WEB222 Final Assessment – Fall 2022

Assessment Weight:
20% of your final course

Submission Deadline:

1. Phase 1: Dec 1, 2022 during the class. In this phase, the team works on project planning, and start building some basic steps of the project. This step should be submitted by the end of Thursday class.
2. Phase 2: Dec 5 submit deliverables that has been identified in phase 1
3. Phase 3: Dec 8 submit deliverables that has been identified in phase 1
4. Phase 4: Dec 11 Project submission + video
5. Phase 5: Dec 12 Project presentation/evaluation

Overview

The WEB222 final assessment is equivalent to a final exam but can be done as a group project.

The assessment has three parts and builds on your knowledge from the course to create websites. Specifically, you will be asked to do the following:

1. Create a sample product/service Website which provides a dynamic list of service or product to users

    - This could be an online store, and music hub page, a Christmas,birthday gift service, …In the 1st phase of the project you need to choose the scope and prepare the data
    - Your site will sell/provide several different product/service categories, and many products in those categories. Because a store's products and categories will change frequently, we often separate our data from its UI representation. This allows us to quickly make changes and have the store's web site always use the most current inventory information.
        0. *NOTE: in a real e-commerce web store, our data would be stored in a database. We will simulate working with a database by using JavaScript Objects and Arrays.*
        1. *You need to prepare your own dataset and display them dynamically in the page (using card design)*

2. Research and Implement a Static Hosting solution to deploy the project into any of the online hosting services (Vercel, Netlify,….), so it can be viewed online (worth 5%)

Task 1: Pick Your Store and Product/Service Inventory

You need to decide on the following details for your store:

- **Name**: what is your store called? Pick something unique and relevant to your products.

- **Slogan or Description**: what is your store's slogan or what is a short description of what you sell? This will help a user to determine if your store's site is worth reading.

- **Products**: what does your store sell? Baked goods? Ferraris? Cosmetics? Candles? Sneakers? It's up to you! Pick something that no one else is going to choose. No two students can use the same store products. Your store must have a **minimum of 20 items, and at least 2 of these should be Discontinued (see below)**. You are free to make things up. Be creative.

- **Product Categories**: your products will fit into one or more categories. For example, if you are selling Winter Gloves, you might have the following categories: "Men's Gloves", "Women's Gloves", and "Children's Gloves" or maybe "Active", "Formal", "Decorative". Your store should have a **minimum of 4 categories**. Each product must belong to one or more of these categories.

Task1-1: Modelling your Store Data

Categories

Each category needs two things:

- **id**: a unique **String** that identifies this category. For example: "c1" or "category-01" or "V1StGXR8". It doesn't matter what format you choose as long as each category has its own unique value.

- **description**: a human-readable **String** meant for display. While the id is a unique key for the data used by programs, the description is meant to be shown to a user. For example: "Men's Shoes" or "Pickup Trucks" or "Skydiving Tours."

Products/Services

Each product needs the following things:

- **id**: a unique **String** that identifies this product. For example: “p1” or “product-01” or “V1StGXR8”. It doesn’t matter what format you choose as long as each product has its own, unique value. Also, make sure the product id and category id are different.

- **name**: a short **String** that names the product (e.g., "Gingerbread Cookie")

- **description**: a longer **String** that defines the product

- **price**: a **Number** of whole cents (i.e., an Integer value) for the product’s unit price. When we store currency data, we often do so as integers vs. floats, and convert it for display (e.g., 100 = $1.00, 5379 = $53.79)

- **discontinued**: a **Boolean** indicating whether or not the product has been discontinued. If this property is absent, your system should assume that it is NOT discontinued.

- **categories**: an **Array** that includes one or more category ids. Each product belongs to one or more categories (e.g., ["c1"] or ["c1", "c2"]). Make sure you match the category id to your format above.

Your category and product data will go in `src/categories.js` and `src/products.js` respectively. See these files for technical details about how to code your data.

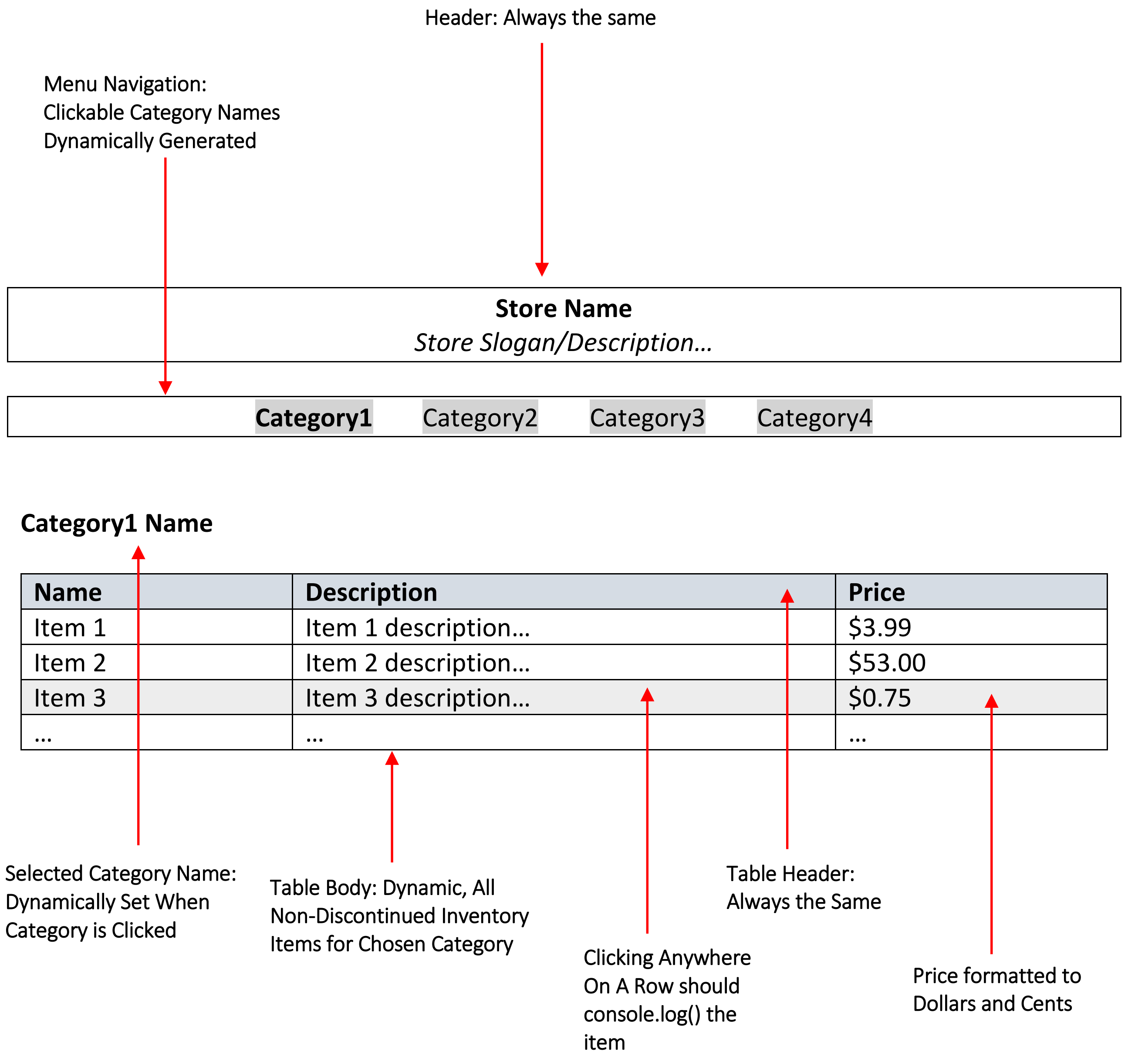
Take some time now to enter all of your store’s data.

Task 2: Web Site Layout

You need to design the layout for your website. In phase 1, you need to design the wireframe and later you design the site using HTML/CSS/JS. You site should include a section to show the list of available products/services, a section about team member (include your resume + skills + hobbies, similar to Assignment 1), and a FORM (can be contact , feedback, ….). The Layout should contain a menu which help user to navigate etween each section of the site

Some of your site will be static (i.e., coded in HTML directly in index.html) and never change. Other parts of the site will be dynamic (i.e., created using DOM API calls at run-time) and will update in response to various events and user actions.

Here is a basic wireframe of what your site needs to include (to show list of products/services in each category), and which parts are static or dynamic. NOTE: don’t worry too much about how it looks. Focus on the structure and functionality.



Your website will be expected to have the following structure and elements:

- Make sure you choose the most appropriate, **semantic HTML5 elements** for all of your page's content (e.g., not everything should be a <div> or <p>). Your page must be valid HTML5.

- All CSS and JavaScript should be put in separate .css or .js files (i.e., not embedded in the HTML), and use proper indentation, formatting, and include appropriate comments.

- All website content should be centered in the viewport, having equal margins on the left and right. The page content should cover 90% of the viewport width (i.e., the width of the browser window). However, the content width should be limited to 1,100 pixels, never growing larger than that.

- Your site should use elements of a **Responsive Design**. That is, it should work equally well, and optimize space and sizes, on both desktop screens (more than 400 px wide) and mobile phones (less than 400 px wide). Research and use CSS Media Queries to define classes and rules that work on a narrow screen (400 pixels or less) vs. your full desktop (greater than 400 pixels). For example, font sizes, margins, layout choices (e.g., removing or moving elements), image sizes, etc. could all be different if viewed on a phone vs. desktop. There are lots of ways to implement a Responsive web site. Research and use a few of these techniques in order to accomplish this requirement.

Task 3: Dynamic Content

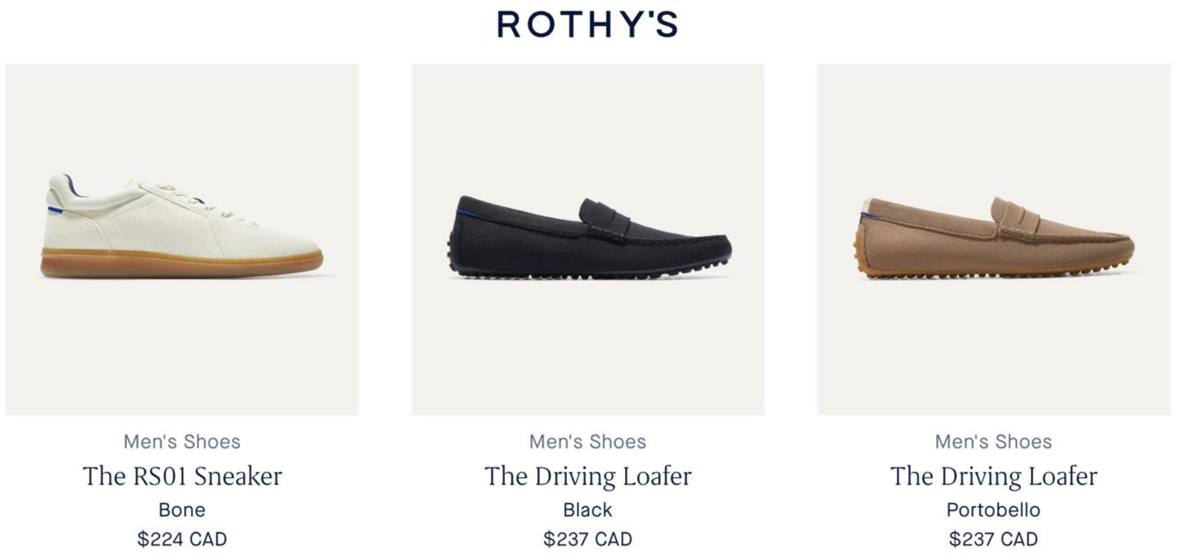
All of your store's dynamic content will be written in JavaScript in the `src/app.js` file. Using JS, you can display the product/service data from a JS array into a proper format on the page (like using table)

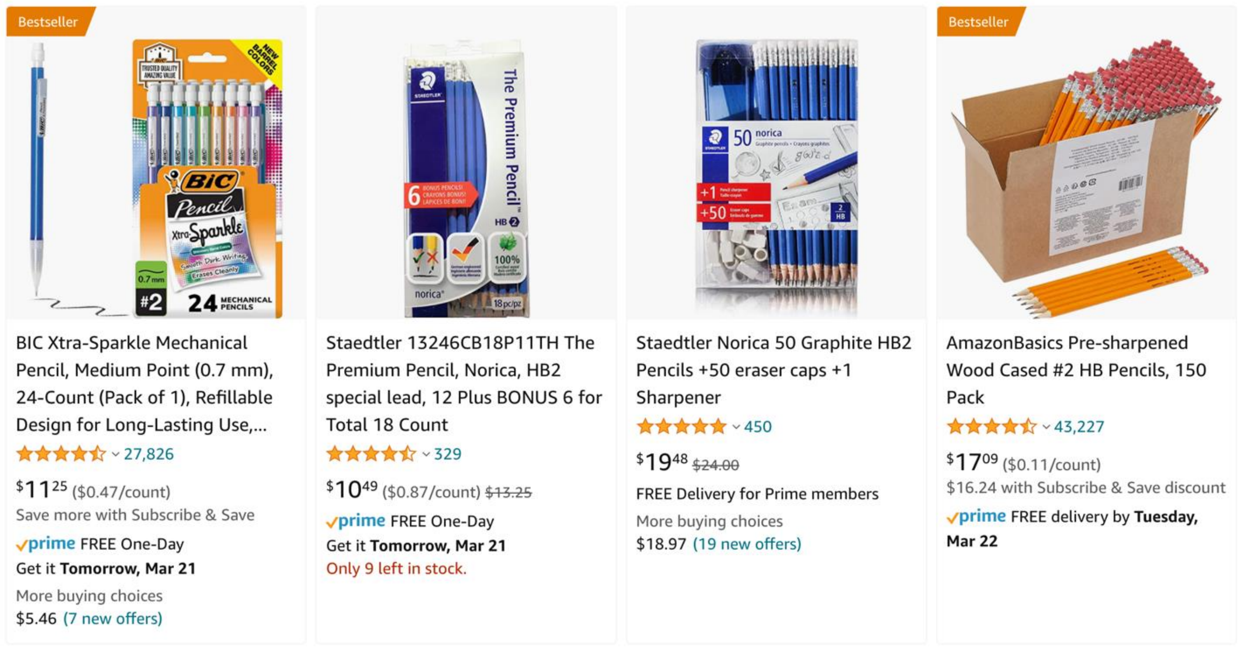
However, instead of displaying your products in an HTML table, you will create visual product "cards" that show a picture, name, description, and price.

You must do all the work for this assignment on your own. You may consult your notes, use the web for inspiration, but you should not copy code directly from other sites, or other students. If you need help, ask your professor.

Cards

Cards on the web, much like trading or playing cards, are rectangular areas that allow you to visually present a lot of related data. We often see them used in online stores, social media, and anywhere that we want to mix images, titles, and text in a rectangle. Here are some real-world examples from Rothy's, Amazon, and airbnb:




Bestseller

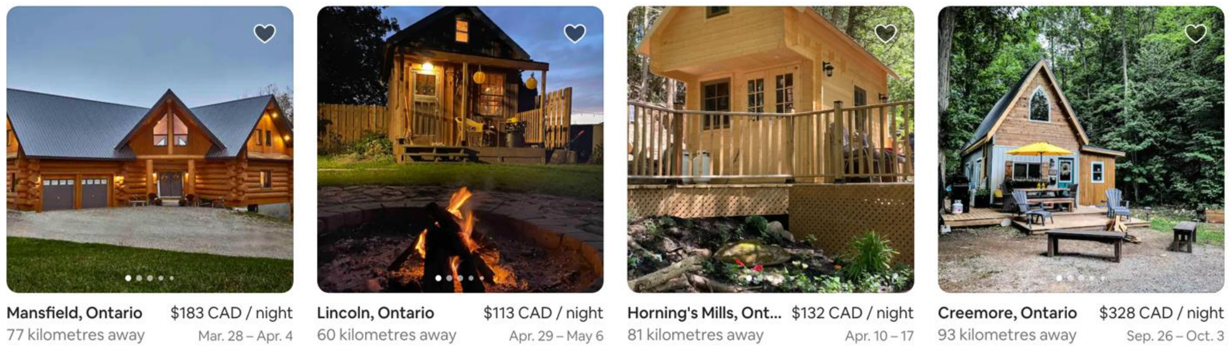
BIC Xtra-Sparkle Mechanical Pencil, Medium Point (0.7 mm), 24-Count (Pack of 1), Refillable Design for Long-Lasting Use,...
27,826
$11.25 ($0.47/count)
Save more with Subscribe & Save
prime FREE One-Day
Get it **Tomorrow, Mar 21**
More buying choices
$5.46 (7 new offers)

Staedtler 13246CB18P11TH The Premium Pencil, Norica, HB2 special lead, 12 Plus BONUS 6 for Total 18 Count
329
$10.49 ($0.87/count) ~~$13.25~~
prime FREE One-Day
Get it **Tomorrow, Mar 21**
Only 9 left in stock.

Staedtler Norica 50 Graphite HB2 Pencils +50 eraser caps +1 Sharpener
450
$19.48 ~~$24.00~~
FREE Delivery for Prime members
More buying choices
$18.97 (19 new offers)

Bestseller

AmazonBasics Pre-sharpened Wood Cased #2 HB Pencils, 150 Pack
43,227
$17.09 ($0.11/count)
$16.24 with Subscribe & Save discount
prime FREE delivery by **Tuesday, Mar 22**


Mansfield, Ontario $183 CAD / night
77 kilometres away Mar. 28 – Apr. 4

Lincoln, Ontario $113 CAD / night
60 kilometres away Apr. 29 – May 6

Horning's Mills, Ont... $132 CAD / night
81 kilometres away Apr. 10 – 17

Creemore, Ontario $328 CAD / night
93 kilometres away Sep. 26 – Oct. 3

There are lots of resources you can use to learn more about creating a card, for example:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Layout_cookbook/Card
- https://www.w3schools.com/howto/howto_css_cards.asp
- https://www.freecodecamp.org/news/learn-css-basics-by-building-a-card-component/

Update Your Store to Use Cards

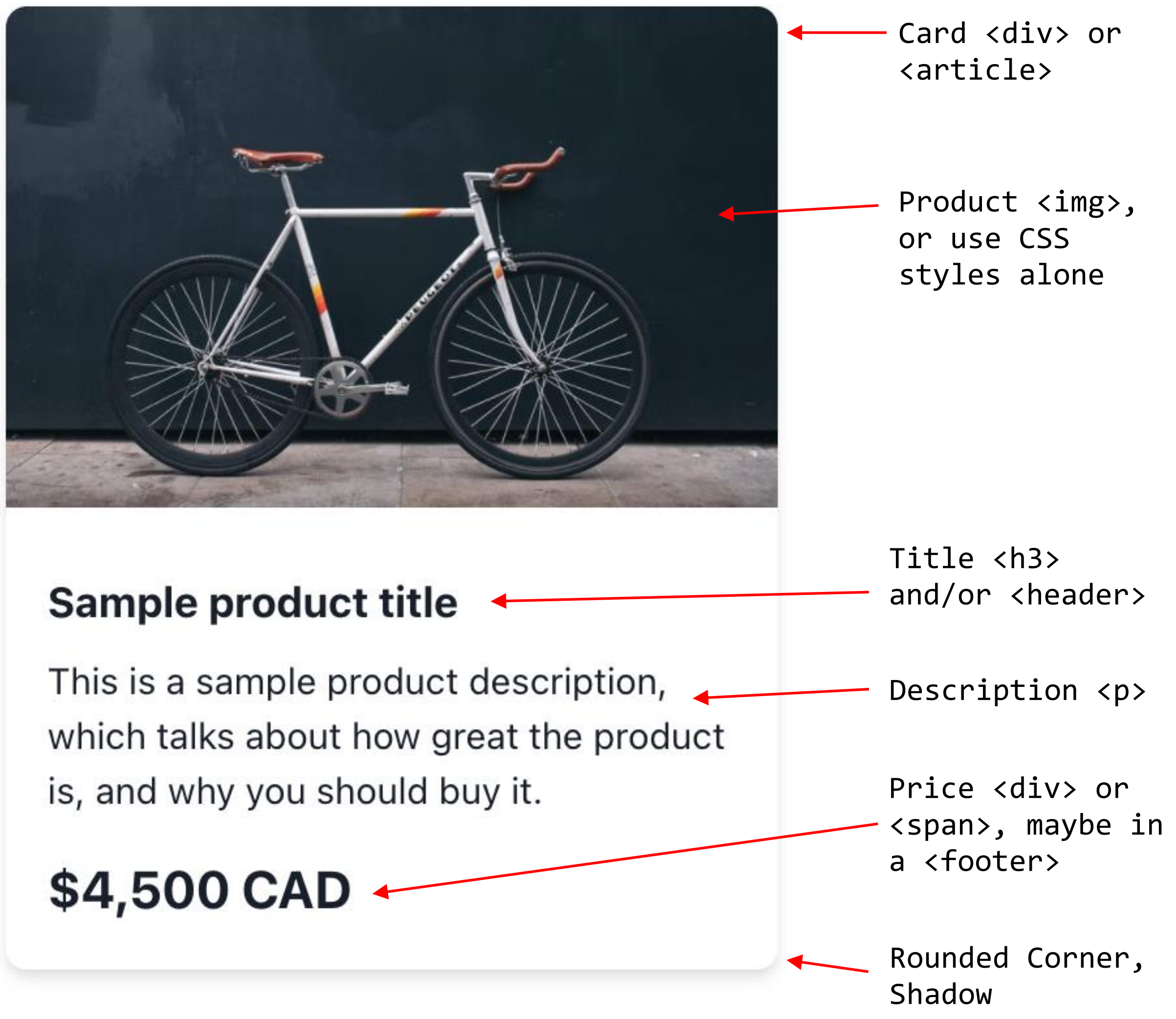
Modify your solution to Assignment 4 in order to **replace the HTML table with rows of cards**. To do this, you should follow these steps:

1. Create a copy of your Assignment 4 project, so you don't lose your previous work.

2. Start simple. In your HTML file, create a single product card (i.e., a <div>) that includes an <img> of the product, a heading (e.g., <h2> or <h3>) for the name, a <p> for the

description, and a <span> for the price (you can modify the HTML elements you use, these are just suggestions).

Use CSS classes on your card’s elements in order to apply colours, fonts, margins, padding, borders, etc. until you have something that you like. Here’s an example, which uses rounded corners, a subtle shadow, different font sizes, and a large photo at the top.



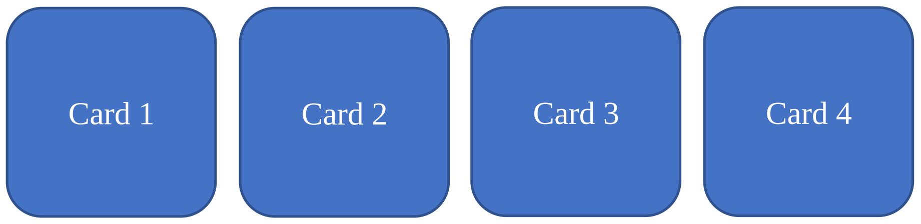
Sample product title

This is a sample product description, which talks about how great the product is, and why you should buy it.

$4,500 CAD

Card <div> or <article>
Product <img>, or use CSS styles alone
Title <h3> and/or <header>
Description <p>
Price <div> or <span>, maybe in a <footer>
Rounded Corner, Shadow

3. Create rows of cards. Use **Flexbox** or **CSS Grid** to create rows that repeat your cards across and down. For now, you can copy and paste your card from step 1 over and over in order to repeat it. Make your page look good with rows of 3 or 4 cards. Adjust the spacing, size, etc. until you’re happy with how it looks.



4. Find **at least** 3 product images to use in your cards. You don't have to find an image for every single product (i.e., you can repeat the same ones), but you should have at least 3 unique images.

   Make sure you optimize the images so they are not too big to download (i.e., don't use a 5000x6000 image in a card that uses 400x200).

   You can use https://squoosh.app/ for images that you download. Or you can also use a trick with https://unsplash.com/ images to resize them automatically via the URL. For example, the bike above is https://unsplash.com/photos/tG36rvCeqng. Here's the full-sized image https://images.unsplash.com/photo-1485965120184-e220f721d03e (it's 3.8M in size, and 4440x2960). We can reduce that image by adding some parameters to the image URL: **?auto=format&fit=crop&w=750&q=80** to crop and resize it to 750 pixels wide, and reduce the quality a bit to 80%, like this: https://images.unsplash.com/photo-1485965120184-e220f721d03e?auto=format&fit=crop&w=750&q=80 See https://unsplash.com/documentation#dynamically-resizable-images for more details.

5. Update your `src/products.js` file so that each Product Object has an `imageUrl` property. This imageUrl should be the URL or filename of an image to use for this product in your card. Multiple products can use the same image URL if you don't have enough unique images for each product.

6. Remove the card's HTML you wrote earlier and write a function in JavaScript that creates them dynamically instead. Your function should accept a product Object from your products array and create a card in the DOM, using the HTML and class names you wrote above. Here's some code to get you started:

```
function createProductCard(product) {
  // Create a <div> to hold the card
  const card = document.createElement('div');
  // Add the .card class to the <div>
  card.classList.add("card");

  // Create a product image, use the .card-image class
  const productImage = document.createElement('img');
  productImage.src = product.imageUrl;
  productImage.classList.add("card-image");
  card.appendChild(productImage);

  // ... rest of your card building code here

  // Return the card's <div> element to the caller
  return card;
}
```

7. Modify your page load and button click events so that instead of creating table rows for each product, you call your `createProductCard()` function and get back the card's <div>…</div> element, which you can now place in the DOM (e.g., using appendChild()). Clicking your category buttons should show the correct product cards on the page.

Task 4. Static Hosting

You are asked to research and implement a static hosting solution for your web site, so that it is accessible via a public URL. You do not need to spend any money to achieve this, since many free hosting services exist:

1. Netlify - https://www.netlify.com/
2. Vercel - https://vercel.com/

Please submit the public URL for your project. All pages, images, etc. must work and not return 404s or other errors.

Bonus :

- **(additional 1~3%):** use your creativity to add some bootstrap flavor + additional functionality to the site to present dynamic data ☺

Project Submission:

Add the following declaration at the top of .js/.html files

```
/******************************************************************************
*  WEB222 – Project
*  I declare that this assignment is my own work in accordance with SenecaAcademic Policy.
*  No part of this assignment has been copied manually or electronically from any other source
*  (including web sites) or distributed to other students.
*
*  Group member Name: _____________ Student IDs: __________ Date: _____________________
******************************************************************************/
```

- Compress (.zip) the files in your Visual Studio working directory (this is the folder that you opened in Visual Studio to create your client side code).
- Complete & Submit the project document (given template).
- Record a detailed walk-through/demonstration video presentation of the project

Important Note:

- 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.
- **LATE SUBMISSIONS for assignments**. There is a deduction of 15% for Late submissions, and after two days it will grade of zero (0).
- Assignments should be submitted along with a video-recording which contains a detailed walkthrough of solution. Without recording, the assignment can get the maximum of 1/3 of the total.