To-Do List Web Application Explanation

HTML (index.html)

The HTML file sets up the structure of the web application.

1. DOCTYPE and Meta Information:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>To-Do List Web App</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
```

The <!DOCTYPE html> declaration defines this document as an HTML5 document.

The <meta charset="UTF-8"> sets the character encoding for the document to UTF-8.

The <meta name="viewport" content="width=device-width, initial-scale=1.0"> ensures that the web application is responsive on all devices.

The <title> tag sets the title of the web page.

The <link> tag links the HTML document to an external CSS file (styles.css).

2. Container and Heading:

```
<div class="container">
    <h1>To-Do List Web App</h1>
```

To-Do List Web Application Explanation

A <div> with class container wraps the entire content for styling purposes.

An <h1> tag displays the heading of the application.

3. Input Section:

```
<div class="input-section">
    <input type="text" id="title" placeholder="Title" required>
    <textarea id="description" placeholder="Description" required></textarea>
    <button onclick="addTask()">Save</button>
</div>
```

This section contains an input field for the title, a textarea for the description, and a button to save the task.

The required attribute ensures that the fields are not left empty.

The button has an onclick attribute which calls the addTask() function when clicked.

4. Task List Table:

```
<table>
    <thead>
        <tr>
            <th>Title</th>
            <th>Description</th>
            <th>Delete</th>
        </tr>
    </thead>
    <tbody id="task-list">
        <!-- Tasks will be added here dynamically -->
    </tbody>
</table>
```

The <table> element contains a header (<thead>) with columns for Title, Description, and Delete.

The <tbody> with id task-list will dynamically hold the tasks.

5. Script Inclusion and Closing Tags:

```
</div>
<script src="script.js"></script>
</body>
</html>
```

The JavaScript file (script.js) is included before the closing </body> tag.

CSS (styles.css)

The CSS file styles the web application.

1. Global Styles:

```
body {
    font-family: Arial, sans-serif;
    background-color: #083d77;
    color: #fff;
    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

    margin: 0;

}
```

The body is styled with a font, background color, text color, and flex properties for centering content.

2. Container Styles:

```
.container {

    width: 80%;

    max-width: 800px;

    background-color: #fff;

    padding: 20px;

    border-radius: 8px;

    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

    color: #000;

}
```

The .container class is styled with width, background color, padding, border radius, and box shadow for a card-like appearance.

3. Heading Styles:

```
h1 {
    text-align: center;
    margin-bottom: 20px;
}
```

The h1 tag is centered and given some margin at the bottom.

4. Input Section Styles:

```
.input-section {
    display: flex;
    flex-direction: column;
    margin-bottom: 20px;
}
.input-section input, .input-section textarea {
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}
.input-section button {
    padding: 10px;
    background-color: #28a745;
    color: #fff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}
.input-section button:hover {
    background-color: #218838;
}
```

The .input-section is styled with flex direction, padding, margin, and button styles.

5. Table Styles:

```
table {
    width: 100%;
    border-collapse: collapse;
}
thead th {
    border-bottom: 2px solid #ddd;
    padding: 10px;
}
tbody td {
    border-bottom: 1px solid #ddd;
    padding: 10px;
    text-align: center;
}
tbody td button {
    background-color: #dc3545;
    color: #fff;
    border: none;
    padding: 5px 10px;
    border-radius: 4px;
    cursor: pointer;
}
tbody td button:hover {
    background-color: #c82333;
}
```

The table, thead, and tbody are styled for a clean layout, with specific styles for the delete button.

JavaScript (script.js)

The JavaScript file handles the functionality of adding and deleting tasks.

1. Add Task Function:

```
function addTask() {
    const title = document.getElementById('title').value;
    const description = document.getElementById('description').value;

    if (!title || !description) {
        alert("Please fill out this field.");
        return;
    }

    const taskList = document.getElementById('task-list');
    const row = document.createElement('tr');


    const titleCell = document.createElement('td');
    titleCell.textContent = title;
    row.appendChild(titleCell);


    const descriptionCell = document.createElement('td');
    descriptionCell.textContent = description;
    row.appendChild(descriptionCell);


    const deleteCell = document.createElement('td');
    const deleteButton = document.createElement('button');
    deleteButton.textContent = 'X';
    deleteButton.onclick = () => taskList.removeChild(row);
    deleteCell.appendChild(deleteButton);
    row.appendChild(deleteCell);


    taskList.appendChild(row);


    document.getElementById('title').value = '';
    document.getElementById('description').value = '';
}
```

The addTask function retrieves the title and description values.

To-Do List Web Application Explanation

It checks if the fields are empty and shows an alert if they are.

It creates a new row (<tr>) and cells (<td>) for the title, description, and delete button.

The delete button has an onclick event that removes the row from the table.

The new row is appended to the task-list and the input fields are cleared.