W02 Learning Activity: Handling DOM Events

Overview

Handling DOM events in JavaScript is a fundamental part of building interactive web pages. In this activity, you'll learn all about the different events you can handle, how to handle them, and how to pass data between the event handler and the rest of your code.

> "Events are things that happen in the system you are programming – the system produces (or "fires") a signal of some kind when an event occurs, and provides a mechanism by which an action can be automatically taken (that is, some code running) when the event occurs. Events are fired inside the browser window, and tend to be attached to a specific item that resides in it. " – MDN

Prepare

1. Reference: Introduction to Events – MDN
2. Practice: **Some common DOM event concepts/use cases**. These are just a few examples, and there are many other events and use cases.
    - **Click Event**: Triggered when a user clicks on an element. It's widely used for handling button clicks, link clicks, or any interaction involving a mouse click.

```
buttonElement.addEventListener('click', function() {
  // Code to execute when the element is clicked
});
```

> The `addEventListener` method takes two arguments: the event name and a function to execute when the event is triggered. The function is called an event handler. We will learn more about writing functions later in the course. For now, just know that the function is a block of code that will be executed when the event is triggered. The function is also called a callback function because it is called back when the event is triggered.

- **Keyup Event**: Triggered when a key is released. Useful for handling keyboard input.

```
buttonElement.addEventListener('keyup', function() {
  // Code to execute when a key is released
});
```

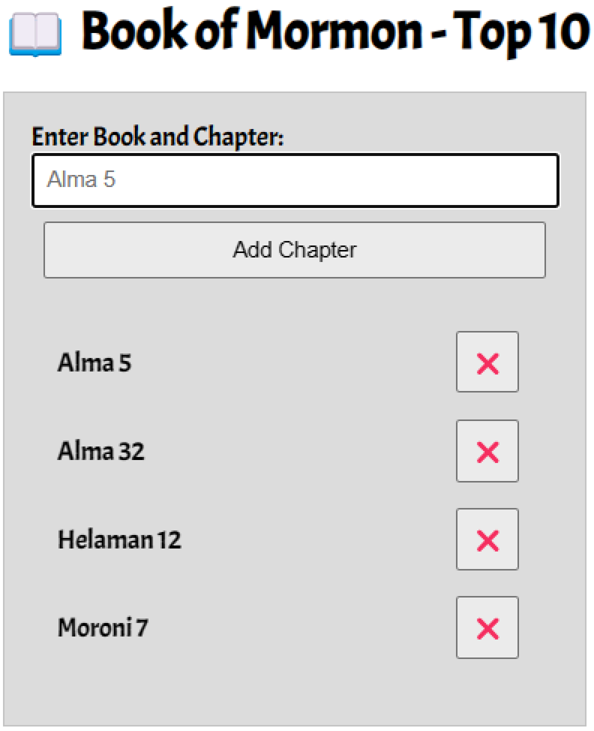
- **DOMContentLoaded Event**: Triggered when the HTML document has been completely parsed and all deferred scripts have been executed. It's widely used for initializing JavaScript applications.

```
document.addEventListener("DOMContentLoaded", function() {
  // Code to execute when the DOM is parsed
});
```

Activity Instructions

This activity adds functionality to the BOM Top 10 application started in the previous learning activity on the DOM. We will organize the code to respond to the **Add Chapter** button click in order to build the list item. In addition, the delete button functionality will be added along with some user interface expectations.

1. Open up your JavaScript file that is supporting the **bom.html** application if needed.
2. Create a **click** event listener for the **Add Chapter** button using an **addEventListener**.



📖 Book of Mormon - Top 10

Enter Book and Chapter:

Alma 5

Alma 5

Alma 32

Helaman 12

Moroni 7

▶ Check Your Understanding

3. Within the **Add Chapter** button click event function block, between the opening and closing of the callback function argument { … }, do the following:

✓ Check to make sure the input is not blank before completing the remaining tasks in this list by using an `if` block that provides a message or does nothing and returns the `.focus()` to the input field.

▶ Check Your Understanding

✓ Move the code that you wrote in the last activity into this `if` block in order to support the correct flow of the program.

▶ Check Your Understanding

✓ Add an event listener to the delete button that removes the li element when clicked.

▶ Check Your Understanding

✓ Change the input value to nothing or the empty string to clean up the interface for the user.

▶ Check Your Understanding

✓ Whether or not a list item was created, the focus (active cursor) should be sent to the input element.

▶ Check Your Understanding

Testing & Submission

- Thoroughly test your application in the browser. This can be done locally.
- What else would you add to increase this applications usability?
- Commit and push your work to your GitHub Pages enabled wdd131 repository.

Week 02 Home