WDD 131

JavaScript: DOM Manipulation

Overview

One of the key skill sets of any frontend web developer is the ability to manipulate the document dynamically using the [DOM](#), a JavaScript object that is built after the browser initially builds the document. To manipulate the DOM means to read, edit, update, or delete the document, including CSS properties, dynamically. The DOM is the outline of the HTML and content nodes.

The purpose of this activity is to introduce the HTML DOM and to learn how to change or manipulate the document using JavaScript.

> ▶ Associated Course Learning Outcomes

Prepare

> [The JavaScript DOM explained in 5 minutes!](#) - Bro Code

- Reference: [Manipulating documents](#) - MDN.

  > Note that the [Active learning: A dynamic shopping list](#) example found at the end of the MDN article will **help** you complete this week's activities.

- Practice: **Some common DOM manipulation concepts/use cases**.

  - To **select** an HTML element from the document - using the [querySelector](#) method.
    This line of code selects the first instance of an article element from the document and assigns the element as a reference to the variable named **article**.

    ```
    const article = document.querySelector('article');
    ```

- To **change** the innerHTML property of an existing element.
  This line of code uses an existing variable that has already selected an element and changes its innerHTML property value.

  ```
  article.innerHTML = 'innerHTML supports <strong>HTML</strong> tags. Th
  ```

- To **change** the inline CSS style of an element.
  This line of code changes the text-align CSS property of the selected element.

  ```
  article.style.textAlign = 'right';
  ```

- To **change** an attribute of an element.
  This line of code adds and attribute class to the element and gives it a value.

  ```
  article.setAttribute('class', 'active');
  ```

  > An alternative way to change an attribute of an element, specifically the class attribute, is by directly manipulating the element's classList.
  >
  > ```
  > articleElement.classList.add('active');
  > ```
  >
  > This method is often preferred when working with class manipulation because classList provides additional methods like add, remove, toggle, and contains, making it more convenient and expressive for managing classes on an element.

- To **create** an element.
  This line of code creates a new <p> element and stores it in the variable named paragraph.

  ```
  const paragraph = document.createElement('p');
  ```

- To **append** an element with additional content or elements.
  These lines of code add content to the end of the article element.

  ```
  article.appendChild(paragraph);
  article.append(paragraph, 'Hello World Addition!');
  ```

The append() method allows you to include multiple arguments to append to the element in the specified order.

- To **remove** an element from the document.
  These lines of code remove the paragraph from the article and then, the article itself.

  ```
  article.removeChild(paragraph);
  article.remove();
  ```
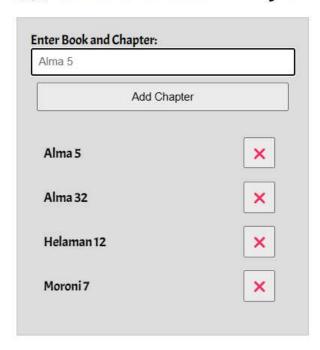
Activity Instructions

This Book of Mormon application will be added upon in future learning activities. We start now with the interface and basic DOM assignments and build.

This app allows users to enter their favorite Book of Mormon chapters and display them on a list that is updated automatically on the screen. Entries can then be deleted from the displayed list.

1. Create an HTML file named "**bom.html**" in the **week02** folder.

2. Your **bom.html** HTML document should include the basic meta tags and an appropriate title.

3. Create an external CSS file and a JS file and place them in appropriate sub-folders within the week02 folder.

> ▶ Check Your Understanding

4. Copy and paste the basic interface (the HTML and CSS) from the following CodePen [BOM Top 10](#).

5. In your blank js file, declare three (3) variables that hold references to the **input**, **button**, and **list** elements.

▶ Check Your Understanding

6. Create a `li` element that will hold each entries chapter title and an associated delete button.

▶ Check Your Understanding

7. Create a delete `button`.

▶ Check Your Understanding

8. Populate the `li` element variable's `textContent` or `innerHTML` with the input value.

▶ Check Your Understanding

9. Populate the button textContent with a ✖.

▶ Check Your Understanding

10. Append the li element variable with the delete button.

▶ Check Your Understanding

11. Append the li element variable to the unordered list in your HTML.

▶ Check Your Understanding

So far, we have just setup the interface and completed some basic DOM interaction. The application **will not work at this point**. The next activity will teach how to respond to events, like button clicks. We need to wait for the user to make an entry into the favorite chapter text field before processing.

Submission

- Continuously save your work. We will add additional functionality in the next activity.

Optional Resources

- [DOM Manipulation](#) – by Web Dev Simplified – 14 key techniques in summary demonstration format. Take special note of the concept of **datasets**.

- We do need to consider screen readers and how they will interpret anything that we have in the content. For example, the delete button just has an emoticon and may not read correctly as the button to remove a chapter. What can we do? One solution is to create a [aria-label attribute](#) on the button with a value like "Remove Alma 5"