Assignment 4 – Update Detail page based on URL params

Lab A & B due: Tuesday, October 04, 2022. 11:59 pm ET
Lab C & D due: Thursday, September 29, 2022. 11:59 pm ET
This is an **individual** assignment.

1 Learning objectives

- Further practice your JavaScript skills.
- Practice adapting your webpage dynamically based on user input.

2 Tasks

In this assignment, you will adapt the contents of the Detail page based on which cinnamon roll the user clicked on the Gallery page.

Here is a demonstration video of the functionality you should implement in this HW:
https://drive.google.com/open?id=1-wtClQEhcraay0XY8z81bwaAxe4GZPwT

You will be provided with a "database" of the price and image for each type of roll, in JSON format as follows.

```
const rolls = {
    "Original": {
        "basePrice": 2.49,
        "imageFile": "original-cinnamon-roll.jpg"
    },
    "Apple": {
        "basePrice": 3.49,
        "imageFile": "apple-cinnamon-roll.jpg"
    },
    ...
};
```

The above content is stored in a file `rollsData.js` that you can find here:
https://github.com/interactive-structures/teach-pui/blob/main/homework-materials/HW4/rollsData.js

Download this file and add it to **your own** solution directory as a copy.

2.1 Update Gallery page

(1) Update all links on the Gallery page so that they point to the Detail page with the corresponding roll parameter. The roll type should be the *keys* in your `rolls` object copied above.
For example, when a user clicks on the walnut cinnamon roll on the Gallery page, they should be redirected to a URL such as

`detail.html?roll=Walnut`

(detail.html may be substituted for the name of your Detail page's HTML file).

2.2 Update Detail page

(2) Read the database from the file `rollsData.js` and to get access to the roll dictionary. You can do this by including it using an additional `<script>` tag in the HTML code that points to `rollsData.js`.

(3) Create an empty array called `cart`.

(4) Parse the URL parameter and store the current roll type as a variable. You can use the following code to get the roll type from a URL:

```
const queryString = window.location.search;
const params = new URLSearchParams(queryString);
const rollType = params.get('roll');
```

See example (from Lab 4) here:
https://github.com/interactive-structures/teach-pui/tree/main/in-lab-examples/puinote-lab04/url-params

(5) Extract the current roll's information (name, price, image path) from the dictionary in step (1) and update relevant DOM elements:

(a) The Heading should be <Type> Cinnamon Roll, e.g., 'Apple Cinnamon Roll', 'Walnut Cinnamon Roll'.

(b) The image on the product page should show the correct roll type. Note that the database only contains the image file name; you will need to specify the full path to this file (e.g., `images/products/apple-cinnamon-roll.jpg`), similar to how you specify the `img src` attributes on the Gallery page's HTML code.

(6) When the user selects an option from the glazing or pack size dropdown menu, update the total price field, similar to HW3. Remember to use the correct base price from the database.

2.3 Add to cart

Note: In this assignment, you will start preparing your shopping cart implementation. The contents of this cart will not be stored when you reload the page, we will do that in the next assignments.

(7) When the user clicks on "Add to Cart," save all of the current product information (roll type, glazing, pack size, base price) into an instance of the class Roll. You can use the following class definition:

```
class Roll {
    constructor(rollType, rollGlazing, packSize, basePrice) {
        this.type = rollType;
        this.glazing =  rollGlazing;
        this.size = packSize;
        this.basePrice = basePrice;
    }
}
```

(8) Then add this instance to the array `cart` created in Step 2. Print your entire cart array to the console after everytime you add items using `console.log()`.

Note: this final step does not result in any changes to the DOM (the current page should stay at Product after clicking on "Add to Cart"). We will check your JavaScript code to determine if you implement it correctly.

2.3 Use proper code style

Follow the code style instructions from previous assignments, i.e., external `js` files, no external libraries, follow code style guidelines.

3 Submission

(1) **Deploy** your webpage on GitHub.
Check that your code is online (**pushed**) & that your webpage works properly online.

(2) Submit your homework repository to **Gradescope**.
Verify that it was submitted to Gradescope before the deadline!