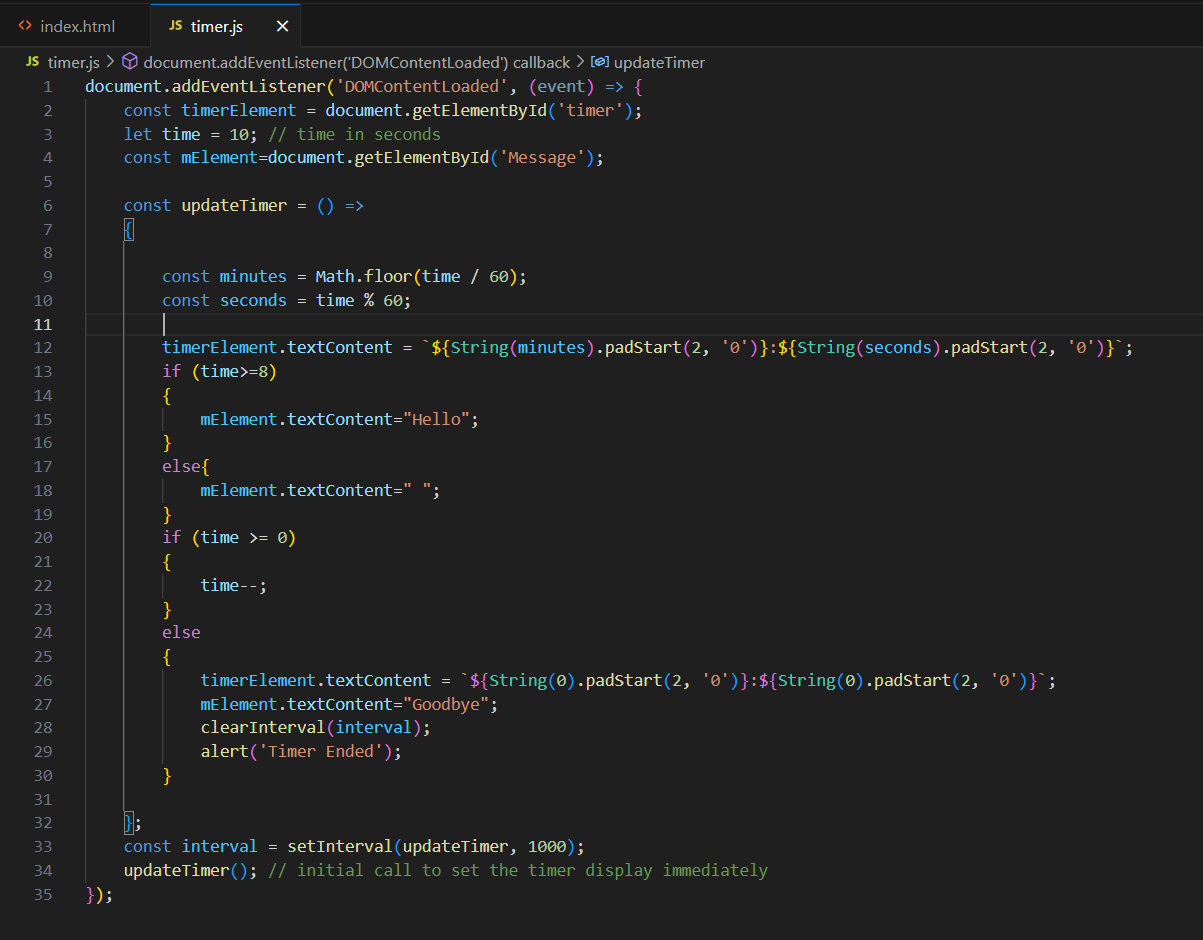
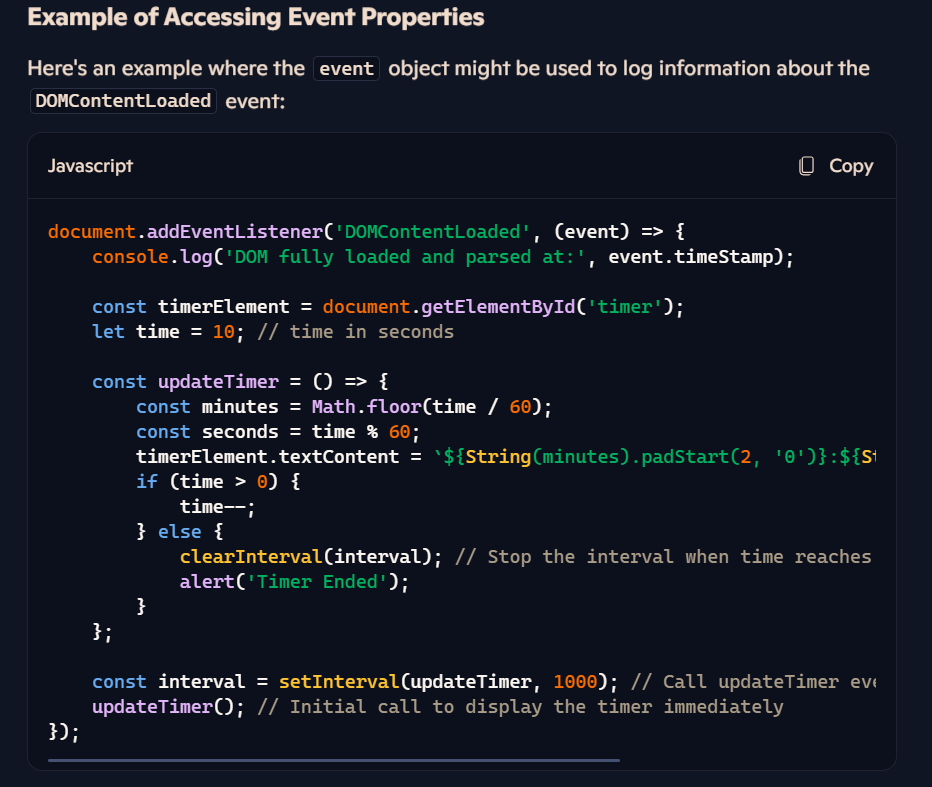
TO DO APP

TIMER:

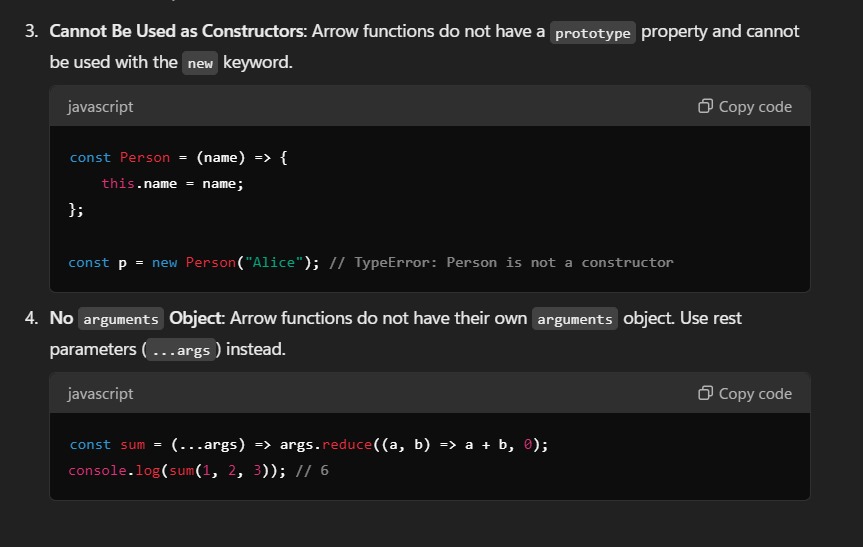


* document: is an object to call addEventListener().
* ‘DOMContentLoaded’ : Ensures all the html elements are successfully loaded before the .js file runs.
* (event) : is the event handler function that excutes after the DOMContentLoaded is fired. The event object is a parameter automatically passed to the event handler function when an event occurs. It contains information about the event and provides methods to interact with it. In this case, the event parameter refers to the DOMContentLoaded event, which signifies that the initial HTML document has been completely loaded and parsed.

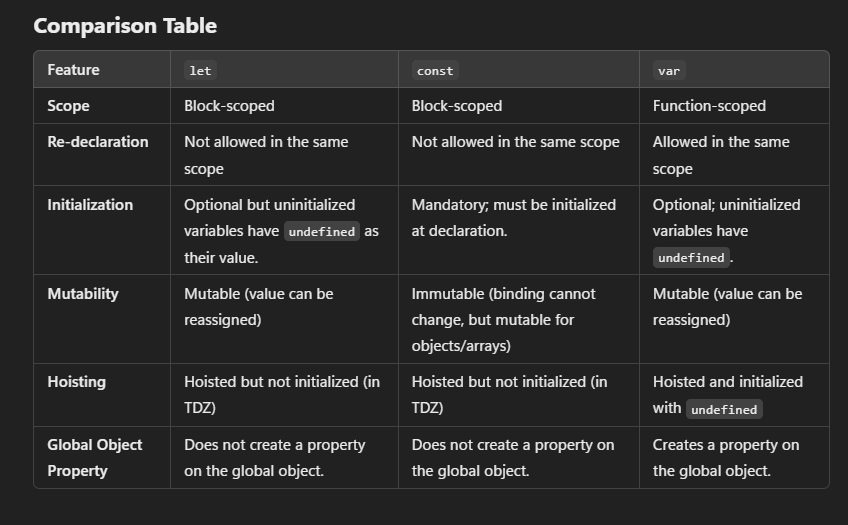


event.timeStamp: This property gives the timestamp of when the event was created. It can be useful for debugging or performance monitoring.

* addEventListener() : The addEventListener method attaches the event handler to the specified element. Here, it's used to attach a handler to the html document using document. It allows you to specify the type of event to listen for and the function to execute when the event occurs.
* ‘=>’ : The arrow function is used to declare a function. It is basically needed here as they do not need there own parameter and use the rest parameters(…args).



* const : it is block scoped and used to make a variable constant. In our program, the const minute and second is being updated religiously as it is being declared and initialized in a function, so it is local to it and can be updated in each function call.
* let : it is block scoped and allows us to change the value unlike const.(var is function scoped and is better not to use it)



* const interval = setInterval(updateTimer, 1000); : this statement calls the *updateTimer* function after every *1000 milisecond* (1 sec) delay so that our timer purpose is solved.
* updateTimer() [line 34] : Initial call to set the timer display immediately.
* getElementById('timer') : is used to get the ‘timer’ element from the index.html file for timer template. Syntax:

*getElementById(‘ <id\_name> ’)*

* textContent : The textContent property in JavaScript is used to get or set the text content of an element and its descendants. It provides a straightforward way to work with the text inside HTML elements without dealing with HTML tags or attributes. Syntax:

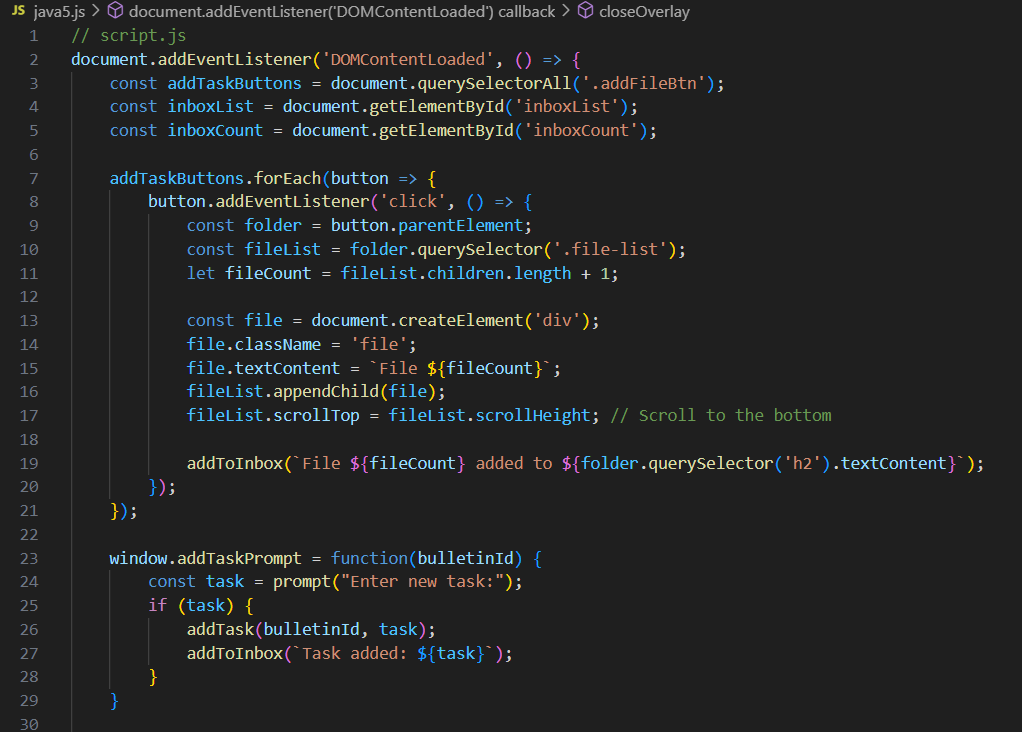
*<element\_variable\_name>.textcontent>*

* **Getting Text Content**: When used without an assignment, it retrieves the text content of the specified element.
* **Setting Text Content**: When used with an assignment, it sets the text content of the specified element.
* String(minutes) : This is type casting minutes(int variable) to string.
* padStart(2,’0’) : Makes 2 spaces and initializes them to 0.
* timerElement.textContent = `${String(minutes).padStart(2, '0')}:${String(seconds).padStart(2, '0')}`;

This whole statement prints the current timer seconds in the format MM:SS.

* mElement.textContent="Hello"; : Prints Hello for 2 seconds, then for the remaining time else is executed which prints space, lastly in the the good bye message is display.
* clearInterval(interval); : This is used to terminate the continuous function calls done by setInterval().
* alert(‘Timer Ended’); : this brings a pop up on the website to display Timer Ended.

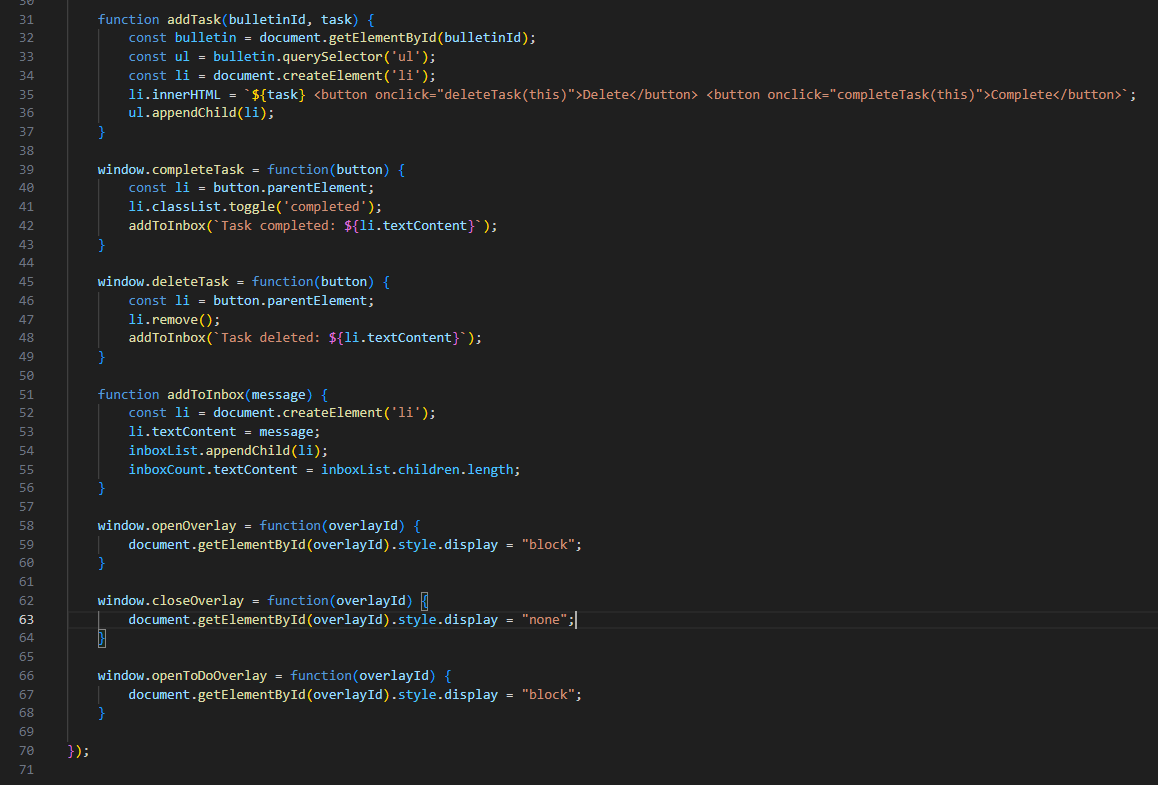
ADD FILE, OPEN & CLOSE MODALS, DELETE TO DOs:



* querySelectorAll(‘inboxCount’) : It is a method in JavaScript that allows you to select all elements in the document that match a specified CSS selector. The querySelectorAll method returns a static NodeList of all elements in the document that match the specified group of selectors with the name of ‘addTaskButtons’, in this case. The returned NodeList is not live, meaning it does not automatically update if the document changes. Syntax:

*querySelectorAll(‘ <class\_name> ’)*

* addTaskButtons.forEach(): This loops through each button in the addTaskButtons NodeList.
* const folder = button.parentElement; : This selects the parent element of the button that was clicked. This is assumed to be the "folder" where the files will be added.
* const fileList = folder.querySelector('.file-list');: This selects the first child element within the folder that has the class file-list.
* let fileCount = fileList.children.length + 1;: This calculates the number of existing child elements within fileList and adds 1 to get the count for the new file.
* const file = document.createElement('div');: This creates a new div element.
* file.className = 'file';: This sets the class of the new div to file.
* fileList.appendChild(file);: This appends the new div element to the fileList.
* fileList.scrollTop = fileList.scrollHeight;: This ensures that the file list scrolls to the bottom to show the newly added file.



* innerhtml : Allows you to write on the html page using html tage along with javascript.
* ul.appendChild(li) : To add li to ul.
* classList : The classList property of an HTML element (in this case, the <li>) provides access to the list of classes assigned to that element.
* toggle(‘completed’) : The toggle method is used to add or remove a class from an element's class list. If the class is already present, toggle will remove it. If the class is not present, toggle will add it.