WEB222 Assignment 4

Overview

This assignment is designed to have you practice working with HTML and the DOM in order to create both *static* and *dynamic* web content.

You are asked to prototype a fictional online store. Your store will sell several different product 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.

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.

Pick Your Store and Product 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 3 categories**. Each product must belong to one or more of these categories.

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.

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

Products

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.

- **title**: 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.

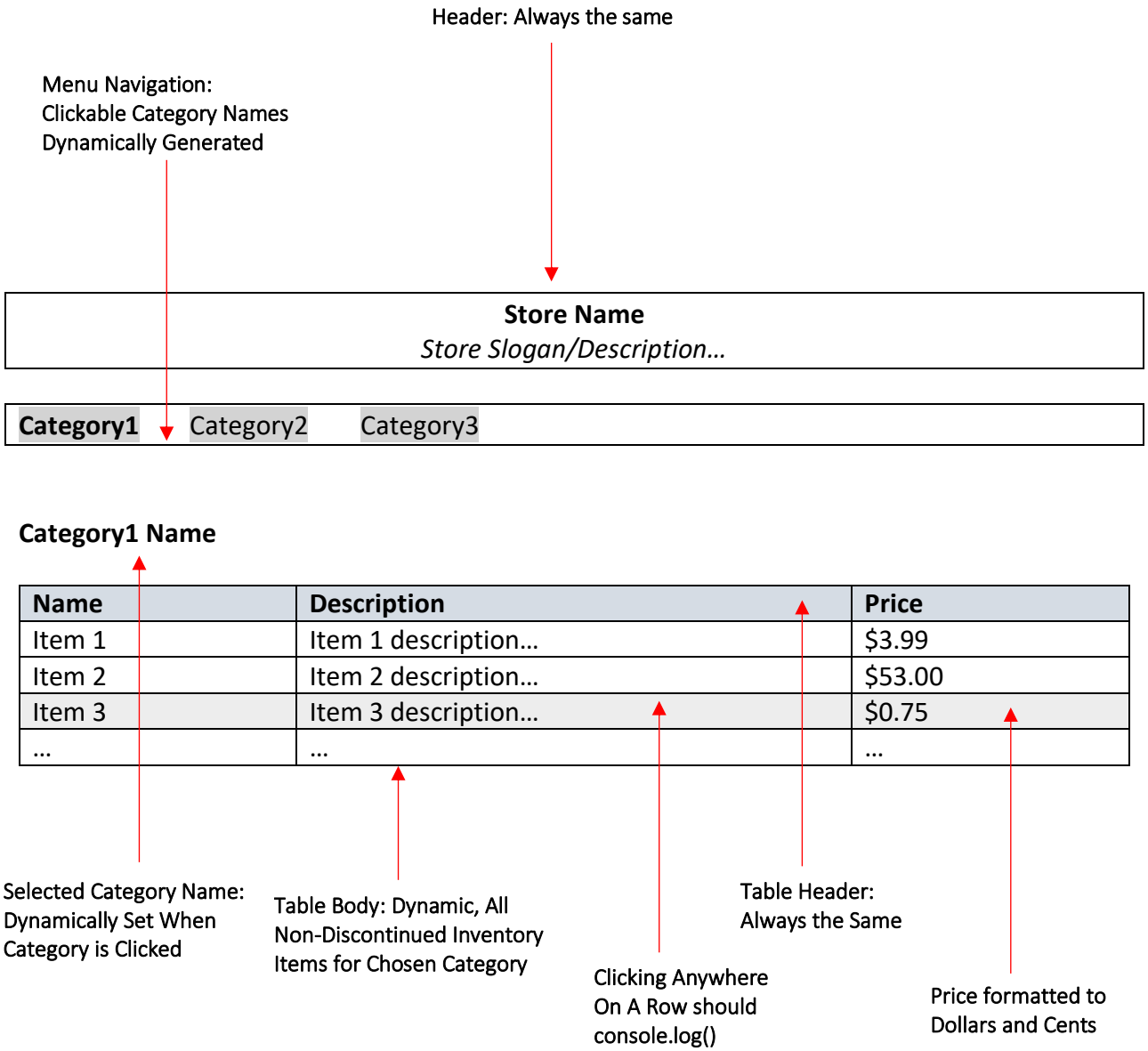
Take some time now to enter all of your store’s data.

Store Web Site HTML

Your store’s HTML file is located in `src/index.html`. A brief HTML skeleton has been created, and you are asked to fill in the rest using your information above.

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, and which parts are static or dynamic. NOTE: don't worry too much about how it looks. Focus on the structure and functionality.



Dynamic Content

All of your store's dynamic content will be written in JavaScript in the `src/app.js` file. Here is a list of the tasks you need to complete:

1. Create an event handler to run when the page is loaded. Your function should do the following:
    a. Create all of the buttons for your store's Categories
        i. Loop through all of your category objects and create a <button> element for each, adding it to the <nav id="menu">…</nav>
        ii. Use each Category's name for the button's text
        iii. When the button is clicked, show that category's products in your <tbody>…</tbody>. See below for more details.
    b. Show a list of products in the <tbody>…</tbody> of your Table. By default, you should use your first category. See below for more details
2. Write a function that will show a product list in the <tbody>…</tbody> based on a category Object:
    a. Update the text of the Selected Category Title above your table with the category's name
    b. Clear the current <tr>…</tr> rows from the <tbody>…</tbody>. HINT: innerHTML = ""
    c. Filter your products Array (i.e., use Array.prototype.filter()) to get:
        i. All products in the chosen category. HINT: use Array.prototype.includes()
        ii. All products that are NOT discontinued
    d. Loop (use Array.prototype.forEach()) over your filtered product list and add them to the table's body:
        i. Create a <tr> element
            1. Add a click handler to your <tr> that will console.log() the product whenever the user clicks it
        ii. Create <td> elements for the product's title, description, and price
            1. Convert the price in cents to dollars and cents
            2. Format it as Canadian Currency (HINT: research Intl.NumberFormat() and currency formatting)
        iii. Append these <td> elements to the <tr>
        iv. Append this <tr> to the <tbody>

In your solution, you will likely require all of the following:

- console.log() and NOTE that you can log Objects like so: console.log({ object })
- document.querySelector() to find elements in the DOM
- document.createElement() to create new DOM elements
- node.appendChild() to add an element as a child of a DOM node
- element.innerHTML to modify the HTML content of an element. Don't overuse this!

You are encouraged to use what you learned in the first 3 assignments and write proper functions and semantic HTML.

Coding:

Use the website starter project in the assignment ZIP file. Install all dependencies by running the following command in the root of the assignment (e.g., in the same directory as package.json):

npm install

Your code should all be placed in the src/ directory.

Running a Web Server:

You can start a local web server to test your code in a browser by running the following command:

npm start

This will start a server on http://localhost:8080, which you can open in your web browser

To stop the server, use CTRL + C

Submission:

When you are finished, run the following command to create your submission ZIP file:

npm run prepare-submission

This will generate submission.zip, which you can hand in on Blackboard.