Starter Guide – Frontend

Phase 2 - Doc 3

Jana Karra

January 17, 2023

Interphase Dev Team

Introduction

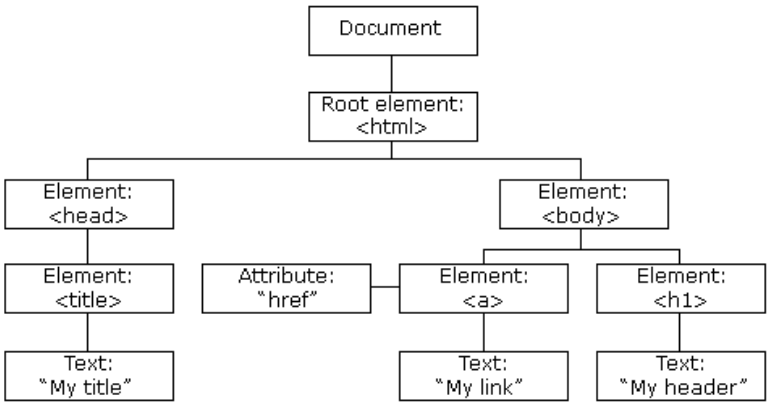
The purpose of this document is to discuss JavaScript DOM Manipulation, the DOM documents and elements, and the events and event listener. The document ends with an assignment that must be completed.

DOM Manipulation

The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.

Ref: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction

When a web page is loaded, the browser creates a Document Object Model of the page. The HTML DOM model is constructed as a tree of Objects:



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page

- JavaScript can create new HTML events in the page.

Ref: https://www.w3schools.com/js/js_htmldom.asp

[25] https://www.w3schools.com/js/js_htmldom.asp

DOM Document and Elements

The HTML DOM document object is the owner of all other objects in your web page.

The document object represents your web page. If you want to access any element in an HTML page, you always start with accessing the document object.

Some examples of how you can use the document object to access and manipulate HTML are:

- Finding HTML Elements
- Changing HTML Elements
- Adding and Deleting Elements
- Adding Events Handlers
- Finding HTML Objects

Ref: https://www.w3schools.com/js/js_htmldom_document.asp;
https://www.w3schools.com/js/js_htmldom_elements.asp

[26] https://www.w3schools.com/js/js_htmldom_document.asp

[27] https://www.w3schools.com/js/js_htmldom_elements.asp

DOM Events and Event Listener

JavaScript can be executed when an **event** occurs, like when a user clicks on an HTML element.

Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted

- When a user strokes a key

An **event listener** is a function that initiates a predefined process if a specific event occurs. So, an event listener "listens" for an action, then calls a function that performs a related task. This event can take one of many forms. Common examples include mouse events, keyboard events, and window events.

JavaScript has the **addEventListener()** method that attaches an event handler to an element without overwriting existing event handlers. You can add many event handlers to one element.

You can add event listeners to any DOM object, not only HTML elements. i.e., the window object.

You can easily remove an event listener by using the **removeEventListener()** method.

Ref: https://www.w3schools.com/js/js_htmldom_events.asp;
https://www.w3schools.com/js/js_htmldom_eventlistener.asp

[28] https://www.w3schools.com/js/js_htmldom_events.asp

[29] https://www.w3schools.com/js/js_htmldom_eventlistener.asp

Assignment

Create a program that simulates a simple to-do list where the user can add a task, edit a task, mark a task as done, and delete a task.
Instructions:

1. Create an HTML file with a form that has a single text input and a submit button. The form should be used to add new items to the list.
2. In the JavaScript file, use the DOM API to select the form and add an event listener that will listen for the submitted event.
3. When the form is submitted, use the DOM API to create a new list item element and add it to the existing list.
4. The task should be added to the list asynchronously, which will ensure that it appears on the page as soon as it has been added.

5. The new list item element should contain the text entered in the form as well as 3 buttons, one for marking the task as done, another for editing the task, and the other is for deleting the task.
6. In JavaScript, create event listeners for the "done", "edit" and "delete" buttons, which will update the task in the array and re-display the list on the page with the updated task.
7. Whenever a task is marked as done, it should be crossed out and not deleted.
8. You should have a delete all button which triggers a function that loops on all crossed out tasks and deletes them.
9. Create a feature that filters the tasks based on their status.
10. Add the ability to search for a specific term, which is highlighted in the tasks when it is searched for.
11. Use LocalStorage to persist the list of tasks even when the page is refreshed.
12. Test your application to ensure that items can be added and removed from the list as expected.
13. Add some CSS to make your application look better.