

Laporan Praktikum
JavaScript – Aplikasi To-Do List



Nama :

Alyshia Cagivani Yasmin

NIM :4523210011

Mata Kuliah : Desain Web A

Tgl Prak : Selasa,1 Oktober 2024

Dosen Pengampu:

Adi Wahyu Pribadi

S1-Teknik Informatika

Fakultas Teknik Universitas Pancasila

2024/2025

Pendahuluan

- **Tujuan**

1. Agar dapat memahami sintaks dasar yang ada pada JavaScript
2. Mengimplementasikan JavaScript Class dan Object dalam aplikasi yang dinamis, menggunakan JavaScript DOM untuk mengambil, memanipulasi, dan memperbarui elemen HTML.
3. memanfaatkan fitur JavaScript untuk menyesuaikan logika aplikasi.

- **Deskripsi singkat tentang apk yang dibuat**

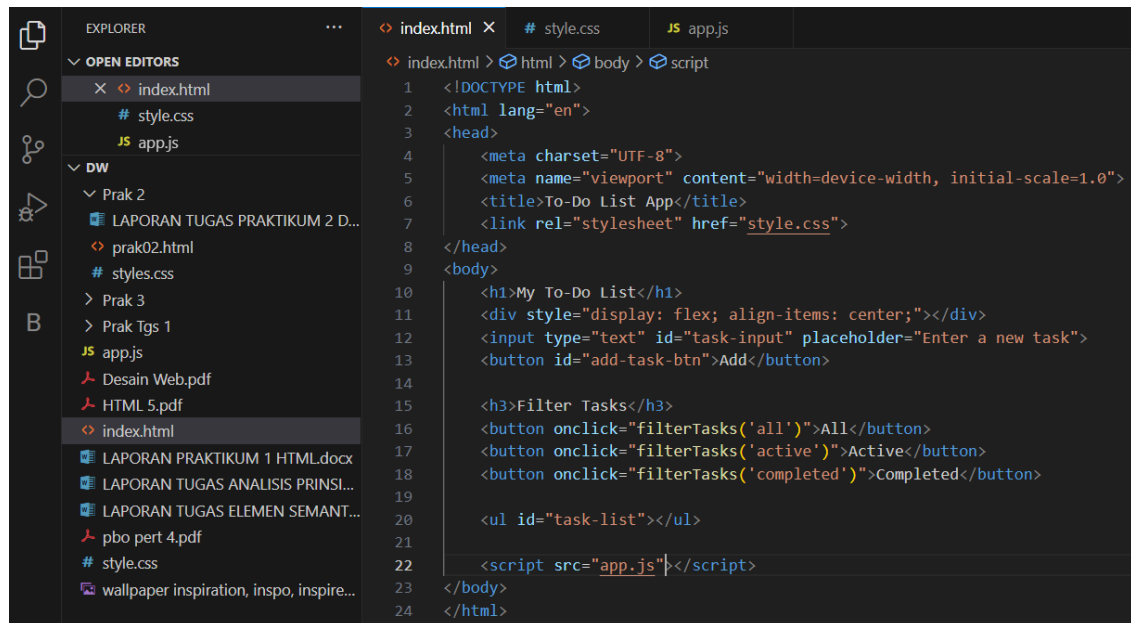
jadi, aplikasi tentang To-Do List ini merupakan aplikasi yang membantu untuk membuat daftar tugas – tugas kuliah atau hal lainnya. Aplikasi ini juga bisa untuk menandai tugas – tugas jika tugas yang dikerjakan sudah selesai dikerjakan, memvalidasi input, menghapus tugas jika tugas yang dimasukkan salah, dan dapat memfilter tugas.

- **Penjelasan Teori Dasar**

1. **DOM (Document Object Model)** -> untuk memanipulasi elemen – elemen yang ada di dalam dokumen HTML. Dengan DOM javascript dapat mengubah konteks, gaya, dll.
2. **Event Listeners** -> digunakan untuk merespon tindakan, contoh event : click on
3. **OOP (Object-Oriented Programming)** -> paradigma pemrograman yang menggunakan konsep "objek" untuk mengatur kode dan data.

Langkah Pengerjaan

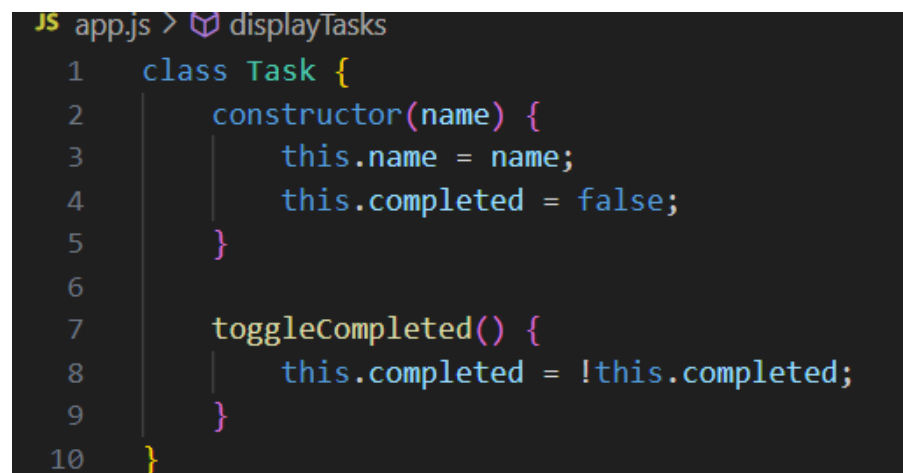
- Membuat file HTML, CSS, dan Java Script
- Pada file HTML membuat form untuk menginput task



The screenshot shows a code editor with three tabs: index.html, style.css, and app.js. The index.html tab is active, displaying the following code:

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>To-Do List App</title>
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10  <h1>My To-Do List</h1>
11  <div style="display: flex; align-items: center;"></div>
12  <input type="text" id="task-input" placeholder="Enter a new task">
13  <button id="add-task-btn">Add</button>
14
15  <h3>Filter Tasks</h3>
16  <button onclick="filterTasks('all')">All</button>
17  <button onclick="filterTasks('active')">Active</button>
18  <button onclick="filterTasks('completed')">Completed</button>
19
20  <ul id="task-list"></ul>
21
22  <script src="app.js"></script>
23 </body>
24 </html>
```

- Memprestasikan tugas yang memiliki property name dan completed di file javascript



The screenshot shows a code editor with a single tab: app.js. The code defines a class Task with the following methods:

```
1 class Task {
2   constructor(name) {
3     this.name = name;
4     this.completed = false;
5   }
6
7   toggleCompleted() {
8     this.completed = !this.completed;
9   }
10 }
```

- **Membuat fungsi untuk menambah tugas**

```
12 //Fungsi Menambahkan Tugas
13 let tasks = [];
14
15 function addTask() {
16     const taskInput = document.getElementById('task-input');
17     const taskName = taskInput.value.trim();
18
19     if (taskName === "") {
20         alert("Task cannot be empty!");
21         return;
22     }
23
24     const task = new Task(taskName);
25     tasks.push(task);
26     displayTasks();
27     taskInput.value = ''; // Kosongkan input setelah menambahkan
28 }
29
30 document.getElementById('add-task-btn').addEventListener('click', addTask);
```

- **Menampilkan tugas di DOM**

```
//Menampilkan Tugas DOM
function displayTasks() {
    const taskList = document.getElementById('task-list');
    taskList.innerHTML = '';

    tasks.forEach((task, index) => {
        const taskItem = document.createElement('li');
        taskItem.className = task.completed ? 'completed' : '';

        const taskText = document.createElement('span');
        taskText.textContent = task.name;

        const taskCheckbox = document.createElement('input');
        taskCheckbox.type = 'checkbox';
        taskCheckbox.checked = task.completed;
        taskCheckbox.addEventListener('click', () => toggleTask(index));

        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Delete';
        deleteButton.addEventListener('click', () => deleteTask(index));

        taskItem.appendChild(taskCheckbox);
        taskItem.appendChild(taskText);
        taskItem.appendChild(deleteButton);

        taskList.appendChild(taskItem);
    });
}
```

- Menandai tugas selesai dan menghapus tugas

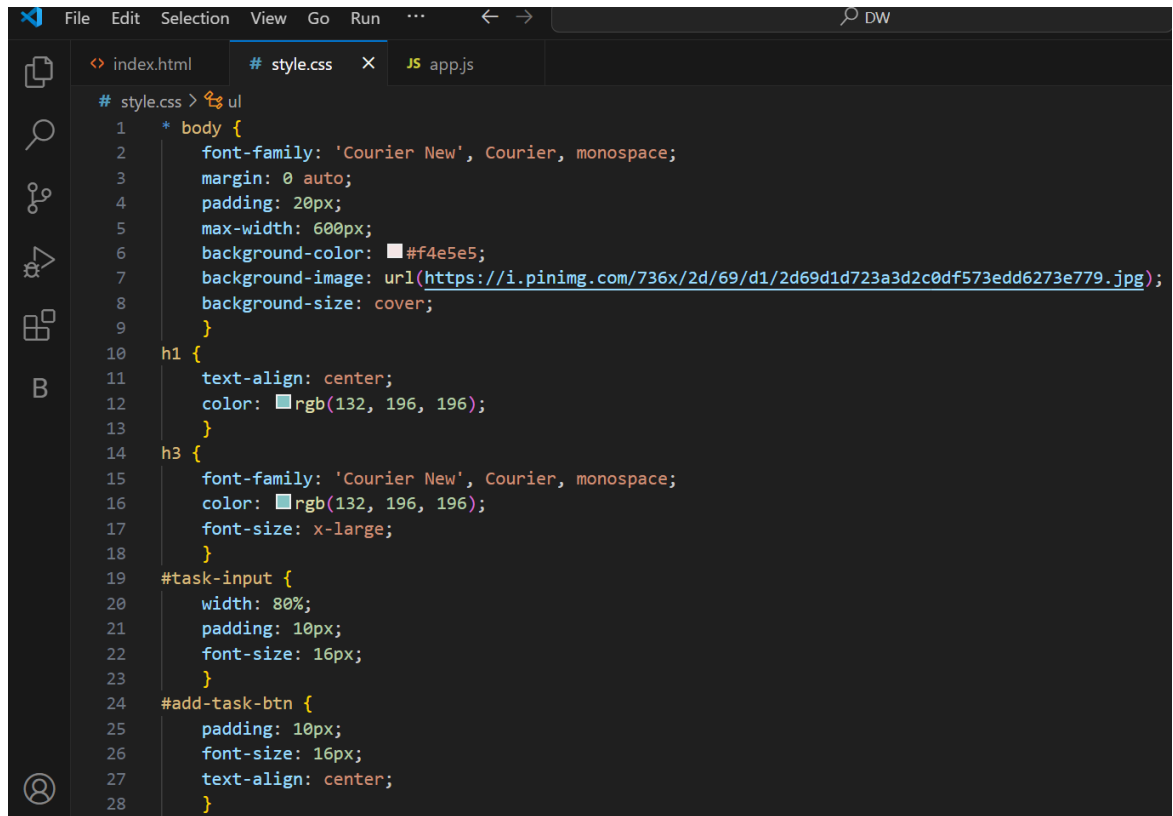
```
62 //Menandai Tugas Selesai dan Menghapus Tugas
63 function toggleTask(index) {
64     tasks[index].toggleCompleted();
65     displayTasks();
66 }
67
68 function deleteTask(index) {
69     tasks.splice(index, 1);
70     displayTasks();
71 }
72
```

- Memfilter tugas berdasarkan status

```
73 //Filter Tugas Berdasarkan Status
74 function filterTasks(filter) {
75     let filteredTasks = tasks;
76
77     if (filter === 'active') {
78         filteredTasks = tasks.filter(task => !task.completed);
79     } else if (filter === 'completed') {
80         filteredTasks = tasks.filter(task => task.completed);
81     }
82
83     const taskList = document.getElementById('task-list');
84     taskList.innerHTML = '';
85
86     filteredTasks.forEach((task, index) => {
87         const taskItem = document.createElement('li');
88         taskItem.className = task.completed ? 'completed' : '';
89
90         const taskText = document.createElement('span');
91         taskText.textContent = task.name;
92
93         const taskCheckbox = document.createElement('input');
94         taskCheckbox.type = 'checkbox';
95         taskCheckbox.checked = task.completed;
96         taskCheckbox.addEventListener('click', () => toggleTask(index));
97
98         const deleteButton = document.createElement('button');
99         deleteButton.textContent = 'Delete';
100         deleteButton.addEventListener('click', () => deleteTask(index));
101
102         taskItem.appendChild(taskCheckbox);
103         taskItem.appendChild(taskText);
104         taskItem.appendChild(deleteButton);
105
106         taskList.appendChild(taskItem);
107     });
108 }
```

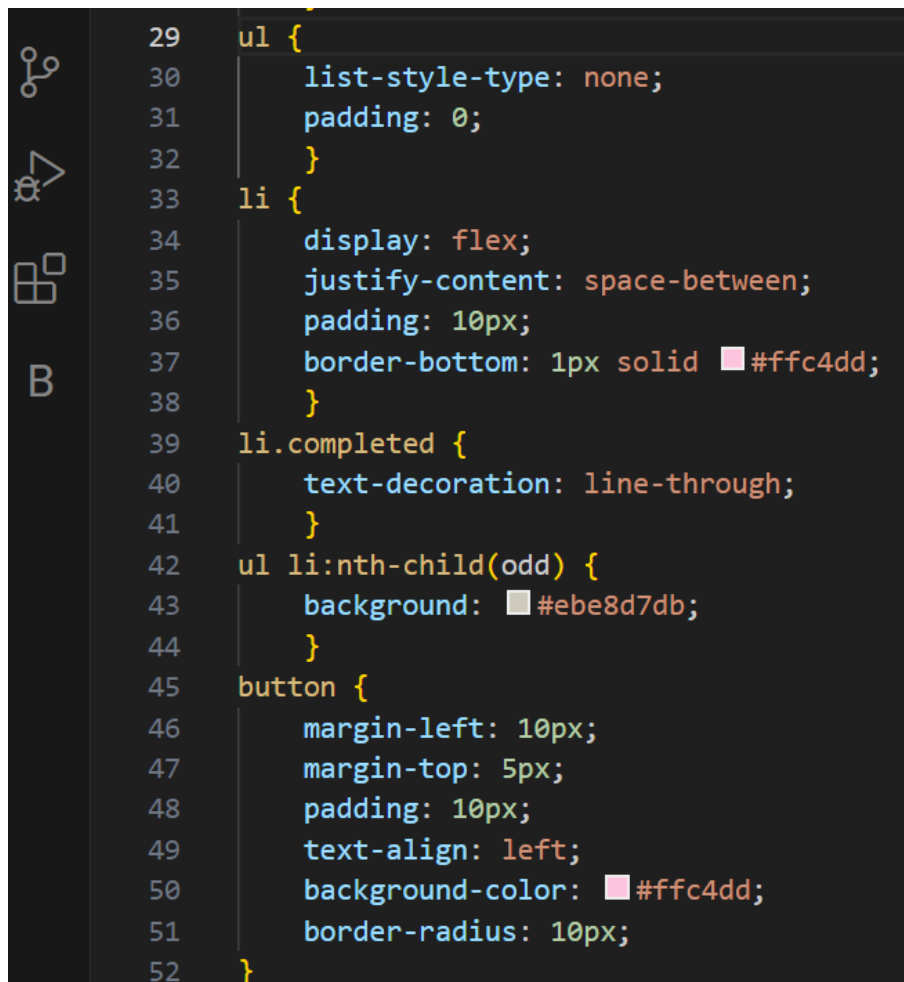
Ln 75, Col 26 Spaces: 4 UTF-8 CRLF {} JavaScript Go Live AI C

- Membuat file CSS untuk mengatur background, mengubah font dll



A screenshot of a code editor with three tabs: index.html, style.css, and app.js. The style.css tab is active, showing CSS code for a task list application. The code includes styles for the body, h1, h3, #task-input, and #add-task-btn. The body has a background color of #f4e5e5, a background image from a URL, and a width of 600px. The h1 and h3 have a color of rgb(132, 196, 196) and a font-family of 'Courier New'. The #task-input has a width of 80% and a font-size of 16px. The #add-task-btn has a padding of 10px and a font-size of 16px.

```
# style.css > ul
1  * body {
2      font-family: 'Courier New', Courier, monospace;
3      margin: 0 auto;
4      padding: 20px;
5      max-width: 600px;
6      background-color: #f4e5e5;
7      background-image: url(https://i.pinimg.com/736x/2d/69/d1/2d69d1d723a3d2c0df573edd6273e779.jpg);
8      background-size: cover;
9  }
10 h1 {
11     text-align: center;
12     color: rgb(132, 196, 196);
13 }
14 h3 {
15     font-family: 'Courier New', Courier, monospace;
16     color: rgb(132, 196, 196);
17     font-size: x-large;
18 }
19 #task-input {
20     width: 80%;
21     padding: 10px;
22     font-size: 16px;
23 }
24 #add-task-btn {
25     padding: 10px;
26     font-size: 16px;
27     text-align: center;
28 }
```



A screenshot of a code editor showing CSS code for a task list application. The code includes styles for ul, li, li.completed, ul li:nth-child(odd), and button. The ul has a list-style-type of none and a padding of 0. The li has a display of flex, justify-content of space-between, padding of 10px, and a border-bottom of 1px solid #ffc4dd. The li.completed has a text-decoration of line-through. The ul li:nth-child(odd) has a background color of #ebe8d7db. The button has a margin-left of 10px, margin-top of 5px, padding of 10px, text-align of left, background-color of #ffc4dd, and a border-radius of 10px.

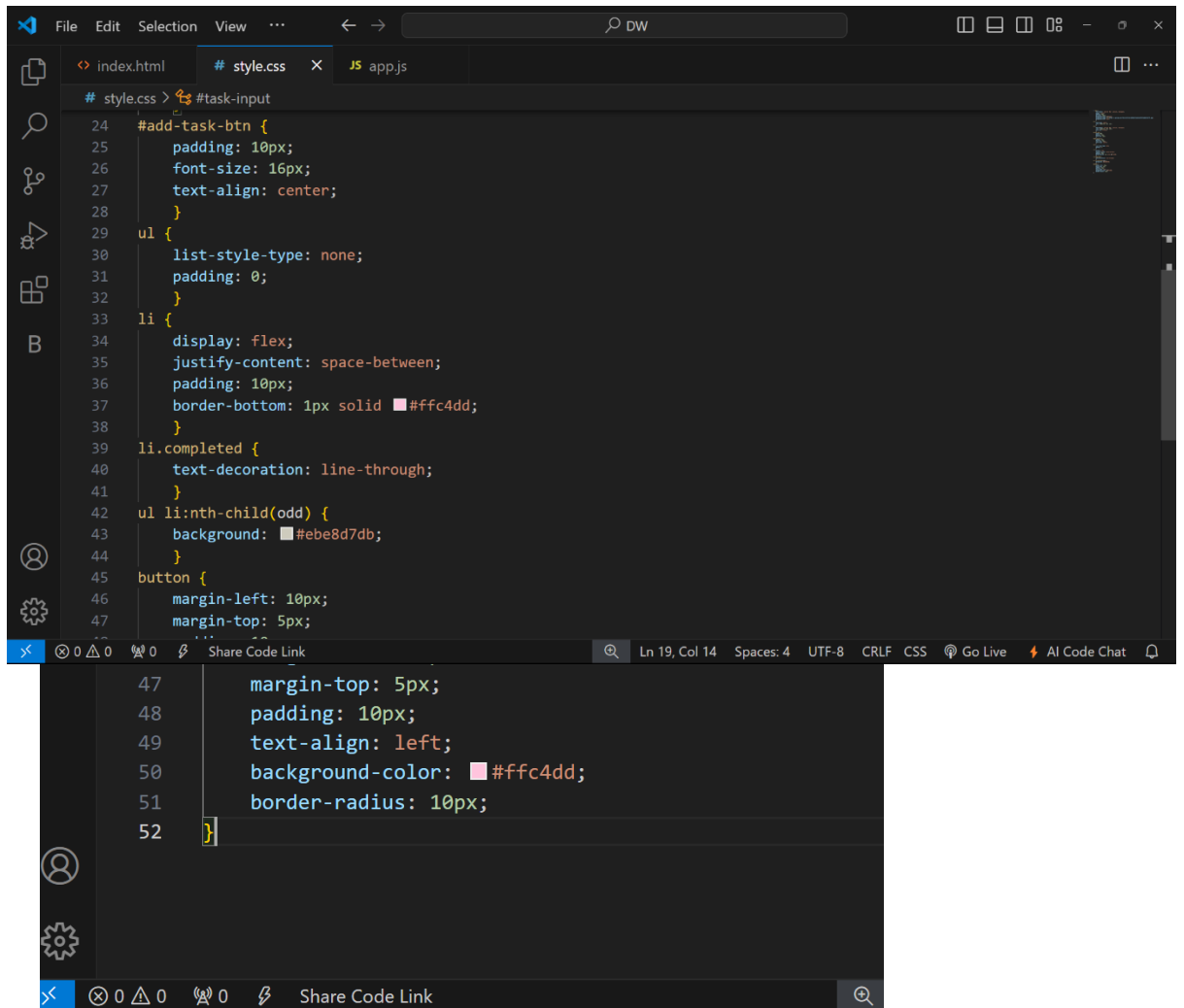
```
29 ul {
30     list-style-type: none;
31     padding: 0;
32 }
33 li {
34     display: flex;
35     justify-content: space-between;
36     padding: 10px;
37     border-bottom: 1px solid #ffc4dd;
38 }
39 li.completed {
40     text-decoration: line-through;
41 }
42 ul li:nth-child(odd) {
43     background: #ebe8d7db;
44 }
45 button {
46     margin-left: 10px;
47     margin-top: 5px;
48     padding: 10px;
49     text-align: left;
50     background-color: #ffc4dd;
51     border-radius: 10px;
52 }
```

Struktur dan Penjelasan Kode

- Sertakan kode JavaScript lengkap yang digunakan dalam aplikasi To-Do List.

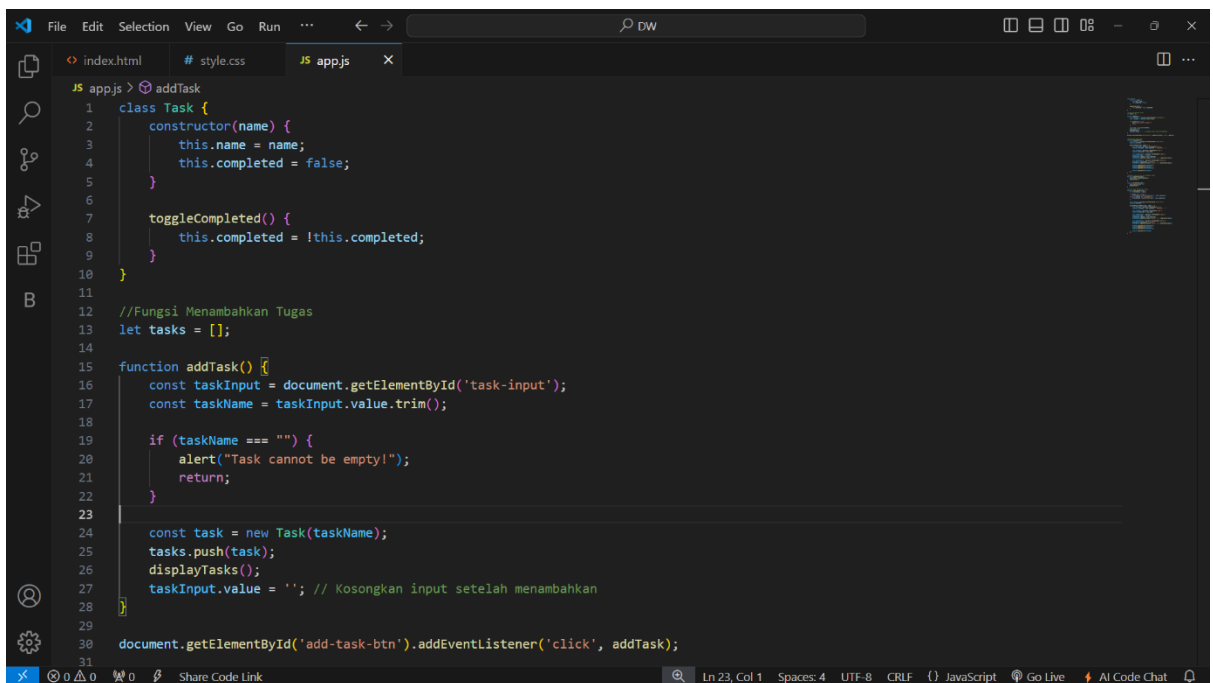
```
File Edit Selection View ... < > DW
index.html X # style.css JS app.js
index.html > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>To-Do List App</title>
7   <link rel="stylesheet" href="style.css">
8 </head>
9 <body>
10  <h1>My To-Do List</h1>
11  <div style="display: flex; align-items: center;"></div>
12  <input type="text" id="task-input" placeholder="Enter a new task">
13  <button id="add-task-btn">Add</button>
14
15  <h3>Filter Tasks</h3>
16  <button onclick="filterTasks('all')">All</button>
17  <button onclick="filterTasks('active')">Active</button>
18  <button onclick="filterTasks('completed')">Completed</button>
19
20  <ul id="task-list"></ul>
21
22  <script src="app.js"></script>
23 </body>
24 </html>
Ln 24, Col 8 Spaces: 4 UTF-8 CRLF HTML Go Live AI Code Chat
```

```
File Edit Selection View ... < > DW
index.html X # style.css X JS app.js
# style.css > ul
1 * body {
2   font-family: 'Courier New', Courier, monospace;
3   margin: 0 auto;
4   padding: 20px;
5   max-width: 600px;
6   background-color: #f4e5e5;
7   background-image: url(https://i.pinimg.com/736x/2d/69/d1/2d69d1d723a3d2c0df573edd6273e779.jpg);
8   background-size: cover;
9 }
10 h1 {
11   text-align: center;
12   color: rgb(132, 196, 196);
13 }
14 h3 {
15   font-family: 'Courier New', Courier, monospace;
16   color: rgb(132, 196, 196);
17   font-size: x-large;
18 }
19 #task-input {
20   width: 80%;
21   padding: 10px;
22   font-size: 16px;
23 }
```



The screenshot shows the VS Code editor with the `style.css` file open. The code defines styles for a task input field and a list of tasks. The `#add-task-btn` selector is highlighted, and its properties are listed: `padding: 10px;`, `font-size: 16px;`, `text-align: center;`, and `border-bottom: 1px solid #ffc4dd;`. The `ul` selector is also highlighted, with properties: `list-style-type: none;`, `padding: 0;`, and `border-bottom: 1px solid #ffc4dd;`. The `li` selector is highlighted, with properties: `display: flex;`, `justify-content: space-between;`, `padding: 10px;`, and `border-bottom: 1px solid #ffc4dd;`. The `li.completed` selector is highlighted, with properties: `text-decoration: line-through;` and `background-color: #e8e8e8;`. The `ul li:nth-child(odd)` selector is highlighted, with properties: `background-color: #e8e8e8;` and `border-bottom: 1px solid #ffc4dd;`. The `button` selector is highlighted, with properties: `margin-left: 10px;` and `margin-top: 5px;`. The status bar at the bottom shows the cursor is at line 19, column 14, with 4 spaces, UTF-8 encoding, and CRLF line endings.

```
# style.css > #task-input
24 #add-task-btn {
25   padding: 10px;
26   font-size: 16px;
27   text-align: center;
28 }
29 ul {
30   list-style-type: none;
31   padding: 0;
32 }
33 li {
34   display: flex;
35   justify-content: space-between;
36   padding: 10px;
37   border-bottom: 1px solid #ffc4dd;
38 }
39 li.completed {
40   text-decoration: line-through;
41 }
42 ul li:nth-child(odd) {
43   background-color: #e8e8e8;
44 }
45 button {
46   margin-left: 10px;
47   margin-top: 5px;
48   padding: 10px;
49   text-align: left;
50   background-color: #ffc4dd;
51   border-radius: 10px;
52 }
```



The screenshot shows the VS Code editor with the `app.js` file open. The code defines a `Task` class and a `addTask` function. The `Task` class has a constructor that takes a `name` parameter and sets `this.name` and `this.completed` to `false`. It also has a `toggleCompleted` method that toggles the `completed` property. The `addTask` function is defined to add a new task to the `tasks` array. It gets the value from the `task-input` field, trims it, and checks if it's empty. If it's empty, it shows an alert. If not, it creates a new `Task` object, pushes it to the `tasks` array, and calls `displayTasks`. The status bar at the bottom shows the cursor is at line 23, column 1, with 4 spaces, UTF-8 encoding, and CRLF line endings.

```
JS app.js > addTask
1 class Task {
2   constructor(name) {
3     this.name = name;
4     this.completed = false;
5   }
6
7   toggleCompleted() {
8     this.completed = !this.completed;
9   }
10 }
11
12 //Fungsi Menambahkan Tugas
13 let tasks = [];
14
15 function addTask() {
16   const taskInput = document.getElementById('task-input');
17   const taskName = taskInput.value.trim();
18
19   if (taskName === "") {
20     alert("Task cannot be empty!");
21     return;
22   }
23
24   const task = new Task(taskName);
25   tasks.push(task);
26   displayTasks();
27   taskInput.value = ''; // Kosongkan input setelah menambahkan
28 }
29
30 document.getElementById('add-task-btn').addEventListener('click', addTask);
31
```



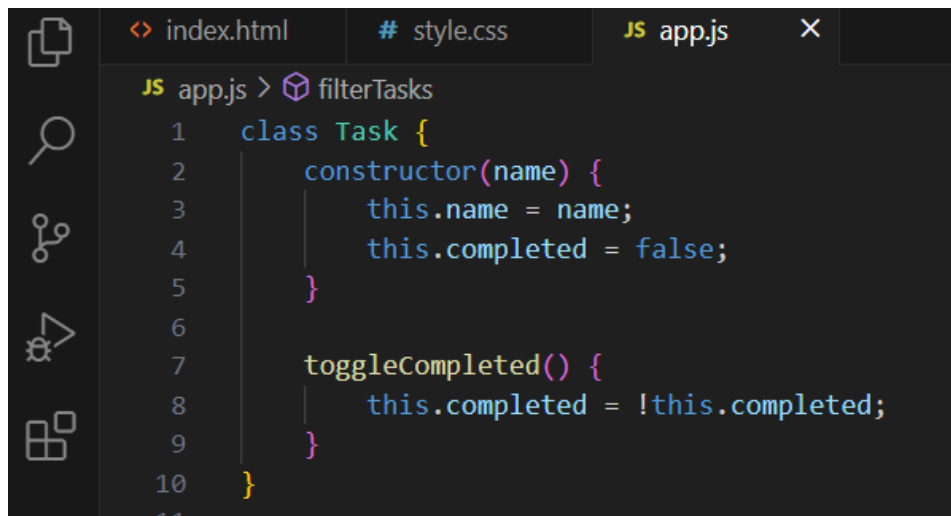
```
File Edit Selection View Go Run ... DW
index.html # style.css JS appjs X
JS appjs > addTask
31
32
33 //Menampilkan Tugas DOM
34 function displayTasks() {
35   const taskList = document.getElementById('task-list');
36   taskList.innerHTML = '';
37
38   tasks.forEach((task, index) => {
39     const taskItem = document.createElement('li');
40     taskItem.className = task.completed ? 'completed' : '';
41
42     const taskText = document.createElement('span');
43     taskText.textContent = task.name;
44
45     const taskCheckbox = document.createElement('input');
46     taskCheckbox.type = 'checkbox';
47     taskCheckbox.checked = task.completed;
48     taskCheckbox.addEventListener('click', () => toggleTask(index));
49
50     const deleteButton = document.createElement('button');
51     deleteButton.textContent = 'Delete';
52     deleteButton.addEventListener('click', () => deleteTask(index));
53
54     taskItem.appendChild(taskCheckbox);
55     taskItem.appendChild(taskText);
56     taskItem.appendChild(deleteButton);
57
58     taskList.appendChild(taskItem);
59   });
60 }
61
```

```
File Edit Selection View Go Run ... DW
index.html # style.css JS appjs X
JS appjs > displayTasks > tasks.forEach() callback
62 //Menandai Tugas Selesai dan Menghapus Tugas
63 function toggleTask(index) {
64   tasks[index].toggleCompleted();
65   displayTasks();
66 }
67
68 function deleteTask(index) {
69   tasks.splice(index, 1);
70   displayTasks();
71 }
72
73 //Filter Tugas Berdasarkan Status
74 function filterTasks(filter) {
75   let filteredTasks = tasks;
76
77   if (filter === 'active') {
78     filteredTasks = tasks.filter(task => !task.completed);
79   } else if (filter === 'completed') {
80     filteredTasks = tasks.filter(task => task.completed);
81   }
82
83   const taskList = document.getElementById('task-list');
84   taskList.innerHTML = '';
85
86   filteredTasks.forEach((task, index) => {
87     const taskItem = document.createElement('li');
88     taskItem.className = task.completed ? 'completed' : '';
89
90     const taskText = document.createElement('span');
91     taskText.textContent = task.name;
```

```
File Edit Selection View Go Run ... DW
index.html # style.css JS appjs X
JS appjs > filterTasks > filteredTasks.forEach() callback
74 function filterTasks(filter) {
86   filteredTasks.forEach((task, index) => {
92
93     const taskCheckbox = document.createElement('input');
94     taskCheckbox.type = 'checkbox';
95     taskCheckbox.checked = task.completed;
96     taskCheckbox.addEventListener('click', () => toggleTask(index));
97
98     const deleteButton = document.createElement('button');
99     deleteButton.textContent = 'Delete';
100    deleteButton.addEventListener('click', () => deleteTask(index));
101
102    taskItem.appendChild(taskCheckbox);
103    taskItem.appendChild(taskText);
104    taskItem.appendChild(deleteButton);
105
106    taskList.appendChild(taskItem);
107  });
108 }
```

- Jelaskan secara singkat setiap bagian dari kode:

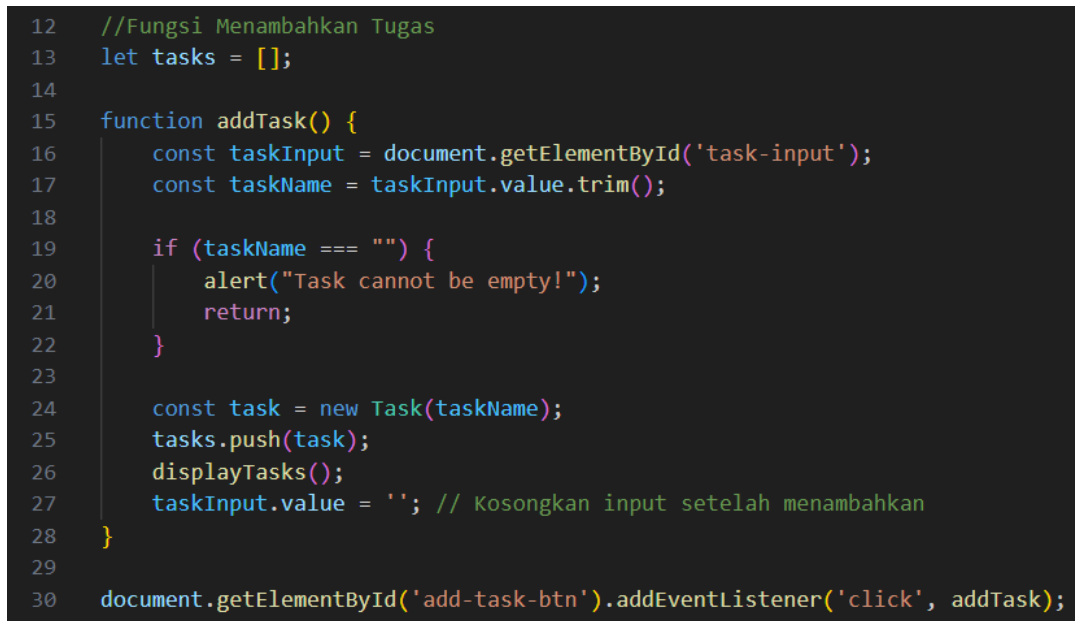
1. Class Task



```
JS app.js > filterTasks
1  class Task {
2      constructor(name) {
3          this.name = name;
4          this.completed = false;
5      }
6
7      toggleCompleted() {
8          this.completed = !this.completed;
9      }
10 }
11
```

Digunakan untuk mempresentasikan objek nama tugas dengan property name dan completed untuk status apakah tugas sudah selesai dikerjakan atau belum

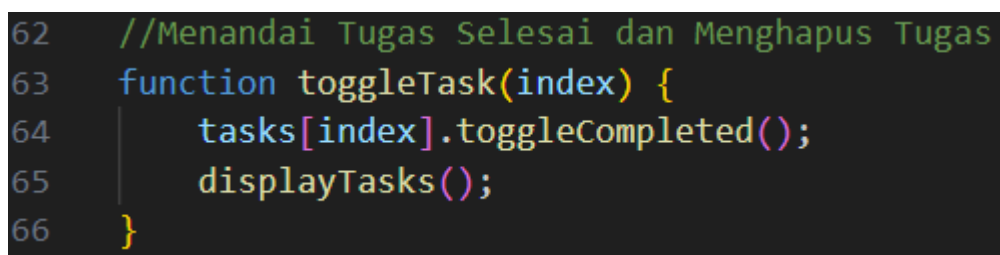
2. Add Task



```
12 //Fungsi Menambahkan Tugas
13 let tasks = [];
14
15 function addTask() {
16     const taskInput = document.getElementById('task-input');
17     const taskName = taskInput.value.trim();
18
19     if (taskName === "") {
20         alert("Task cannot be empty!");
21         return;
22     }
23
24     const task = new Task(taskName);
25     tasks.push(task);
26     displayTasks();
27     taskInput.value = ''; // Kosongkan input setelah menambahkan
28 }
29
30 document.getElementById('add-task-btn').addEventListener('click', addTask);
31
```

Digunakan untuk menambahkan tugas bar uke dalam array

3. Toogle Task



```
62 //Menandai Tugas Selesai dan Menghapus Tugas
63 function toggleTask(index) {
64     tasks[index].toggleCompleted();
65     displayTasks();
66 }
```

Digunakan untuk memanggil ketika pengguna menekan checkbox untuk menandai tugas sudah selesai atau belum. Menggunakan metode toggle() dari class task untuk mengubah status completed

4. Delete Task

```
function deleteTask(index) {  
  tasks.splice(index, 1);  
  displayTasks();  
}
```

Digunakan untuk menghapus tugas

5. Display Task

```
33 //Menampilkan Tugas DOM  
34 function displayTasks() {  
35   const taskList = document.getElementById('task-list');  
36   taskList.innerHTML = '';  
37  
38   tasks.forEach((task, index) => {  
39     const taskItem = document.createElement('li');  
40     taskItem.className = task.completed ? 'completed' : '';  
41  
42     const taskText = document.createElement('span');  
43     taskText.textContent = task.name;  
44  
45     const taskCheckbox = document.createElement('input');  
46     taskCheckbox.type = 'checkbox';  
47     taskCheckbox.checked = task.completed;  
48     taskCheckbox.addEventListener('click', () => toggleTask(index));  
49  
50     const deleteButton = document.createElement('button');  
51     deleteButton.textContent = 'Delete';  
52     deleteButton.addEventListener('click', () => deleteTask(index));  
53  
54     taskItem.appendChild(taskCheckbox);  
55     taskItem.appendChild(taskText);  
56     taskItem.appendChild(deleteButton);  
57  
58     taskList.appendChild(taskItem);  
59   });  
60 }
```

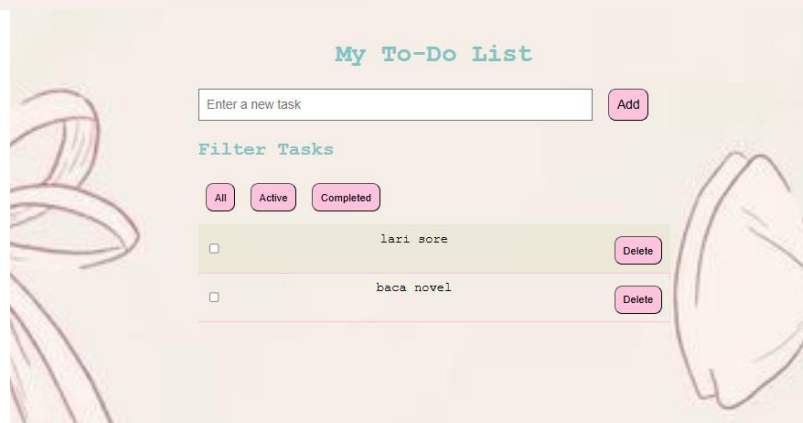
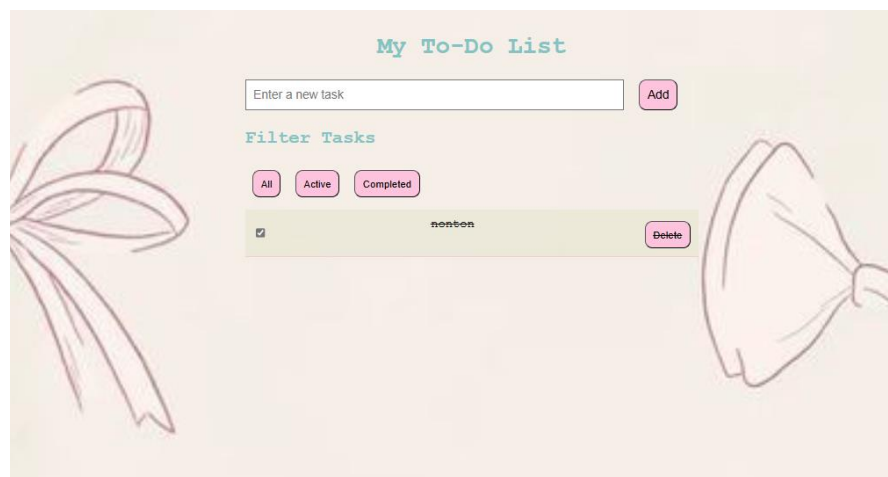
Digunakan untuk menampilkan daftar tugas didalam elemen DOM

6. Event Handling

```
93     const taskCheckbox = document.createElement('input');
94     taskCheckbox.type = 'checkbox';
95     taskCheckbox.checked = task.completed;
96     taskCheckbox.addEventListener('click', () => toggleTask(index));
97
98     const deleteButton = document.createElement('button');
99     deleteButton.textContent = 'Delete';
100    deleteButton.addEventListener('click', () => deleteTask(index));
101
```

Digunakan untuk menangani interaksi pengguna seperti menambahkan tugas, menandai tugas selesai, dan menghapus tugas.

Hasil Uji Coba



Kesimpulan

Pada praktikum ini saya jadi tahu Teknik dasar javascript contohnya DOM, event handling, dan OOP.