

# **CA-3 Report**

## **To Do List using HTML, CSS, Javascript**

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING**

**LOVELY PROFESSIONAL UNIVERSITY  
PHAGWARA,PUNJAB**



**SUBMITTED BY**

**Name of student: Rohan Saraswat**

**Registration Number: 12008941**

**Roll No.: 46**

**Section: KM088**

**Course Code: INT219:FRONT END WEB DEVELOPER**

# INDEX

1. HTML	3
2. CSS	4
3. JavaScript	5
4. HTML vs CSS vs JS	6
5. To Do List Project	7
6. Features	7
7. HTML File	8
8. CSS File	9
9. JavaScript File and working of the project	12
10. Working of Add Task	13
11. Working of Edit/Save Button	13
12. Working of Delete Button	14
13. Learning Outcomes	14
14. References	16

# HTML

15. The HyperText Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.
16. HTML can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript.
17. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.
18. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

# CSS

- **Cascading Style Sheets (CSS)** is a style sheet language used for describing the presentation of a document written in a markup language such as HTML.
- CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.
- CSS is designed to enable the separation of content and presentation, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content.
- The CSS specifications are maintained by the World Wide Web Consortium (W3C)

# JavaScript

- JavaScript often abbreviated as JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS.
- As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries.
- All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.
- JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node.js.
- Although Java and JavaScript are similar in name, syntax, and respective standard libraries, the two languages are distinct and differ greatly in design.

# **HTML vs CSS vs JS**

## **HTML: THE SKELETON:**

**HTML defines the content and structure of your website. Using a body for an analogy, HTML would be your skeleton, holding everything together and providing a base to hang everything off. A HTML file is made up of numerous HTML elements.**

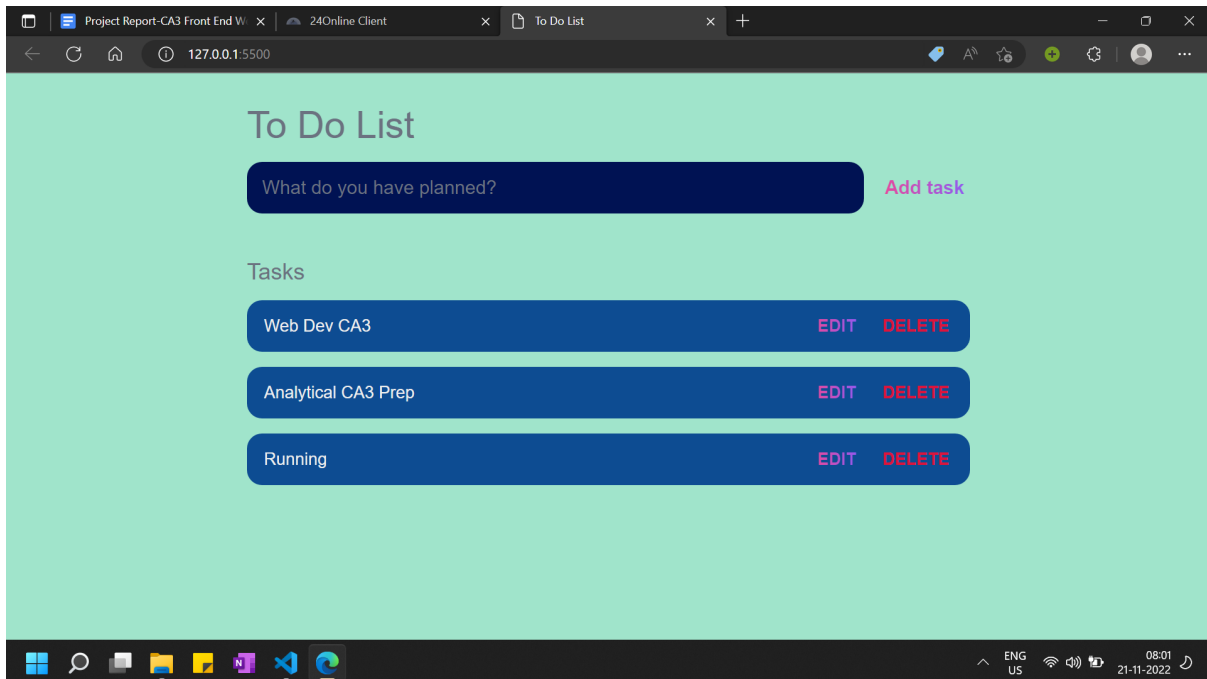
## **CSS: THE SKIN**

**If HTML is the skeleton to our body, then Cascading Style Sheets (CSS) is the skin. CSS is used to change the look and feel of websites.**

## **JAVASCRIPT: THE MUSCLES**

**Up to this point the web was a relatively static place, good for looking up information but not very interactive. In our body analogy, JavaScript takes the place of muscles, providing motion and behaviour to our limbs.**

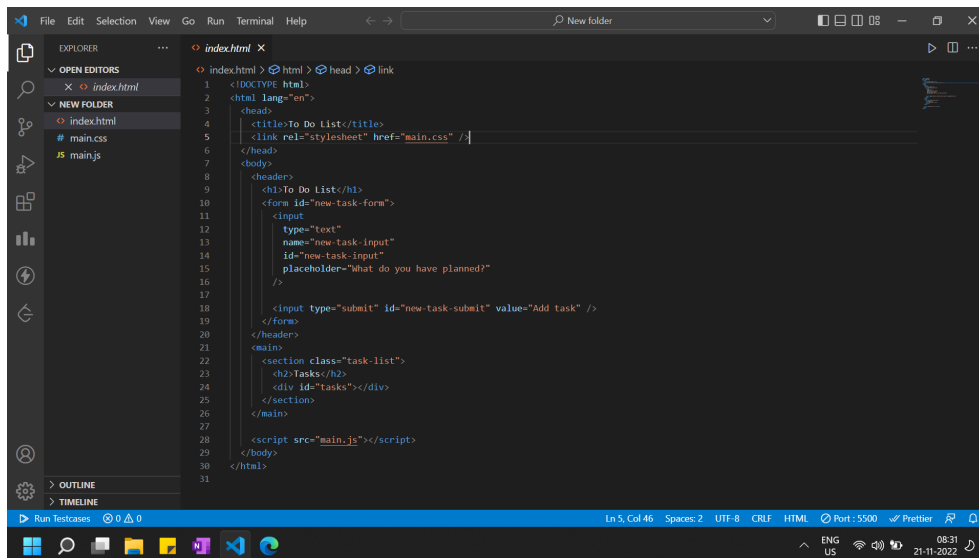
# To Do List Project



## Features

- **Add a Task:** To add a task write the task in the input field and click on “Add Task”.
- **Edit a Task:** Click on “Edit” to edit an added task, if some changes are to be made. On editing the edit button changes to “Save”, click on save button to save the changes.
- **Delete a Task:** If the task has been completed or is no longer required, click on the delete button which deletes the task from the list.

# HTML File

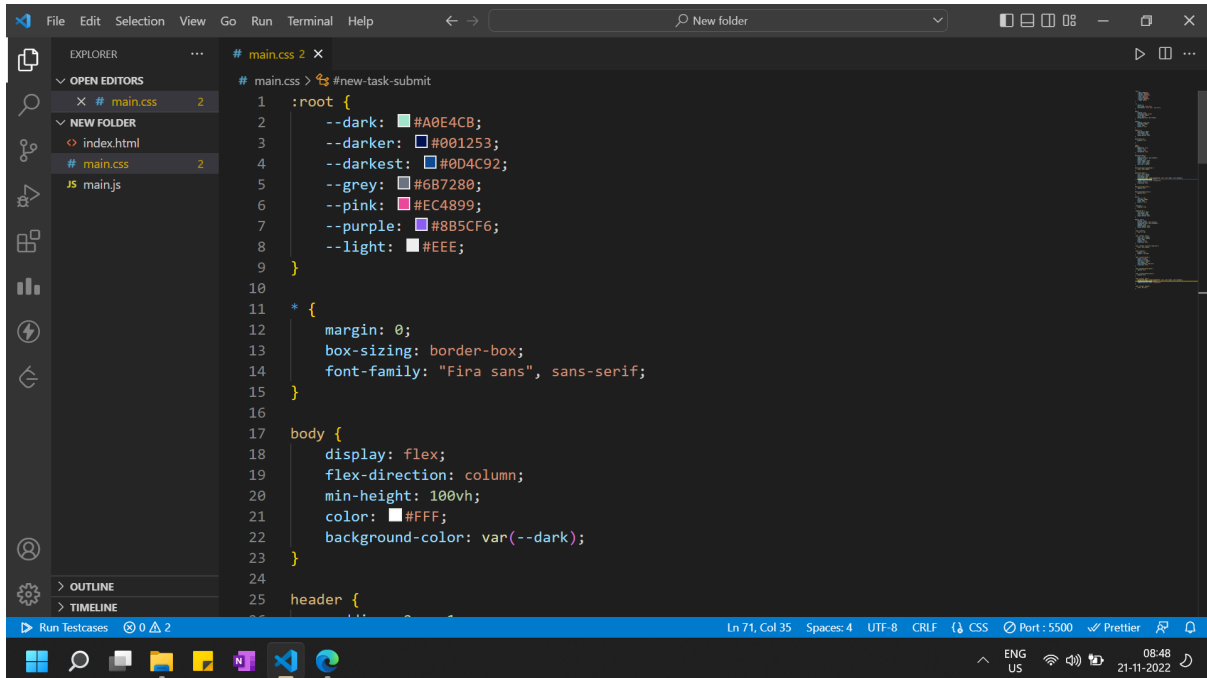


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <title>To Do List</title>
5   <link rel="stylesheet" href="main.css" />
6 </head>
7 <body>
8   <header>
9     <h1>To Do List</h1>
10    <form id="new-task-form">
11      <input
12        type="text"
13        name="new-task-input"
14        id="new-task-input"
15        placeholder="What do you have planned?"
16      />
17      <input type="submit" id="new-task-submit" value="Add task" />
18    </form>
19  </header>
20  <main>
21    <section class="task-list">
22      <h2>Tasks</h2>
23      <div id="tasks"></div>
24    </section>
25  </main>
26
27  <script src="main.js"></script>
28 </body>
29 </html>
```

- H1 tag contains the heading - “To Do List”
- Then comes the input text field, inside the form tag, with an id and a placeholder.
- The submit button is used to submit the value entered by the user and add it to the list of tasks.
- The section tag with the id “task-list” will contain the list of tasks entered by the user, and these tasks will be added with the help of JavaScript.



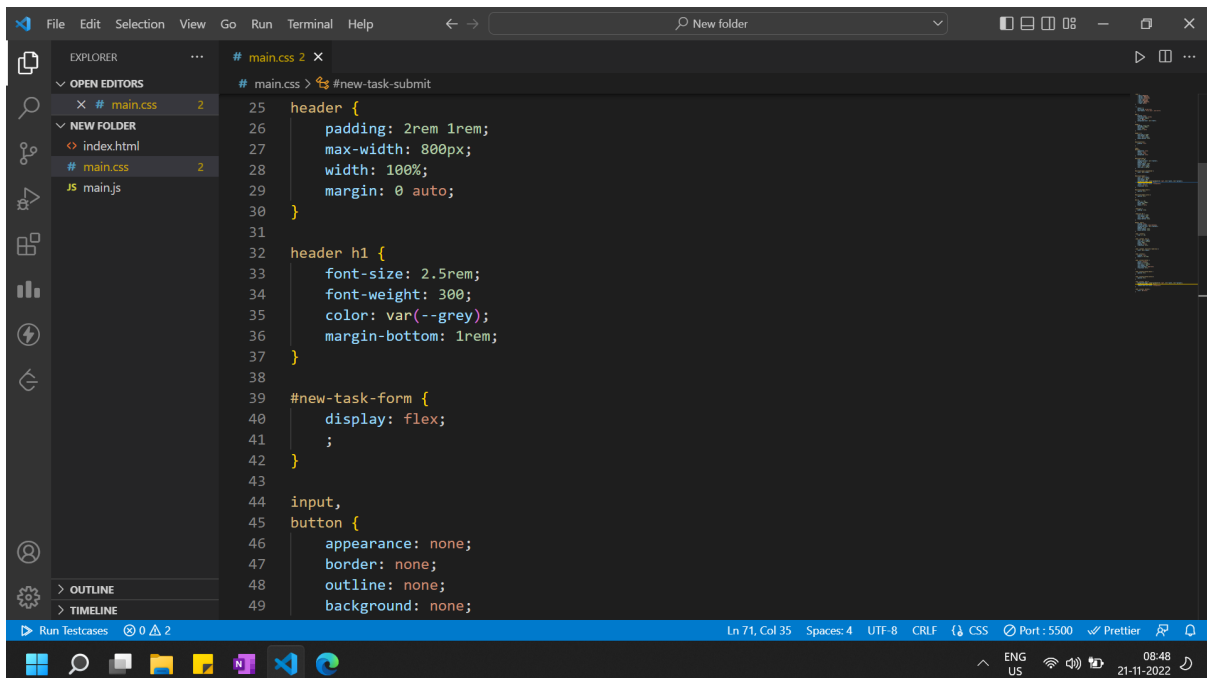
# CSS File



This screenshot shows the Visual Studio Code editor with the `main.css` file open. The Explorer sidebar on the left shows the project structure with `main.css` selected. The main editor area displays the following CSS code:

```
# main.css > #new-task-submit
1 :root {
2   --dark: #A0E4CB;
3   --darker: #001253;
4   --darkest: #0D4C92;
5   --grey: #6B7280;
6   --pink: #EC4899;
7   --purple: #8B5CF6;
8   --light: #EEE;
9 }
10
11 * {
12   margin: 0;
13   box-sizing: border-box;
14   font-family: "Fira sans", sans-serif;
15 }
16
17 body {
18   display: flex;
19   flex-direction: column;
20   min-height: 100vh;
21   color: #FFF;
22   background-color: var(--dark);
23 }
24
25 header {
```

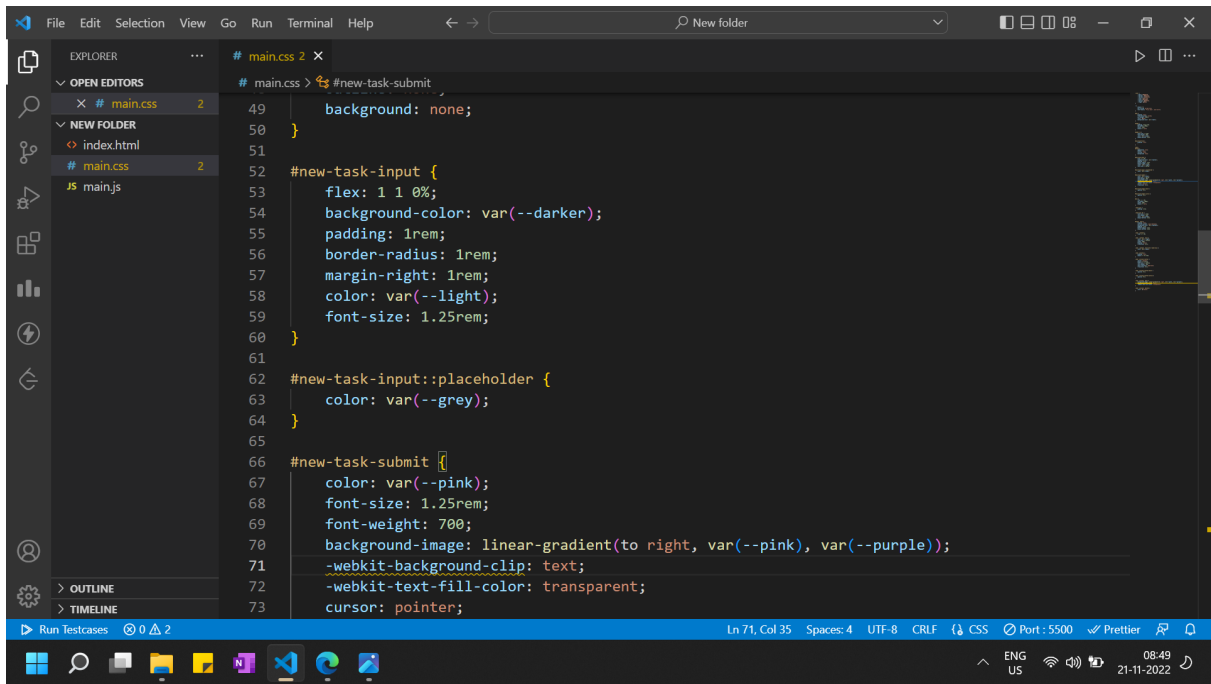
The status bar at the bottom indicates the cursor is at line 71, column 35, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The Port 5500 and Prettier extensions are also visible.



This screenshot shows the Visual Studio Code editor with the `main.css` file open, displaying additional CSS rules. The Explorer sidebar on the left shows the project structure with `main.css` selected. The main editor area displays the following CSS code:

```
# main.css > #new-task-submit
25 header {
26   padding: 2rem 1rem;
27   max-width: 800px;
28   width: 100%;
29   margin: 0 auto;
30 }
31
32 header h1 {
33   font-size: 2.5rem;
34   font-weight: 300;
35   color: var(--grey);
36   margin-bottom: 1rem;
37 }
38
39 #new-task-form {
40   display: flex;
41   ;
42 }
43
44 input,
45 button {
46   appearance: none;
47   border: none;
48   outline: none;
49   background: none;
```

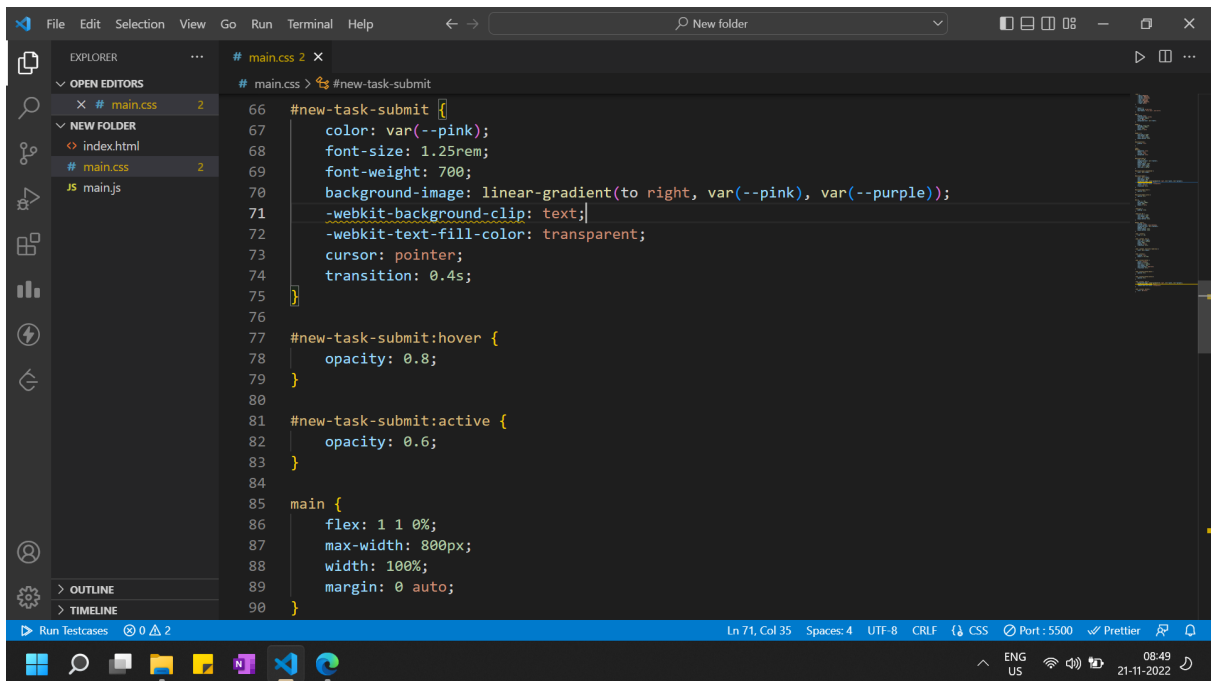
The status bar at the bottom indicates the cursor is at line 71, column 35, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The Port 5500 and Prettier extensions are also visible.



This screenshot shows the Visual Studio Code editor with a CSS file named `main.css` open. The Explorer sidebar on the left shows the project structure with `main.css` and `main.js` files. The main editor area displays the following CSS code:

```
#main.css 2 X
# main.css > #new-task-submit
49   background: none;
50 }
51
52 #new-task-input {
53   flex: 1 1 0%;
54   background-color: var(--darker);
55   padding: 1rem;
56   border-radius: 1rem;
57   margin-right: 1rem;
58   color: var(--light);
59   font-size: 1.25rem;
60 }
61
62 #new-task-input::placeholder {
63   color: var(--grey);
64 }
65
66 #new-task-submit {
67   color: var(--pink);
68   font-size: 1.25rem;
69   font-weight: 700;
70   background-image: linear-gradient(to right, var(--pink), var(--purple));
71   -webkit-background-clip: text;
72   -webkit-text-fill-color: transparent;
73   cursor: pointer;
```

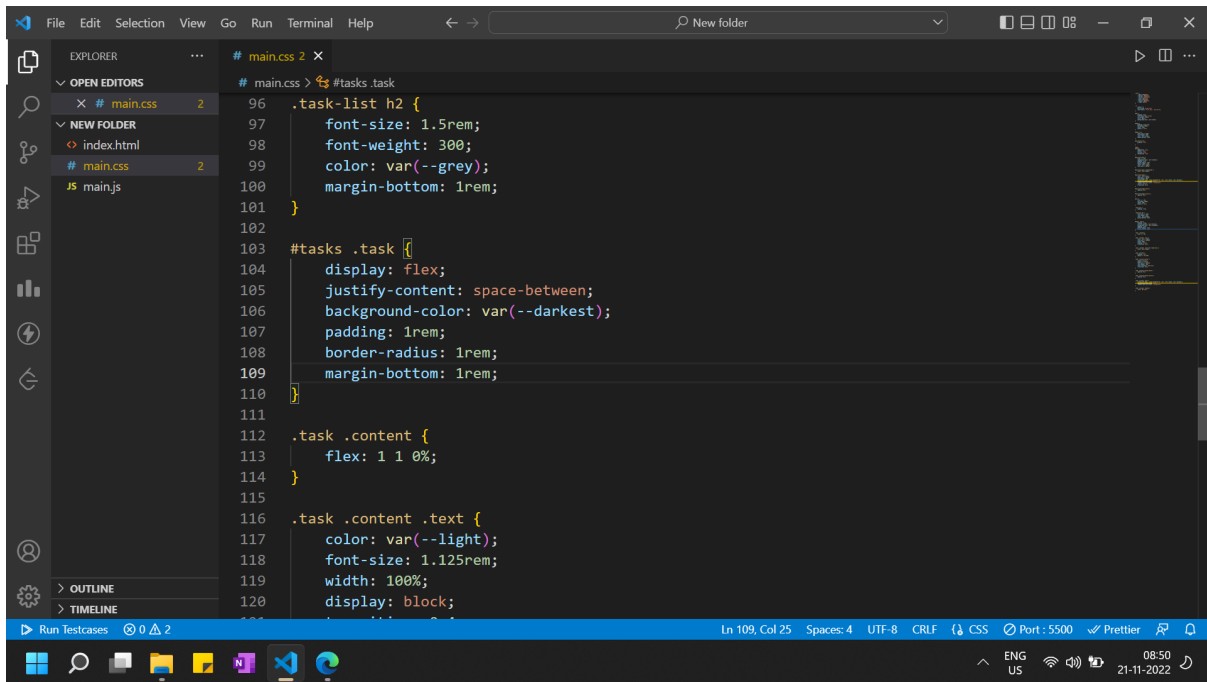
The status bar at the bottom indicates the current position is Line 71, Column 35, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The Port 5500 and Prettier extensions are also visible.



This screenshot shows the Visual Studio Code editor with the same CSS file, `main.css`, open. The main editor area displays the following CSS code:

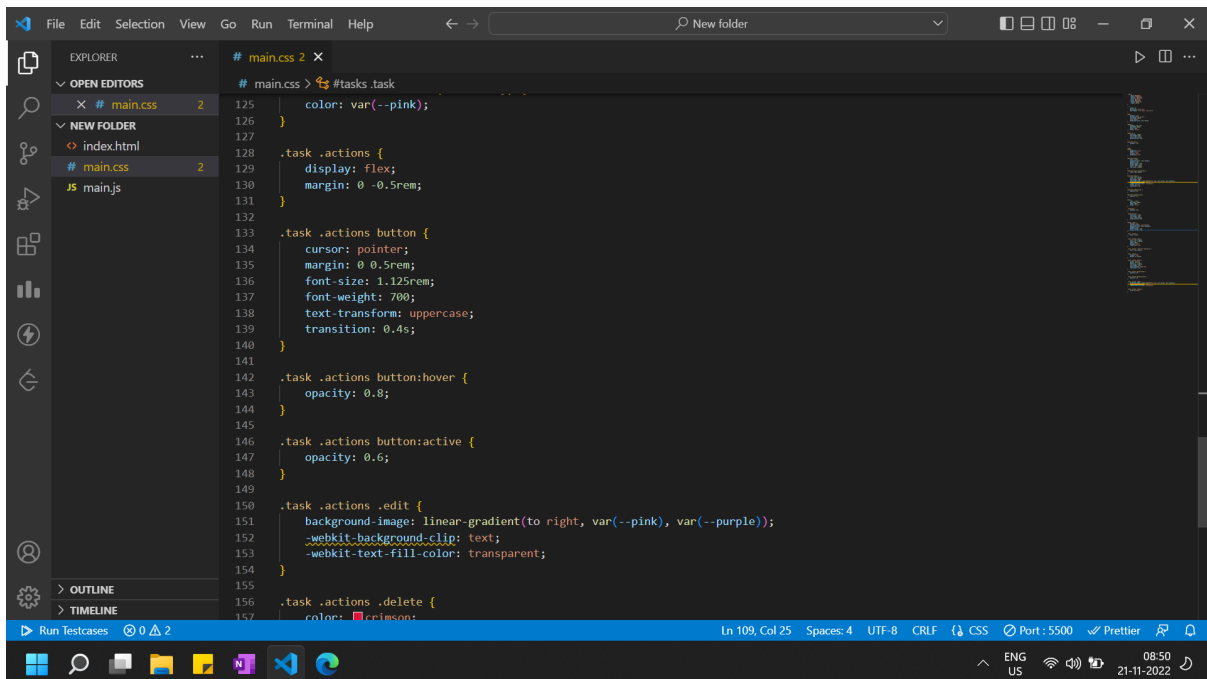
```
#main.css 2 X
# main.css > #new-task-submit
66 #new-task-submit {
67   color: var(--pink);
68   font-size: 1.25rem;
69   font-weight: 700;
70   background-image: linear-gradient(to right, var(--pink), var(--purple));
71   -webkit-background-clip: text;
72   -webkit-text-fill-color: transparent;
73   cursor: pointer;
74   transition: 0.4s;
75 }
76
77 #new-task-submit:active {
78   opacity: 0.8;
79 }
80
81 #new-task-submit:active {
82   opacity: 0.6;
83 }
84
85 main {
86   flex: 1 1 0%;
87   max-width: 800px;
88   width: 100%;
89   margin: 0 auto;
90 }
```

The status bar at the bottom indicates the current position is Line 71, Column 35, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The Port 5500 and Prettier extensions are also visible.



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'main.css' file is open, showing CSS code for a task list. The code includes a selector for the task list, a selector for the task container, and a selector for the task content. The status bar at the bottom indicates the current line is 109, column 25, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The port 5500 is also shown.

```
# main.css 2 X
# main.css > #tasks.task
96 .task-list h2 {
97     font-size: 1.5rem;
98     font-weight: 300;
99     color: var(--grey);
100     margin-bottom: 1rem;
101 }
102
103 #tasks .task {
104     display: flex;
105     justify-content: space-between;
106     background-color: var(--darkest);
107     padding: 1rem;
108     border-radius: 1rem;
109     margin-bottom: 1rem;
110 }
111
112 .task .content {
113     flex: 1 1 0%;
114 }
115
116 .task .content .text {
117     color: var(--light);
118     font-size: 1.125rem;
119     width: 100%;
120     display: block;
```

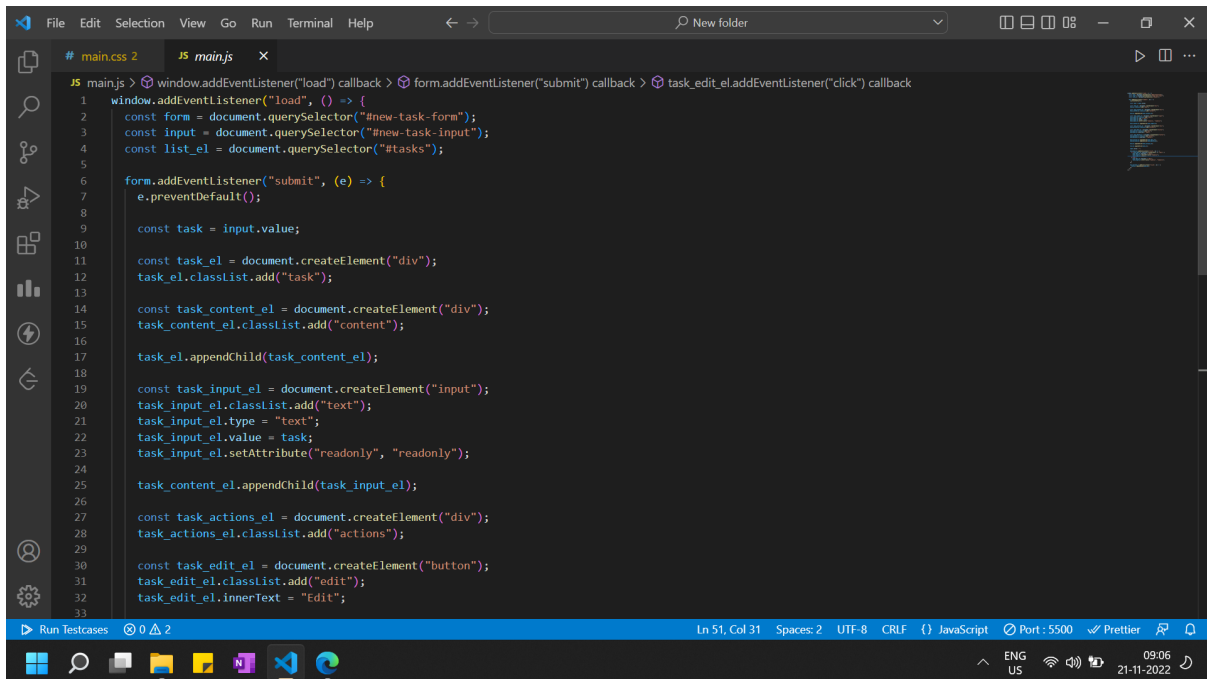


The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'main.css' file is open, showing CSS code for a task list. The code includes a selector for the task list, a selector for the task container, and a selector for the task content. The status bar at the bottom indicates the current line is 109, column 25, with 4 spaces, UTF-8 encoding, CRLF line endings, and CSS syntax highlighting. The port 5500 is also shown.

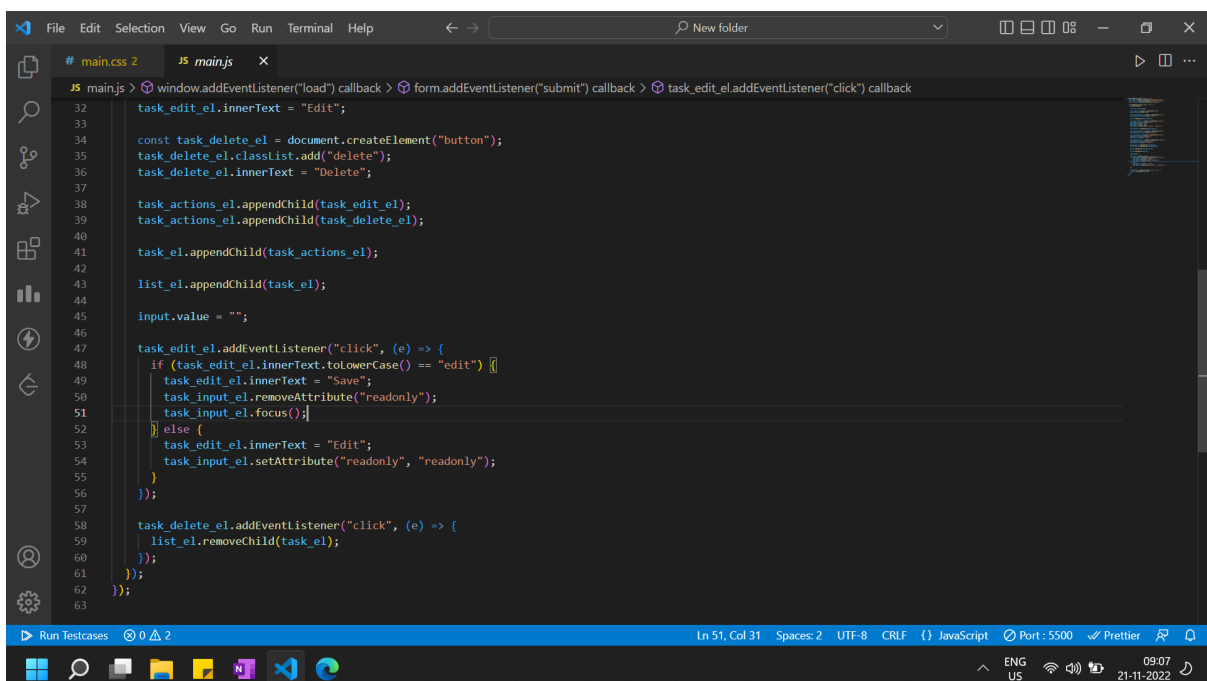
```
# main.css 2 X
# main.css > #tasks.task
125     color: var(--pink);
126 }
127
128 .task .actions {
129     display: flex;
130     margin: 0 -0.5rem;
131 }
132
133 .task .actions button {
134     cursor: pointer;
135     margin: 0 0.5rem;
136     font-size: 1.125rem;
137     font-weight: 700;
138     text-transform: uppercase;
139     transition: 0.4s;
140 }
141
142 .task .actions button:hover {
143     opacity: 0.8;
144 }
145
146 .task .actions button:active {
147     opacity: 0.6;
148 }
149
150 .task .actions .edit {
151     background-image: linear-gradient(to right, var(--pink), var(--purple));
152     -webkit-background-clip: text;
153     -webkit-text-fill-color: transparent;
154 }
155
156 .task .actions .delete {
157     color: red;
```

- The Css file consists of all the global variables and styles given to the classes and ids used in the project.

# JavaScript File and working of the project



```
JS main.js > window.addEventListener("load") callback > form.addEventListener("submit") callback > task_edit_el.addEventListener("click") callback
1 window.addEventListener("load", () => {
2   const form = document.querySelector("#new-task-form");
3   const input = document.querySelector("#new-task-input");
4   const list_el = document.querySelector("#tasks");
5
6   form.addEventListener("submit", (e) => {
7     e.preventDefault();
8
9     const task = input.value;
10
11    const task_el = document.createElement("div");
12    task_el.classList.add("task");
13
14    const task_content_el = document.createElement("div");
15    task_content_el.classList.add("content");
16
17    task_el.appendChild(task_content_el);
18
19    const task_input_el = document.createElement("input");
20    task_input_el.classList.add("text");
21    task_input_el.type = "text";
22    task_input_el.value = task;
23    task_input_el.setAttribute("readonly", "readonly");
24
25    task_content_el.appendChild(task_input_el);
26
27    const task_actions_el = document.createElement("div");
28    task_actions_el.classList.add("actions");
29
30    const task_edit_el = document.createElement("button");
31    task_edit_el.classList.add("edit");
32    task_edit_el.innerText = "Edit";
33  });
```



```
JS main.js > window.addEventListener("load") callback > form.addEventListener("submit") callback > task_edit_el.addEventListener("click") callback
32 task_edit_el.innerText = "Edit";
33
34 const task_delete_el = document.createElement("button");
35 task_delete_el.classList.add("delete");
36 task_delete_el.innerText = "Delete";
37
38 task_actions_el.appendChild(task_edit_el);
39 task_actions_el.appendChild(task_delete_el);
40
41 task_el.appendChild(task_actions_el);
42
43 list_el.appendChild(task_el);
44
45 input.value = "";
46
47 task_edit_el.addEventListener("click", (e) => {
48   if (task_edit_el.innerText.toLowerCase() === "edit") {
49     task_edit_el.innerText = "Save";
50     task_input_el.removeAttribute("readonly");
51     task_input_el.focus();
52   } else {
53     task_edit_el.innerText = "Edit";
54     task_input_el.setAttribute("readonly", "readonly");
55   }
56 });
57
58 task_delete_el.addEventListener("click", (e) => {
59   list_el.removeChild(task_el);
60 });
61
62 });
63
```

# **Working of the Add Task**

- The Js adds life to the project.
- When the user clicks on the add task button, a new div is created using the createElement() method and the relevant classes such as task, content are added to the div.
- 2 more divs with the classes content and actions are added to this div using classList.add().
- All of these divs are then appended to the div with id tasks as child elements using appendChild() method.

# **Working of Edit/Save Button**

- On clicking the edit button the eventListener attached to the button, listens to the event.
  - If the current value of the button is edit then the text is changed to save and the “readonly” attribute is removed using removeAttribute() method. The input field is the focused using the focus() method.

- If the current value of the button is save, then the value is changed to edit and the readonly attribute is added so that the input can no longer be changed.

## **Working of The Delete Button**

- On clicking the delete button the eventListener listens to the event and removes the element using the removeChild() method and thus the event gets removed.

## **Learning Outcomes**

### **HTML**

- Create simple pages using HTML

### **CSS**

- Create stylesheets and adding them to the HTML file

- Styling the pages using CSS
- How to use classes and ids and their making the webpage beautiful.
- CSS box model
- Usage and Types of colors in CSS
- Usage of flexbox

## **JavaScript**

- DOM Manipulation
- Usage of querySelector, getElementById
- Usage of EventListener
- Load, submit, click eventListener in JS
- JS methods to create, update, delete DOM nodes
- Adding, Editing, Removing elements to the DOM
- Adding classes and ids to a newly created DOM element
- Structuring a dynamic website

# **References**

- <https://en.wikipedia.org/wiki/HTML>
- <https://www.w3schools.com/html/>
- <https://en.wikipedia.org/wiki/CSS>
- <https://www.w3schools.com/css/>
- <https://www.w3schools.com/js/>
- <https://en.wikipedia.org/wiki/JavaScript>
- [HTML, CSS and JavaScript: The Anatomy of a Website | \\_nology Blog](#)
- [https://www.w3schools.com/howto/howto\\_js\\_todolist.asp](https://www.w3schools.com/howto/howto_js_todolist.asp)