CPSC 349 - Web Front-End Engineering

Exercise 4

What are we trying to accomplish?

In this exercise you will use JavaScript to implement a fundamental interaction pattern for web applications: fetching data from a REST API and updating the DOM tree to display the data as HTML.

Steps to complete this exercise

Before you begin this exercise you should have completed the *Understanding the DOM — Document Object Model* eBook.

1. Download the code that comes with this exercise, then run `npm install` and `npm start`.

2. The text in `posts.html` comes from the JSONPlaceholder API.

   Examine the following URLs, and compare them to the text that you find in the HTML:

   - `https://jsonplaceholder.typicode.com/posts/1`
   - `https://jsonplaceholder.typicode.com/users/1`
   - `https://jsonplaceholder.typicode.com/posts/1/comments`

3. The current page contains static HTML, but we can retrieve data live from the API. Open the console and reload the page.

   You should see an array of 10 objects retrieved by the `downloadPosts()` function in `script.js`. Click the disclosure triangle to reveal the array's contents. Compare the contents of the array with `https://jsonplaceholder.typicode.com/posts?_page=1`.

   The goal of this exercise is to dynamically load the DOM with data fetched from an API.

4. Use the Document Object Model to add each downloaded post as an `<article>`.

   For each `<article>` tag:

   - Set the `data-post-id` attribute to the `id` field of the post.
   - Create an `<h2>` with the contents of the `title` field.

- Create an `<aside>` and fill in the `<span>` by calling `getUserName()` and passing the `userId` field.
- Create the `<p>` tag by replacing `"\n"` characters with `<br>` tags.

For each `<details>` tag:

- Including the `<summary>`, `<section>`, `<header>`, and `<h3>` tags, but do not add any `<aside>` tags.
- Add a `toggle` event listener to the `<details>` section. This is 90% done for you in my provided code.

5. Remove the static `<article>` and `<details>` tags from `posts.html`, and demonstrate that all 10 articles are created dynamically when the page loads.
6. Returning to the console, if your `<details>` sections and `toggle` event listeners were added correctly, you should see that clicking “See what our readers had to say…” for each article should download and log the comments for the correct `postId`.
7. Add each comment as an `<aside>` tag using the `body` and `name` fields for the two paragraphs. You will need to replace `"\n"` characters with `<br>` tags again.
8. Finally, check that everything continues to work when you change

   ```
   downloadPosts()
   ```

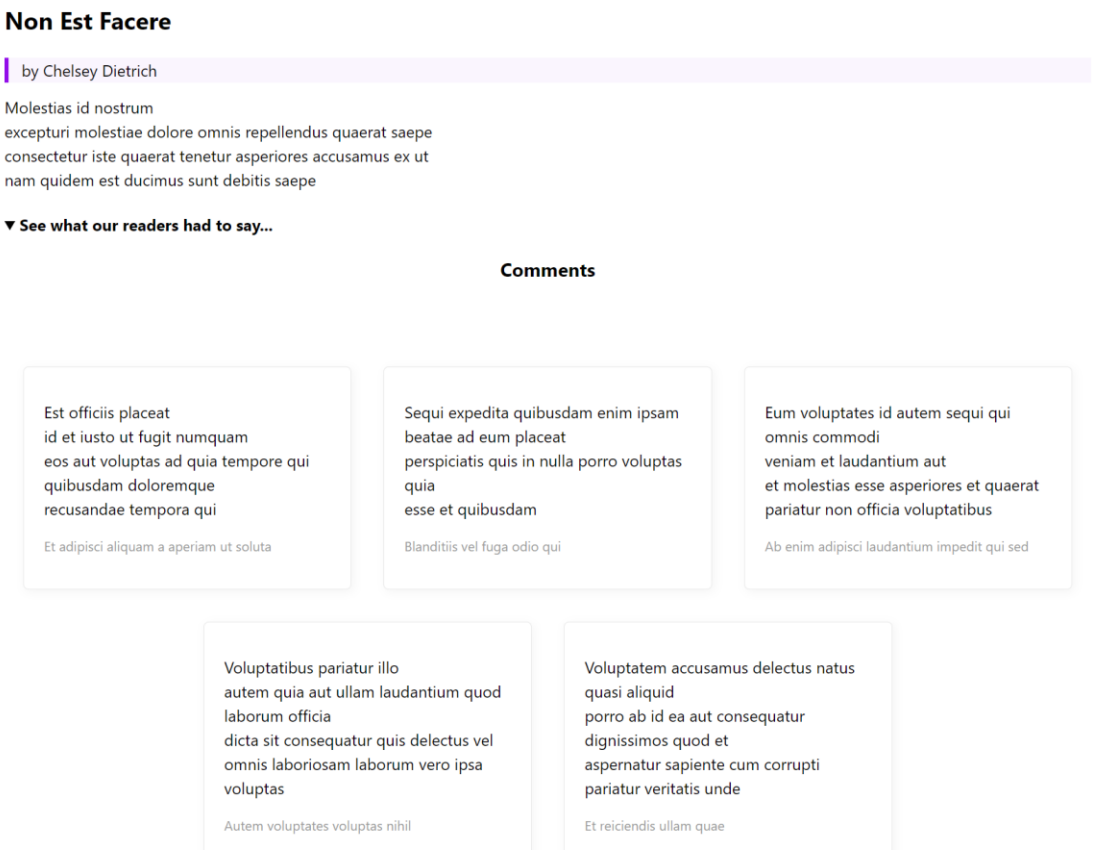
   in `script.js` to

   ```
   downloadPosts(2)
   ```

   to see the second page. Check that the articles, user names, and comments are downloaded and populated correctly.
9. Zip and upload your posts.html and script.js to Canvas by the deadline.

Example

The following screenshot is the first article of the posts on the 5th page. That is, if you run your site with downloadPosts(5) you should see exactly



Hints

- Get the <main> tag and then manipulate the innerHTML
- Use replaceAll() to replace “\n” with
- You’ll want to select the <section> tag to set the innerHTML for <aside>
- JavaScript scripts are read from top to bottom. If you add event listeners to elements that have not yet exist, these events will not be attached.
- Think smart, don’t think hard. Everything you need to complete this assignment should only be done on the posts.html and script.js.