Workshop 02: JavaScript

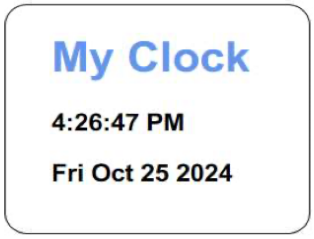
Question 3: Write a JavaScript program to create a digital clock that displays the current time and date in the following format:

- **Time**: hh:mm:ss AM/PM
- **Date**: Displayed in the format of the toDateString() method.

The time should update every second.

Requirements:

1. Define a function displayClock() that:
   - Retrieves the current date and time.
   - Formats the time in 12-hour format with an AM/PM suffix.
   - Pads single-digit minutes and seconds with a leading zero.
   - Updates elements with IDs time and date to display the formatted time and date.
2. Set an interval to call displayClock() every second to keep the display updated.



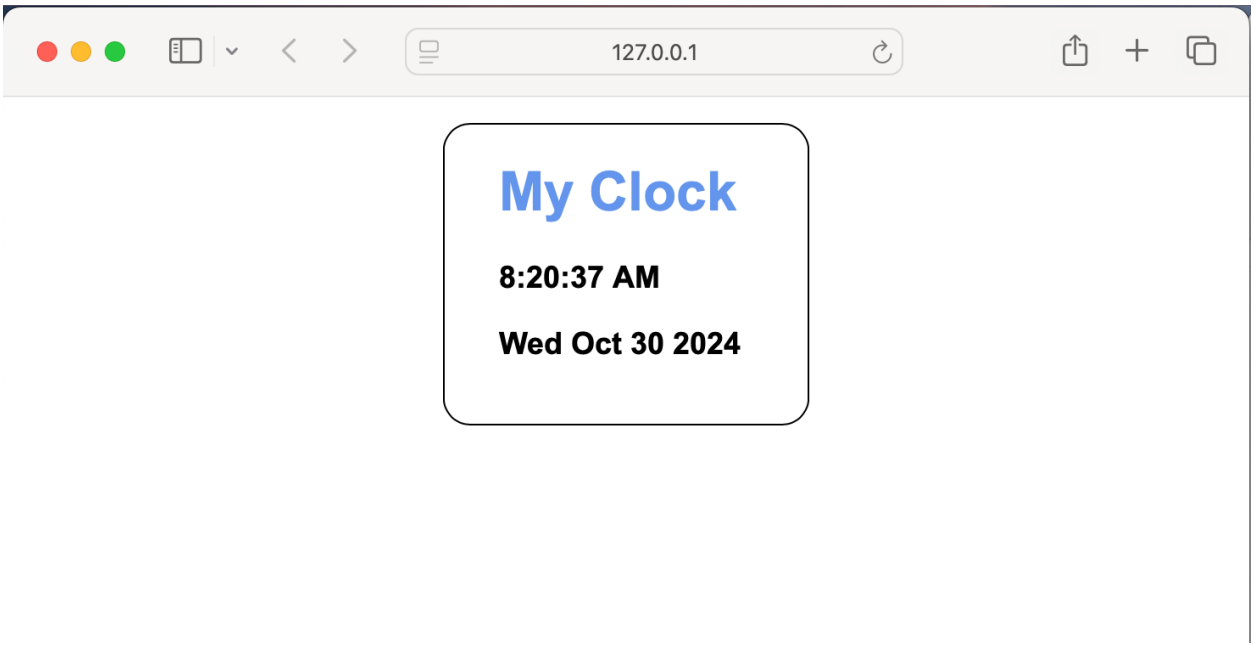
Question_3.html

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <title>Digital Clock</title>
  <link rel="stylesheet" href="Question_3.css" />
 </head>
 <body>
  <h1>My Clock</h1>
  <div id="clock">
   <div id="time"></div>
   <div id="date"></div>
  </div>
  <script src="Question_3.js"></script>
 </body>
</html>
```

Question_3.js

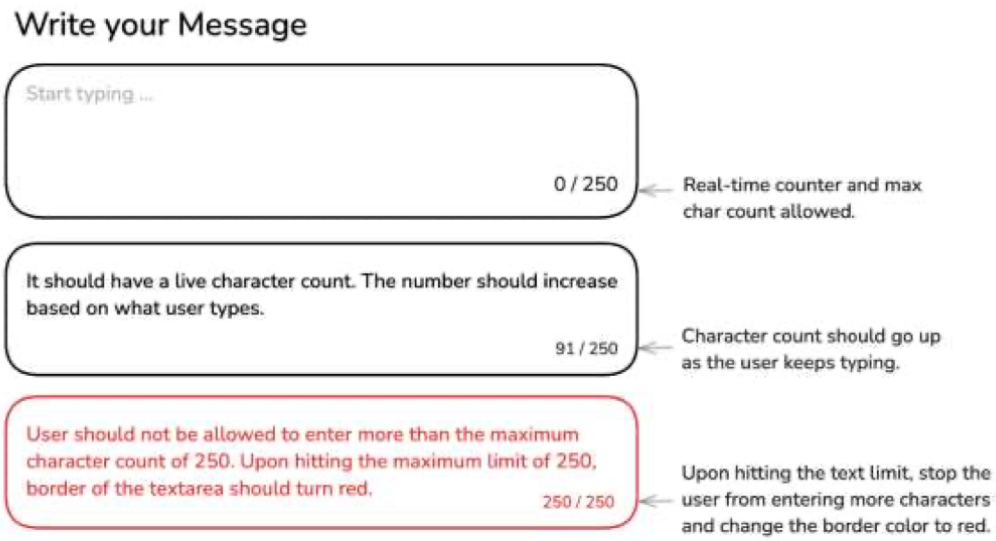
```
function displayClock() {
  const now = new Date();

  // Format time with AM/PM
  let hours = now.getHours();
  const minutes = String(now.getMinutes()).padStart(2, "0");
  const seconds = String(now.getSeconds()).padStart(2, "0");
  const ampm = hours >= 12 ? "PM" : "AM";
  hours = hours % 12 || 12; // Convert to 12-hour format and handle
midnight (0 hour)

  const timeString = ` ${hours}:${minutes}:${seconds} ${ampm}` ;
  const dateString = now.toDateString();

  // Update elements with IDs 'time' and 'date'
  document.getElementById("time").textContent = timeString;
  document.getElementById("date").textContent = dateString;
}

// Call displayClock every second
setInterval(displayClock, 1000);
displayClock();
```

Output:



Github Deployment: ***https://nambobby.github.io/Web-Lab-UWE-IU/Lab_2/Workshop_02/Question_3/Question_3.html***

Question 4 : You are required to create a textarea that tracks and displays the number of characters typed by the user, along with a maximum character limit. As the user types, the character count will update dynamically. Once the limit is reached, the textarea will prevent further input and the border will turn red to visually indicate the limit has been hit.



Question_4.html

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Counter Textarea</title>
  <link rel="stylesheet" href="Question_4.css" />
 </head>
 <body>
  <h1>Write your Message</h1>
  <div id="container">
   <textarea
```

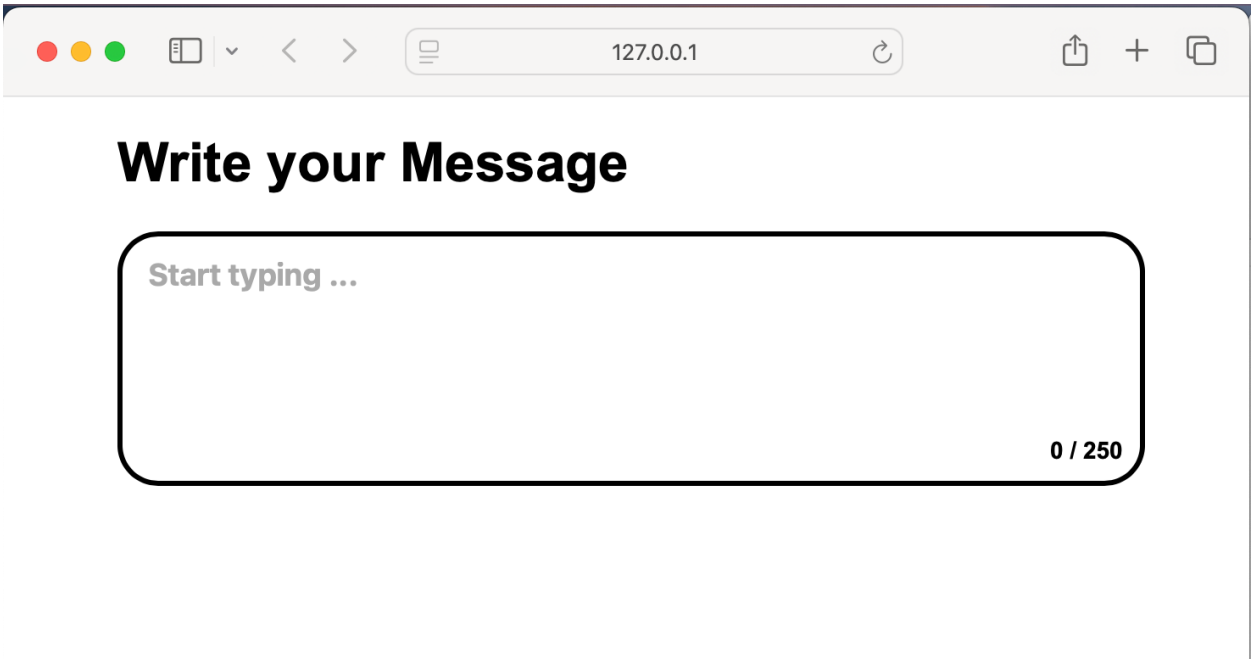
```
      id="message"
      placeholder="Start typing ..."
      maxlength="250"></textarea>
    <div id="charCount">0 / 250</div>
  </div>
  <script src="Question_4.js"></script>
 </body>
</html>
```

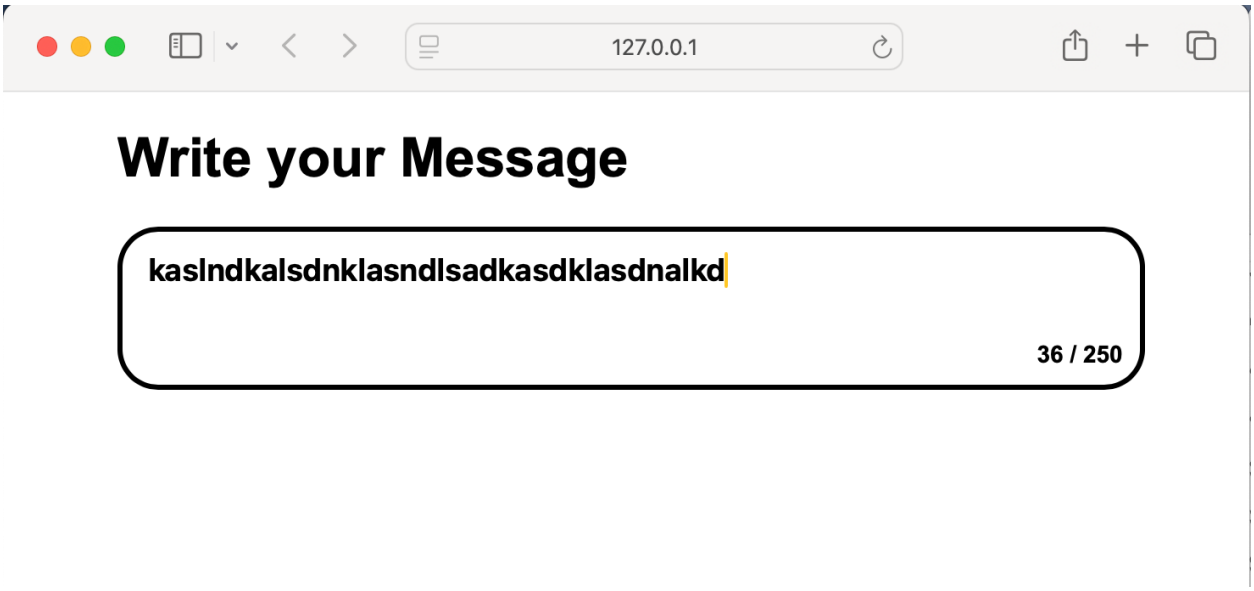
Question_4.js

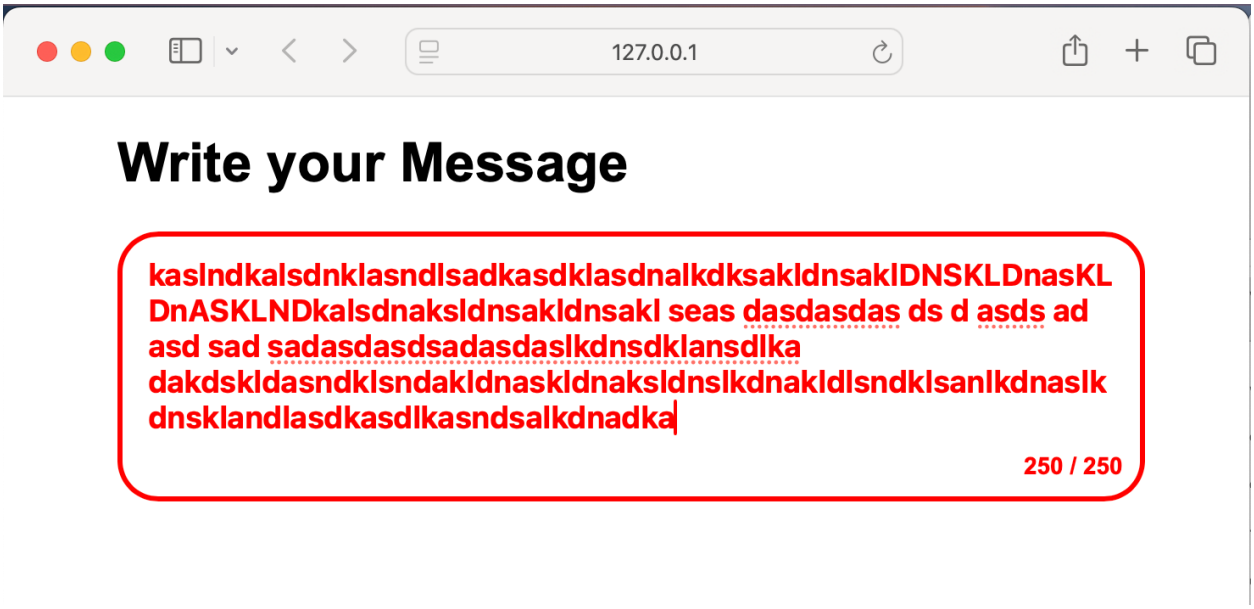
```
const textarea = document.getElementById("message");
const container = document.getElementById("container");
const charCount = document.getElementById("charCount");
const maxChars = 250;

function autoResize() {
  textarea.style.height = 'auto'; // Reset height
  textarea.style.height = ` ${textarea.scrollHeight}px` ;
}

textarea.addEventListener("input", () => {
  const currentChars = textarea.value.length;
  charCount.textContent = ` ${currentChars} / ${maxChars}` ;

  if (currentChars >= maxChars) {
    container.classList.add("limit");
    textarea.classList.add("limit");
    charCount.classList.add("limit");
    textarea.value = textarea.value.substring(0, maxChars); // Prevent
additional input
```

```
 } else {
   container.classList.remove("limit");
   textarea.classList.remove("limit");
   charCount.classList.remove("limit");

 }
 autoResize();
});
```

Output:

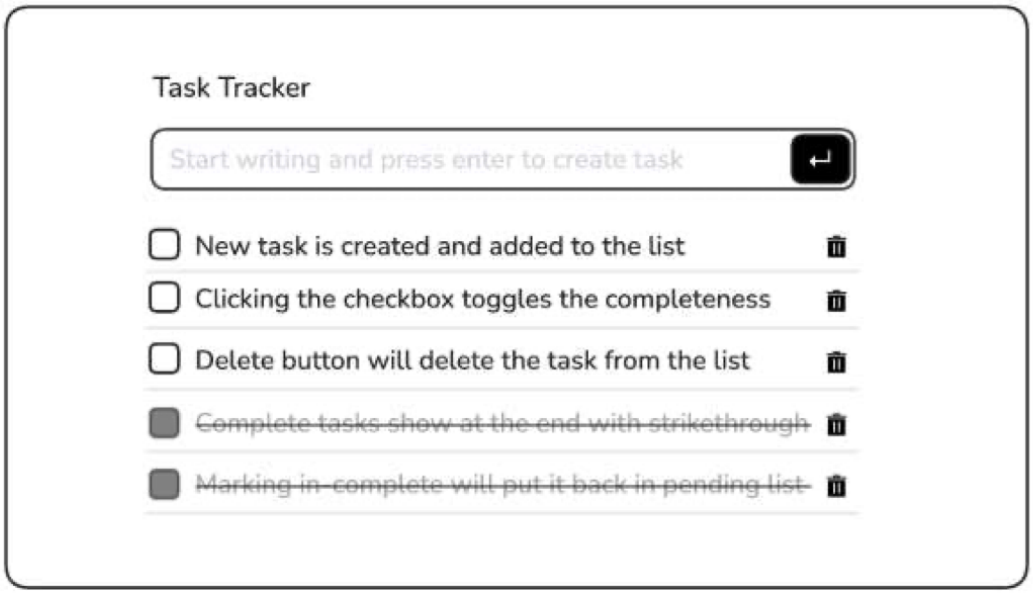






[Github Deployment: https://nambobby.github.io/Web-Lab-UWE-IU/Lab_2/Workshop_02/Question_4/Question_4.html](https://nambobby.github.io/Web-Lab-UWE-IU/Lab_2/Workshop_02/Question_4/Question_4.html)

Question 5: You are required to create a task tracker that lets users add new tasks, mark them as complete, or delete them. Completed tasks will be moved to the end of the list and will have strikethrough, and users can unmark tasks to return them to the pending list.



Question_5.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Task Tracker</title>
    <link rel="stylesheet" href="Question_5.css" />
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-beta3/css/all.min.css">
  </head>
  <body>
    <div class="container">
```

```
    <h2>Task Tracker</h2>
    <div class="input-container">
      <input
        type="text"
        id="taskInput"
        placeholder="Start writing and press enter to create task" />
      <span class="enter-icon-container"><i class="fas fa-arrow-turn-
down icon"></i></span>
    </div>
    <ul id="taskList"></ul>
  </div>
  <script src="Question_5.js"></script>
 </body>
</html>
```

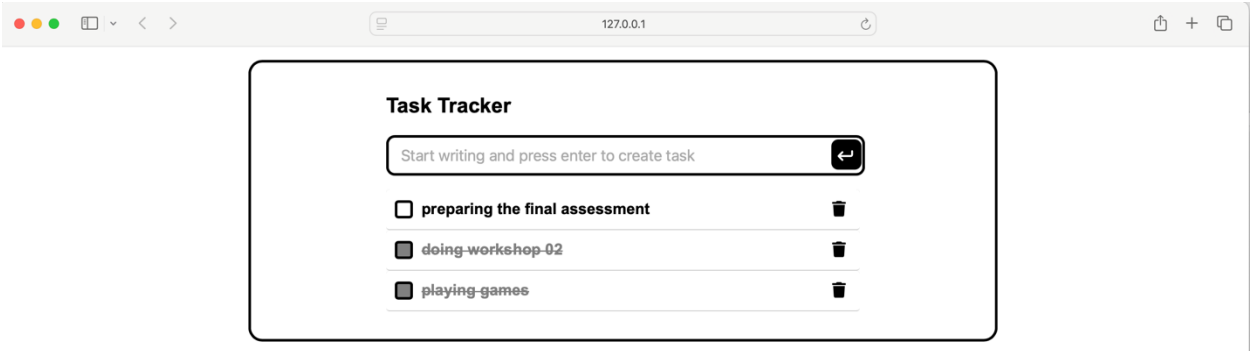
Question_5.js

```
const taskInput = document.getElementById('taskInput');
const taskList = document.getElementById('taskList');
const enterIcon = document.querySelector('.enter-icon-container');

// Function to add a new task
function addTask(taskText) {
  const taskItem = document.createElement('li');
  taskItem.classList.add('task-item');

  const checkbox = document.createElement('input');
  checkbox.type = 'checkbox';
  checkbox.style.display = 'none';
  checkbox.addEventListener('change', () =>
toggleTaskComplete(taskItem));

  const customCheckbox = document.createElement('span');
```

```
  customCheckbox.classList.add('custom-checkbox');
  customCheckbox.addEventListener('click', () => {
    checkbox.checked = !checkbox.checked;
    checkbox.dispatchEvent(new Event('change'));
  });

  const taskContent = document.createElement('span');
  taskContent.classList.add('content');
  taskContent.textContent = taskText;

  const deleteButton = document.createElement('button');
  deleteButton.classList.add('delete-btn');
  deleteButton.innerHTML = '<i class="fas fa-trash"></i>'; // Dustbin
icon
  deleteButton.addEventListener('click', () => deleteTask(taskItem));

  taskItem.appendChild(checkbox);
  taskItem.appendChild(customCheckbox);
  taskItem.appendChild(taskContent);
  taskItem.appendChild(deleteButton);
  taskList.appendChild(taskItem);
}

// Function to handle task completion
function toggleTaskComplete(taskItem) {
  taskItem.classList.toggle('completed');

  if (taskItem.classList.contains('completed')) {
    taskList.appendChild(taskItem);
  } else {
    taskList.insertBefore(taskItem, taskList.firstChild);
  }
}
```

```
// Function to delete a task
function deleteTask(taskItem) {
  taskList.removeChild(taskItem);
}

taskInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter' && taskInput.value.trim() !== '') {
    addTask(taskInput.value.trim());
    taskInput.value = '';
  }
});

enterIcon.addEventListener('click', () => {
  if (taskInput.value.trim() !== '') {
    addTask(taskInput.value.trim());
    taskInput.value = '';
  }
});
```

Output:



Github Deployment: ***https://nambobby.github.io/Web-Lab-UWE-IU/Lab_2/Workshop_02/Question_5/Question_5.html***