the @import statement) * New Text Colour * Text shadow |
| Navigation "Link", ie "Home", "About", etc. etc.   * Anchor element with class "nav-link" * Anchor element with class "nav-link" when "hovered" | * Cool font from "Google Fonts" (do not forget the @import statement) * New Text Colour / hover colour, etc. |
| App Background   * Both body and html elements | * Full page gradient * Solid background colour * Image (consider using "cover" - <https://www.w3schools.com/cssref/css3_pr_background-size.asp>) |
| Table cells inside "odd" numbered table rows   * "td" elements, inside all (odd) "tr" elements (see [:nth-child(odd)](https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child) pseudo-class), inside a "table" element | * Solid Background Colour * Gradient Background |
| All "Heading" elements   * All elements from "h1" to "h6" | * Cool font from "Google Fonts" (do not forget the @import statement) * New Text Colour   Text shadow |
| All "Anchor" elements (HTML Links)   * "a" elements * "a" elements when hovered | * New Text Colour |
| All "List" elements   * "ol" and "ul" elements * "li" elements inside "ol" / "ul" elements | * Fix spacing (consider adding padding / margins to left, top or bottom) |

Once the above elements have been styled, there should be a drastic change in the appearance of the web app. However, if you're enjoying tinkering with the look and feel, please feel free to style additional elements or update the existing html structure of the layouts.

### **Step 3:** Adding New Lego Sets

In the next assignment(s), we will be allowing users to enter their own Lego Sets in the collection using an [HTML Form](https://web700.ca/Working-With-Forms/html-form-elements-overview). To facilitate this, we must update our legoSets module with the following method to the "LegoData" class:

addSet(newSet)

* This method must return a new Promise object, such that:
  + If the set\_num property of the newSet object **already exists** on an object in the "sets" array, then the Promise must be rejected with an appropriate message, such as: "Set already exists"
  + If the set\_num property of the newSet object **does not exist** on any objects within the "sets" array, then add the newSet object to the "sets" array and resolve the promise without any data

To test this functionality, create a new route in your server.js file that:

* Defines a new Lego set that does not exist in the "sets" array, for example:

let testSet = {

set\_num: "123",

name: "testSet name",

year: "2024",

theme\_id: "366",

num\_parts: "123",

img\_url: "https://fakeimg.pl/375x375?text=[+Lego+]"

}

* Attempts to add the above "testSet" by invoking the newly created "addSet" method  
  + If successful, redirect to the "/lego/sets" route
  + If not successful, set the status code to [422](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/422) and return the error.

After you have implemented this method, you should be able to access the http://localhost:8080/lego/add-test route and see that your new set has been added. If the route is accessed again without restarting the server, you should see your error with a 422 status code, since the set has been added

### **Step 4:** Updating Your Deployment

Finally, once you have tested your site locally and are happy with it, update your deployed site by pushing your latest changes to GitHub.

**IMPORTANT NOTE:** Now that we are using the express.static() middleware, do not forget to use:

**app.use(express.static(\_\_dirname + '/public'));**

See <https://web700.ca/Resources/vercel-guide> for more information.

# Assignment Submission:

1. Add the following declaration at the top of your **server.js** file:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* WEB700 – Assignment 04

\*

\* I declare that this assignment is my own work in accordance with Seneca's

\* Academic Integrity Policy:

\*

\* https://www.senecapolytechnic.ca/about/policies/academic-integrity-policy.html

\*

\* Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Student ID: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*

\* Published URL: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. Compress (.zip) your assignment folder and submit the .zip file to My.Seneca under   
   **Assignments** -> **Assignment** 4

# 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.