CADT IDT

W1 PRACTICE

QUIZ APP

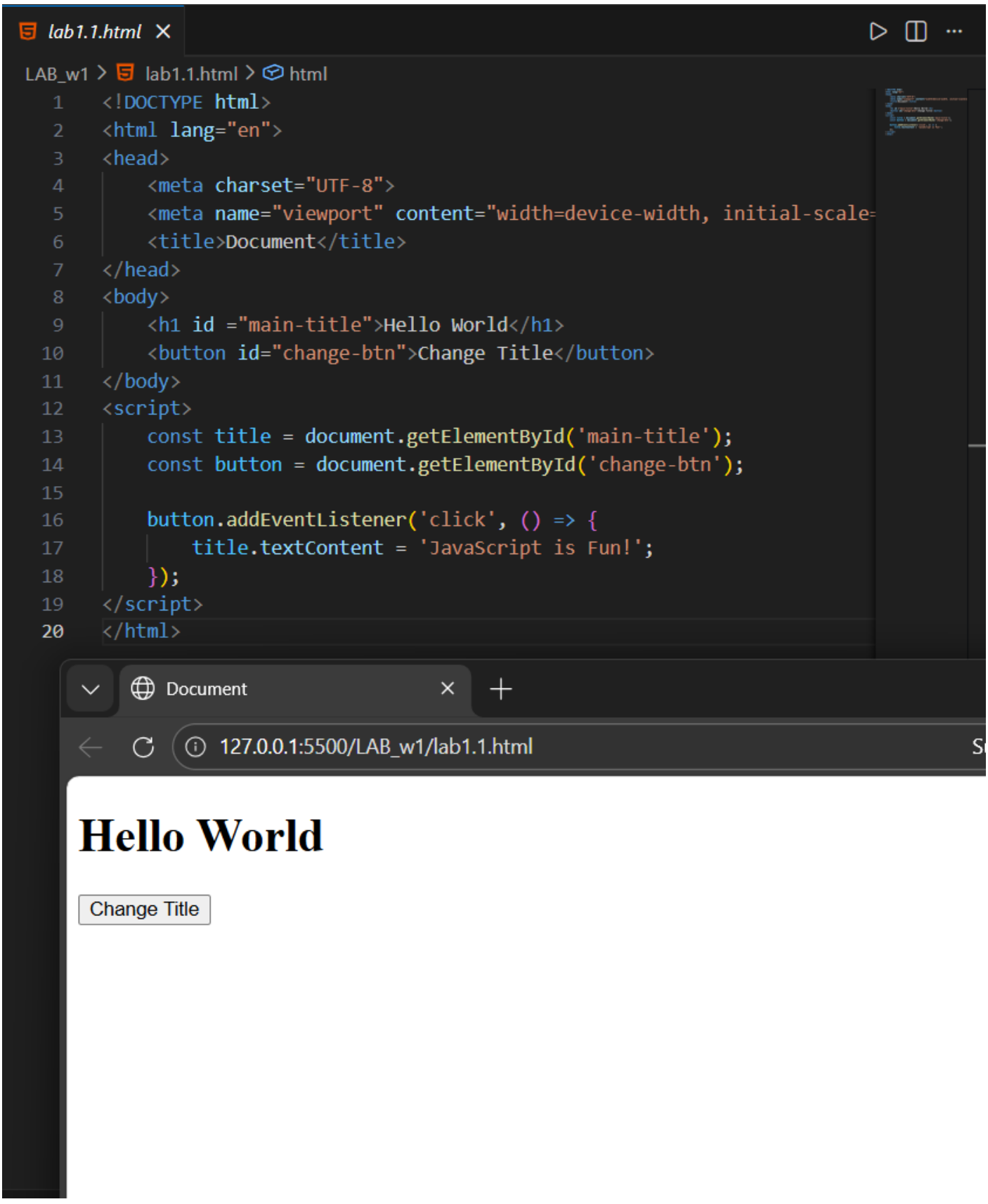
Lab1.1:

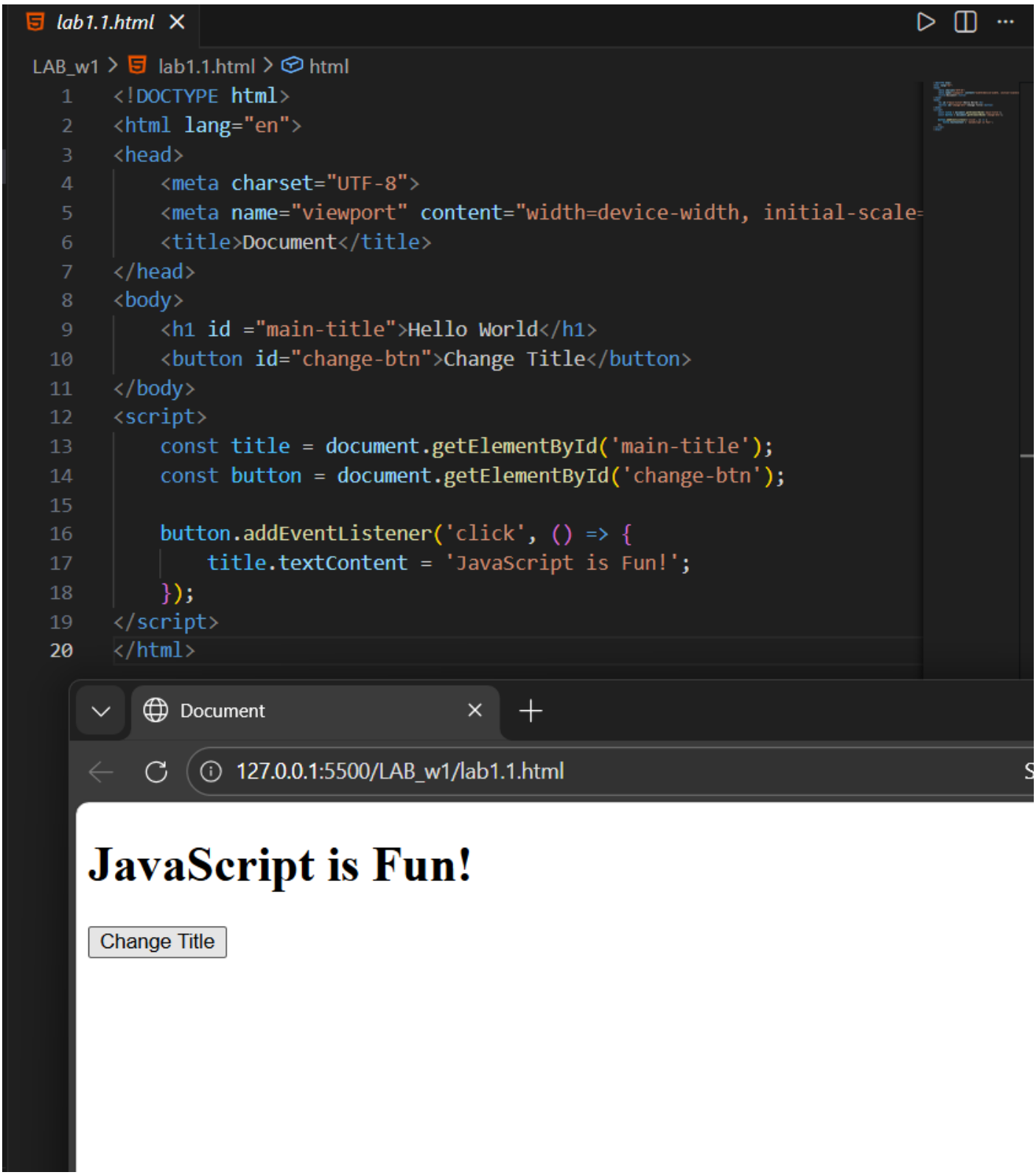
The Selector check

```
<h1 id="main-title">Hello World</h1>
<button id="change-btn">Change Title</button>
```

Todo:

1. Select the `h1` and the `button` using JavaScript.
2. Add a click event listener to the button.
3. When clicked, change the text of the `h1` to "Javascript is fun!".


lab1.1.html
LAB_w1 > lab1.1.html > html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<h1 id ="main-title">Hello World</h1>
<button id="change-btn">Change Title</button>
</body>
<script>
const title = document.getElementById('main-title');
const button = document.getElementById('change-btn');
button.addEventListener('click', () => {
title.textContent = 'JavaScript is Fun!';
});
</script>
</html>
Document
127.0.0.1:5500/LAB_w1/lab1.1.html
Hello World
Change Title



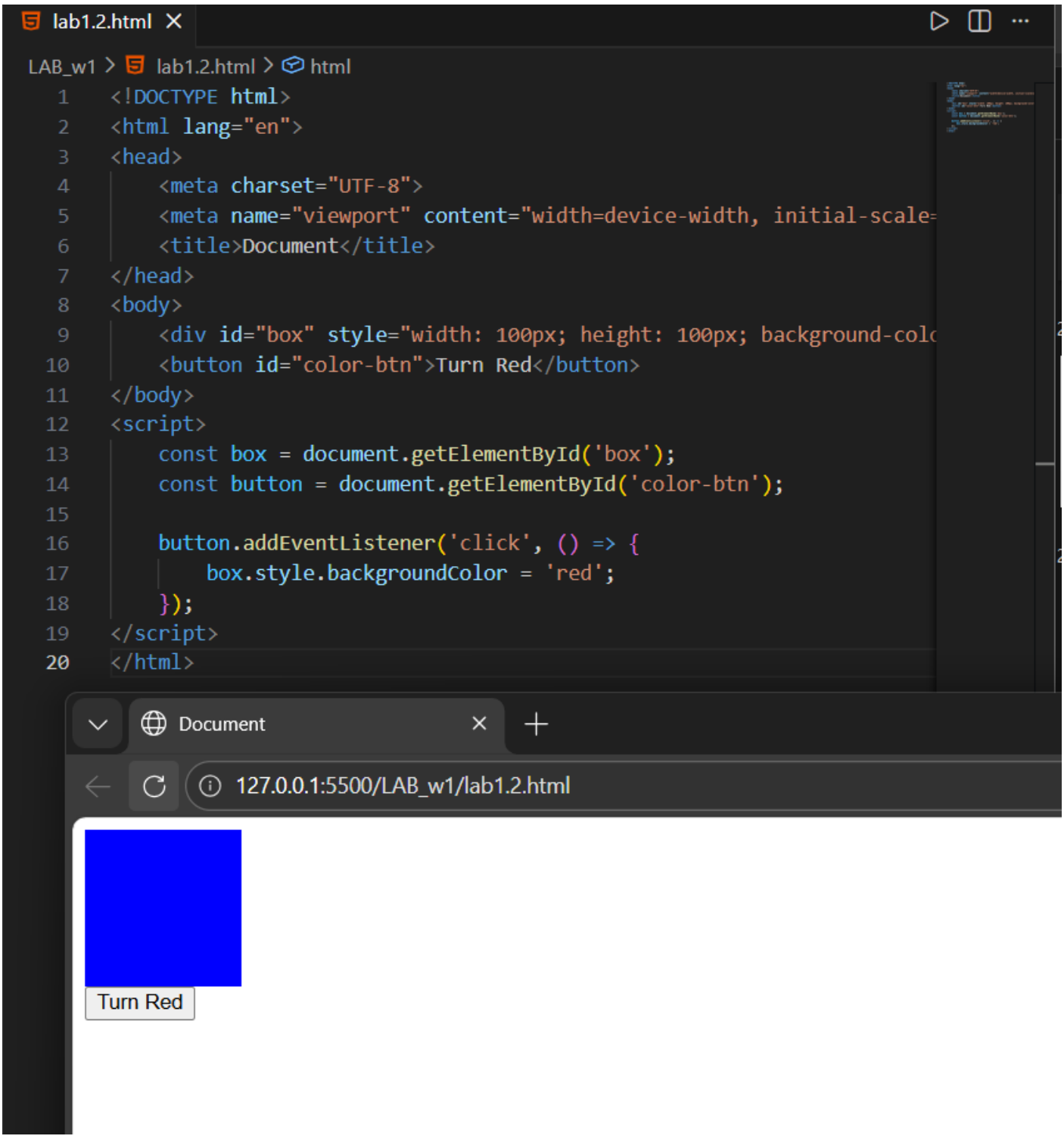
Lab1.2:

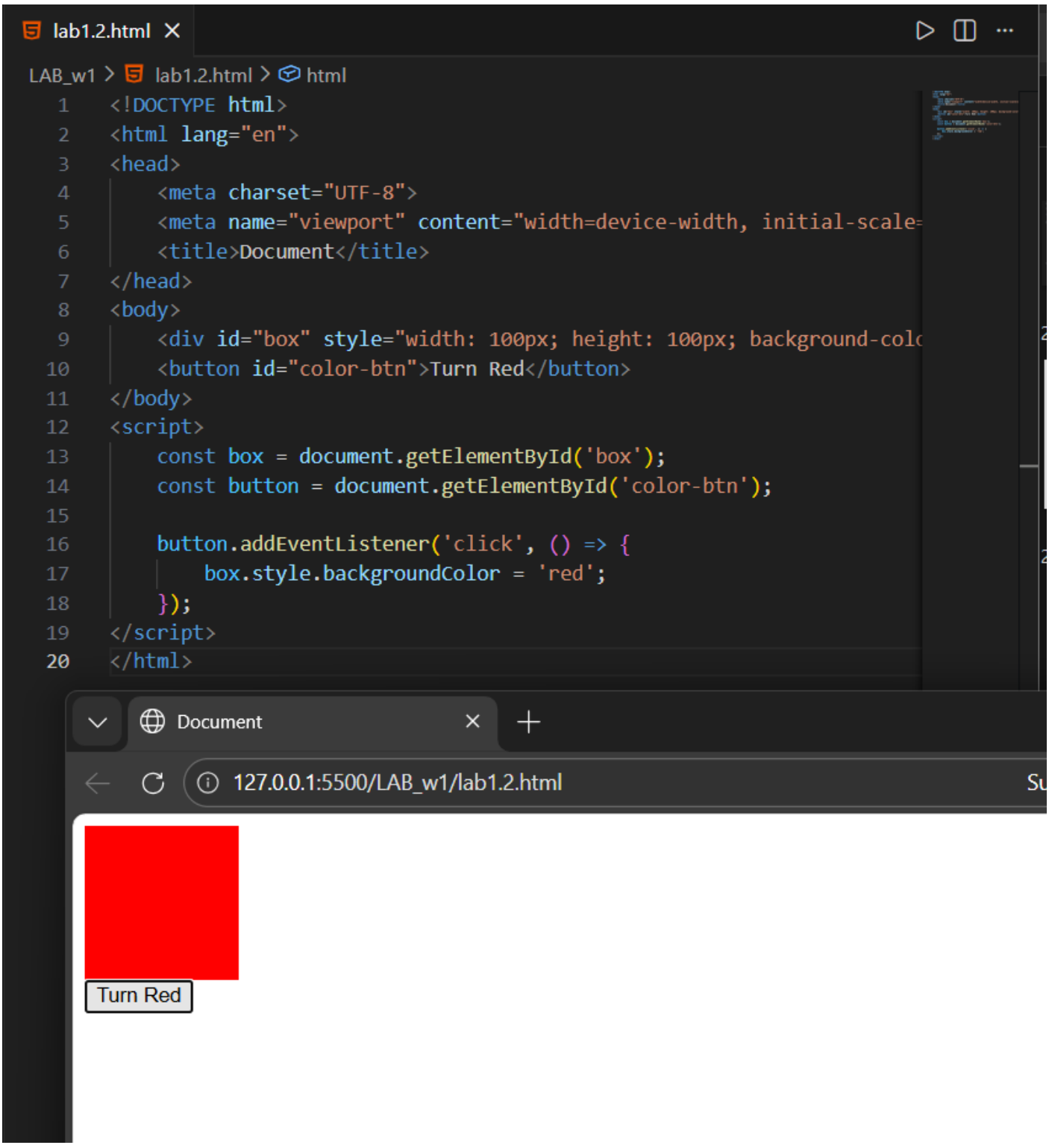
Style Manipulation

```
<div id="box" style="width: 100px; height: 100px; background-color: blue;"></div>
<button id="color-btn">Turn Red</button>
```

Todo:

1. When the button is clicked, change the background color of the box to "red".
2. Make it toggle. If it is red, turn it back to blue, and vice versa.


lab1.2.html
LAB_w1 > lab1.2.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=
6 <title>Document</title>
7 </head>
8 <body>
9 <div id="box" style="width: 100px; height: 100px; background-colo
10 <button id="color-btn">Turn Red</button>
11 </body>
12 <script>
13 const box = document.getElementById('box');
14 const button = document.getElementById('color-btn');
15
16 button.addEventListener('click', () => {
17 box.style.backgroundColor = 'red';
18 });
19 </script>
20 </html>
Document
127.0.0.1:5500/LAB_w1/lab1.2.html
Turn Red


lab1.2.html
LAB_w1 > lab1.2.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=
6 <title>Document</title>
7 </head>
8 <body>
9 <div id="box" style="width: 100px; height: 100px; background-colo
10 <button id="color-btn">Turn Red</button>
11 </body>
12 <script>
13 const box = document.getElementById('box');
14 const button = document.getElementById('color-btn');
15
16 button.addEventListener('click', () => {
17 box.style.backgroundColor = 'red';
18 });
19 </script>
20 </html>
Document
127.0.0.1:5500/LAB_w1/lab1.2.html
Turn Red

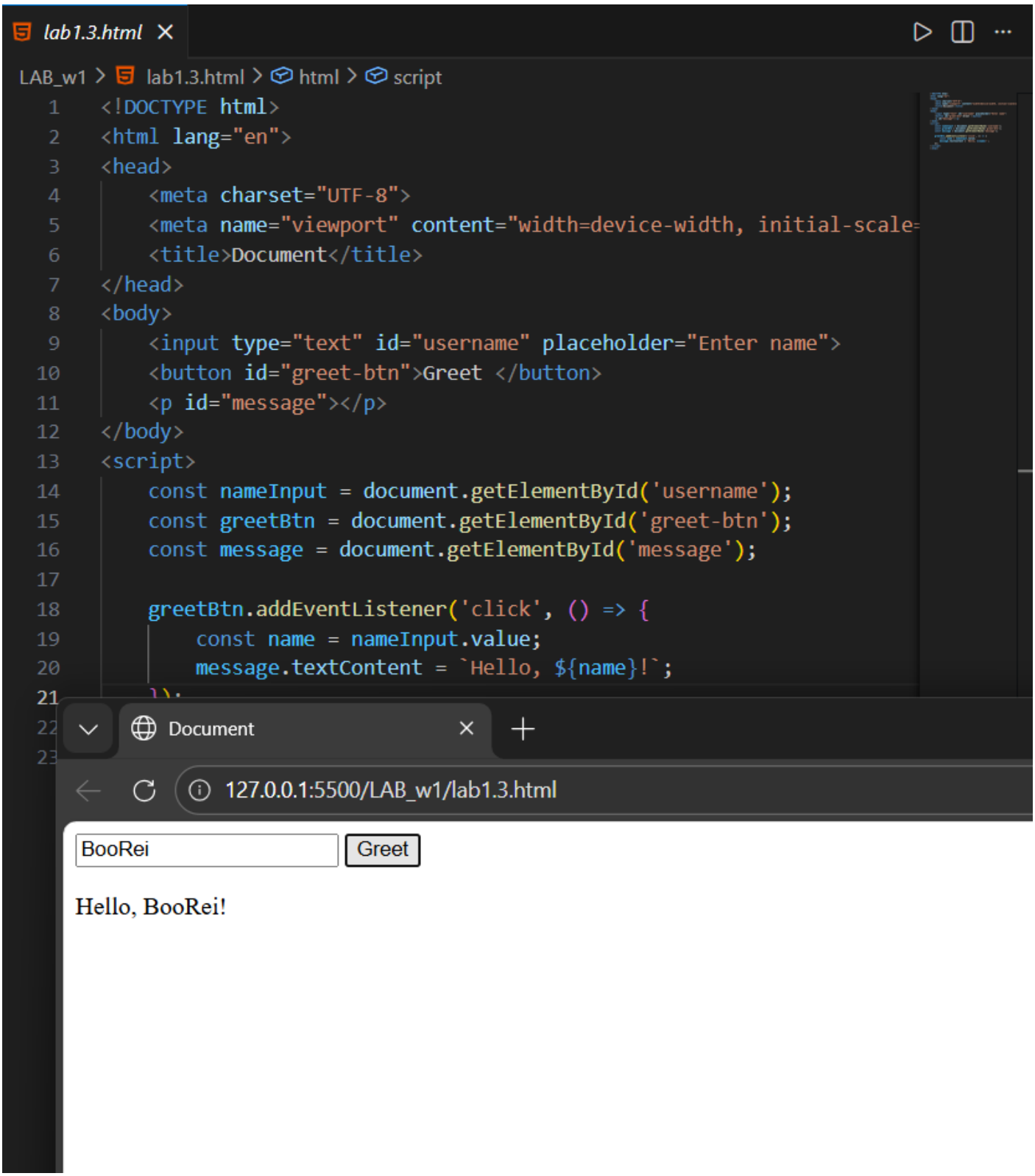
Lab1.3

Input Handling

```
<input type="text" id="username" placeholder="Enter name">
<button id="greet-btn">Greet</button>
<p id="message"></p>
```

Todo:

1. When the button is clicked, get the value typed into the input field.
2. Display "Hello, [Name]!" inside the `<p id="message">`.
3. If the input is empty, display "Please enter a name" in red color.


lab1.3.html
LAB_w1 > lab1.3.html > html > script
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<input type="text" id="username" placeholder="Enter name">
<button id="greet-btn">Greet </button>

</body>
<script>
const nameInput = document.getElementById('username');
const greetBtn = document.getElementById('greet-btn');
const message = document.getElementById('message');
greetBtn.addEventListener('click', () => {
const name = nameInput.value;
message.textContent = `Hello, ${name}!`;
});
Document
127.0.0.1:5500/LAB_w1/lab1.3.html
BooRei
Greet
Hello, BooRei!

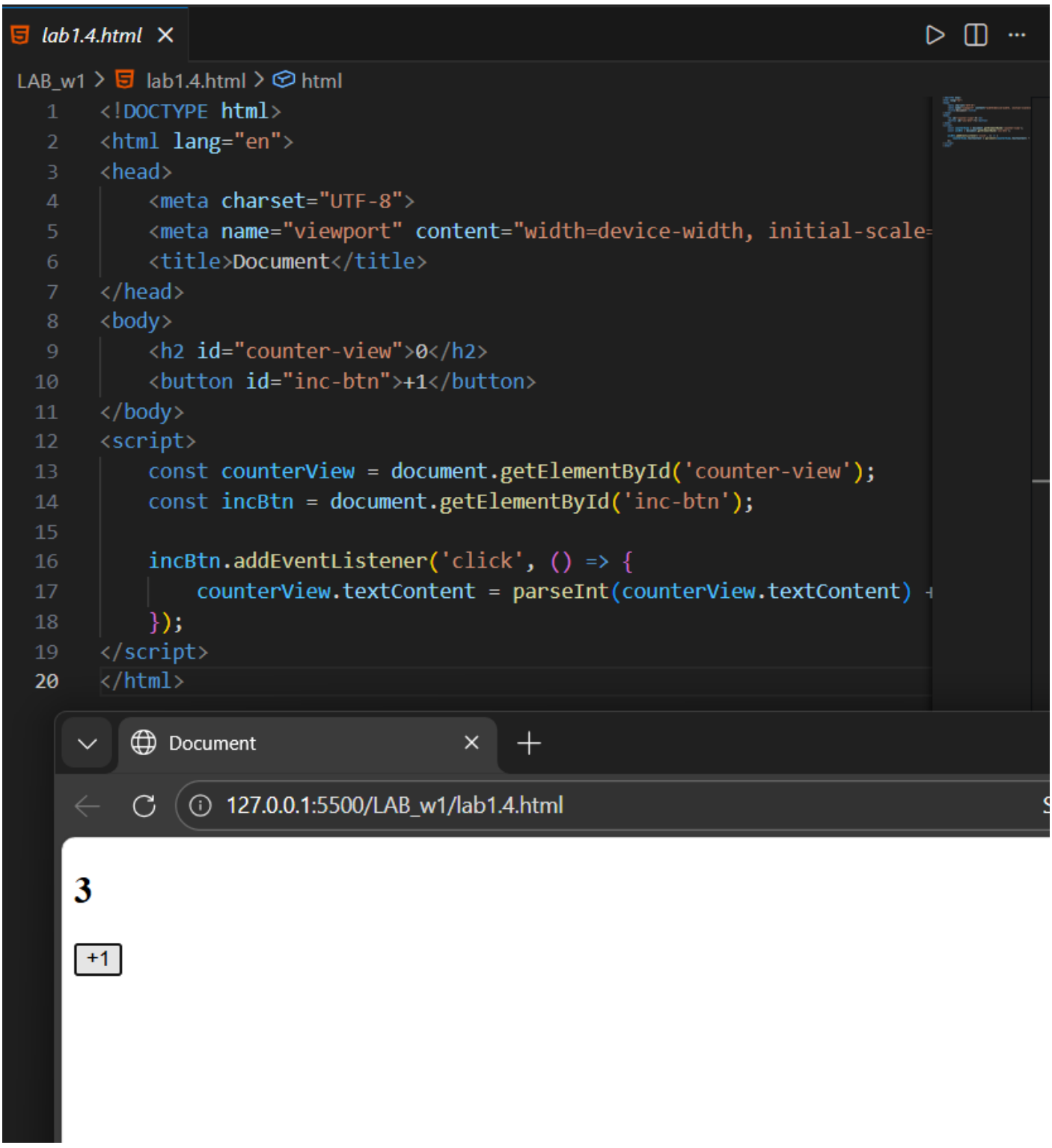
Lab1.4

The Counter (Data Driven)

```
<h2 id="counter-view">0</h2>
<button id="inc-btn">+1</button>
```

Todo:

1. Create a Javascript variable `let count = 0;` (This is your **Data**).
2. Create a function `render()` that updates `counter-view` with the value of `count`.
3. When the button is clicked:
    - Update the variable (`count++`).
    - Call `render()` to update the UI.
    - *Do NOT change the DOM directly inside the event listener.*


lab1.4.html
LAB_w1 > lab1.4.html > html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<h2 id="counter-view">0</h2>
<button id="inc-btn">+1</button>
</body>
<script>
const counterView = document.getElementById('counter-view');
const incBtn = document.getElementById('inc-btn');
incBtn.addEventListener('click', () => {
counterView.textContent = parseInt(counterView.textContent)
});
</script>
</html>
Document
127.0.0.1:5500/LAB_w1/lab1.4.html
3
+1

Lab1.5

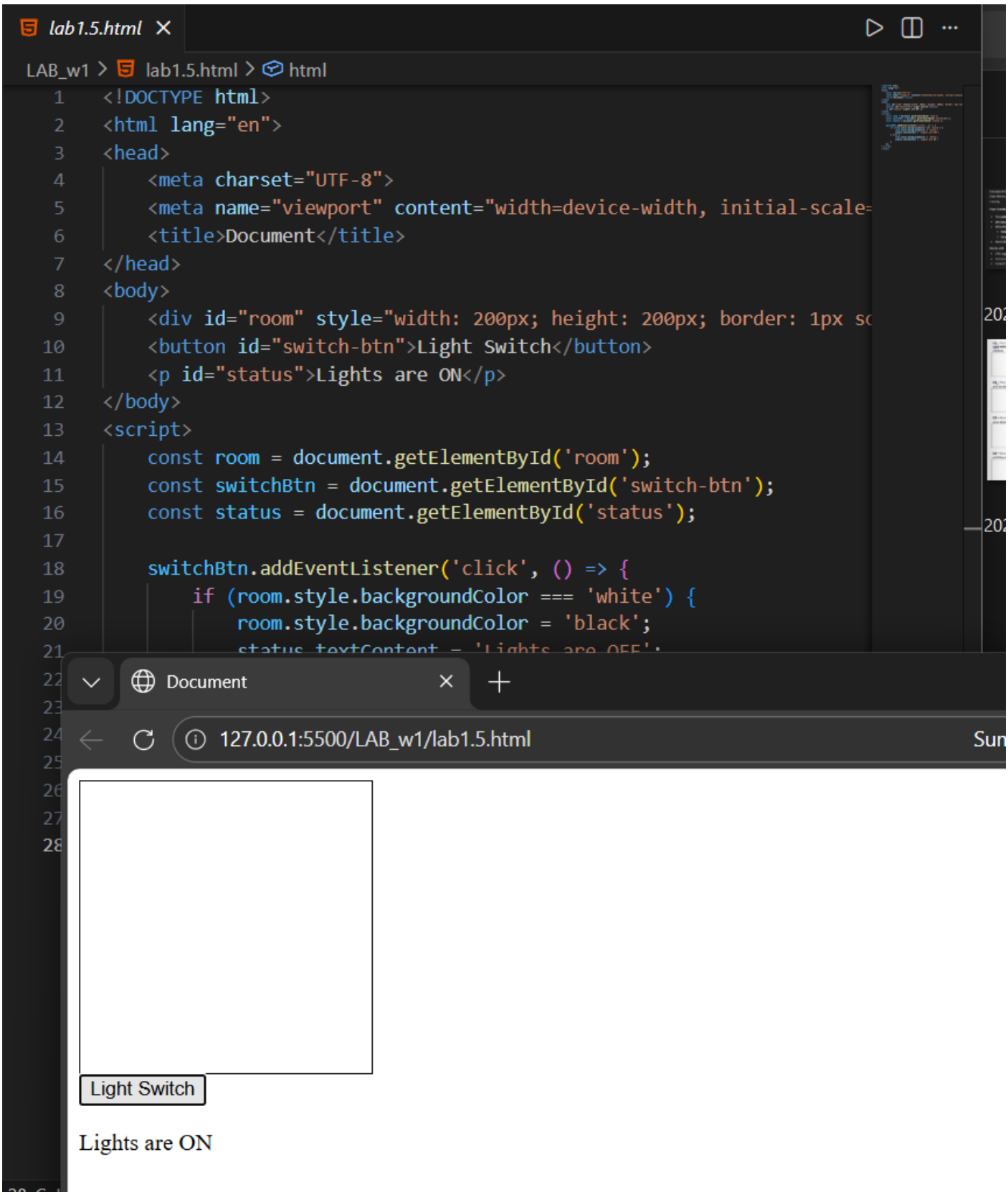
The Light Switch (Boolean Logic)

```
<div id="room" style="width: 200px; height: 200px;
border: 1px solid ■black; background-color: □white;"
></div>

<button id="switch-btn">Light Switch</button>
<p id="status-text">Lights are ON</p>
```

Todo:

1. Create a variable `let isLightOn = true;`.
2. Create a `render()` function:
   - If `isLightOn` is true: set box background to white, text to "Lights are ON".
   - If `isLightOn` is false: set box background to black, text to "Lights are OFF".
3. On button click: toggle the variable (`isLightOn = !isLightOn`) and call `render()`.


lab1.5.html
LAB_w1 > lab1.5.html > html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<div id="room" style="width: 200px; height: 200px; border: 1px sc
<button id="switch-btn">Light Switch</button>
Lights are ON
</body>
<script>
const room = document.getElementById('room');
const switchBtn = document.getElementById('switch-btn');
const status = document.getElementById('status');
switchBtn.addEventListener('click', () => {
if (room.style.backgroundColor === 'white') {
room.style.backgroundColor = 'black';
Document
127.0.0.1:5500/LAB_w1/lab1.5.html
Light Switch
Lights are ON

lab1.5.html

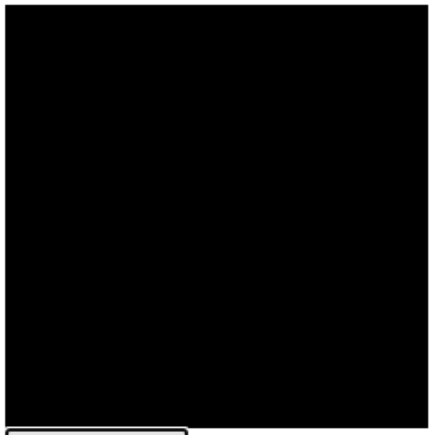
LAB_w1 > lab1.5.html > html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=
    <title>Document</title>
</head>
<body>
    <div id="room" style="width: 200px; height: 200px; border: 1px sc
    <button id="switch-btn">Light Switch</button>
    <p id="status">Lights are ON</p>
</body>
<script>
    const room = document.getElementById('room');
    const switchBtn = document.getElementById('switch-btn');
    const status = document.getElementById('status');

    switchBtn.addEventListener('click', () => {
        if (room.style.backgroundColor === 'white') {
            room.style.backgroundColor = 'black';
            status.textContent = 'Lights are OFF';
```

Document

127.0.0.1:5500/LAB_w1/lab1.5.html



Light Switch

Lights are OFF

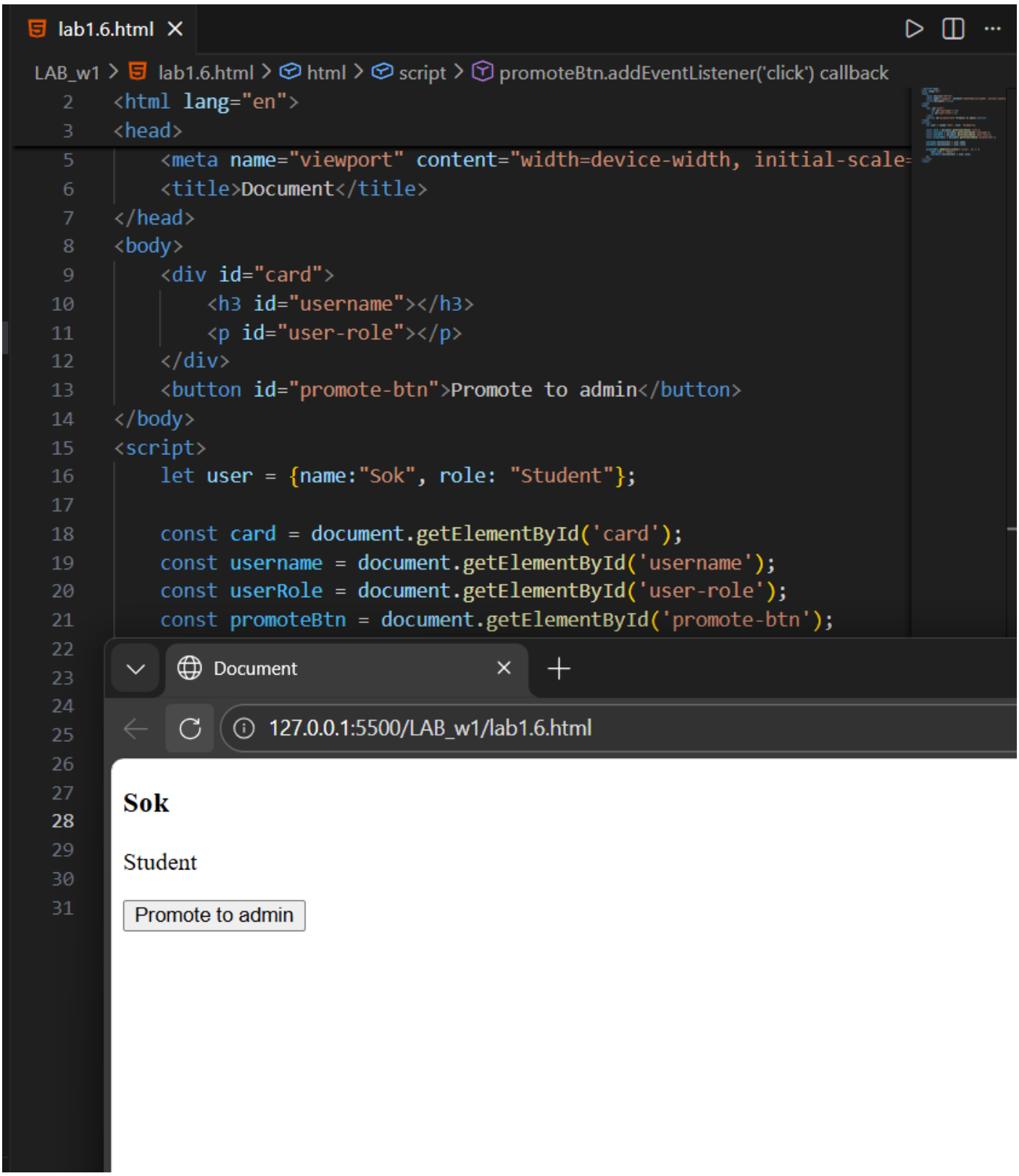
Lab1.6

Simple Object Rendering

```
<div id="card">
    <h3 id="user-name"></h3>
    <p id="user-role"></p>
</div>
<button id="promote-btn">Promote to Admin</button>
```

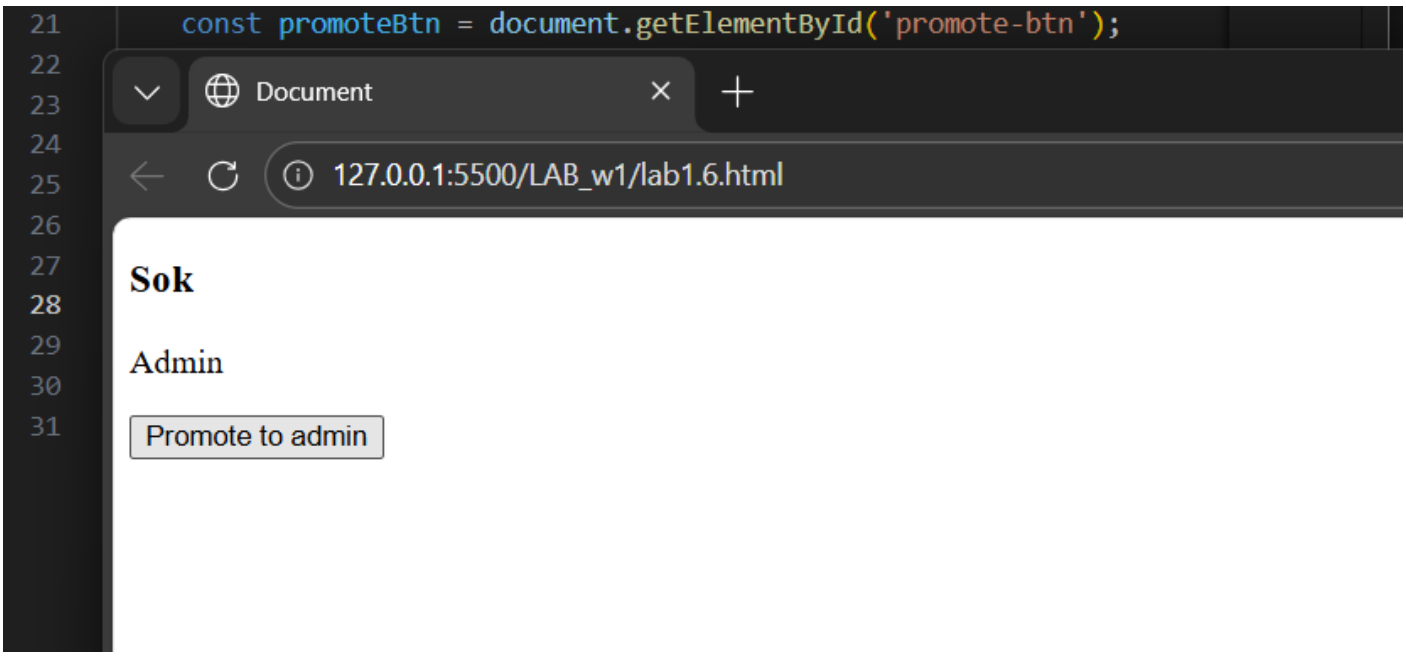
Todo:

1. Create a data object: `let user = { name: "Sok", role: "Student" };`
2. Create a `render()` function that puts the user's name and role into the HTML elements.
3. When the "Promote" button is clicked:
    - Change `user.role` to "Admin". o
      Call `render()`.


lab1.6.html
LAB_w1 > lab1.6.html > html > script > promoteBtn.addEventListener('click') callback
<html lang="en">
<head>
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>

<h3 id="username"></h3>

<button id="promote-btn">Promote to admin</button>
</body>
<script>
let user = {name:"Sok", role: "Student"};
const card = document.getElementById('card');
const username = document.getElementById('username');
const userRole = document.getElementById('user-role');
const promoteBtn = document.getElementById('promote-btn');
Document
127.0.0.1:5500/LAB_w1/lab1.6.html
Sok
Student
Promote to admin



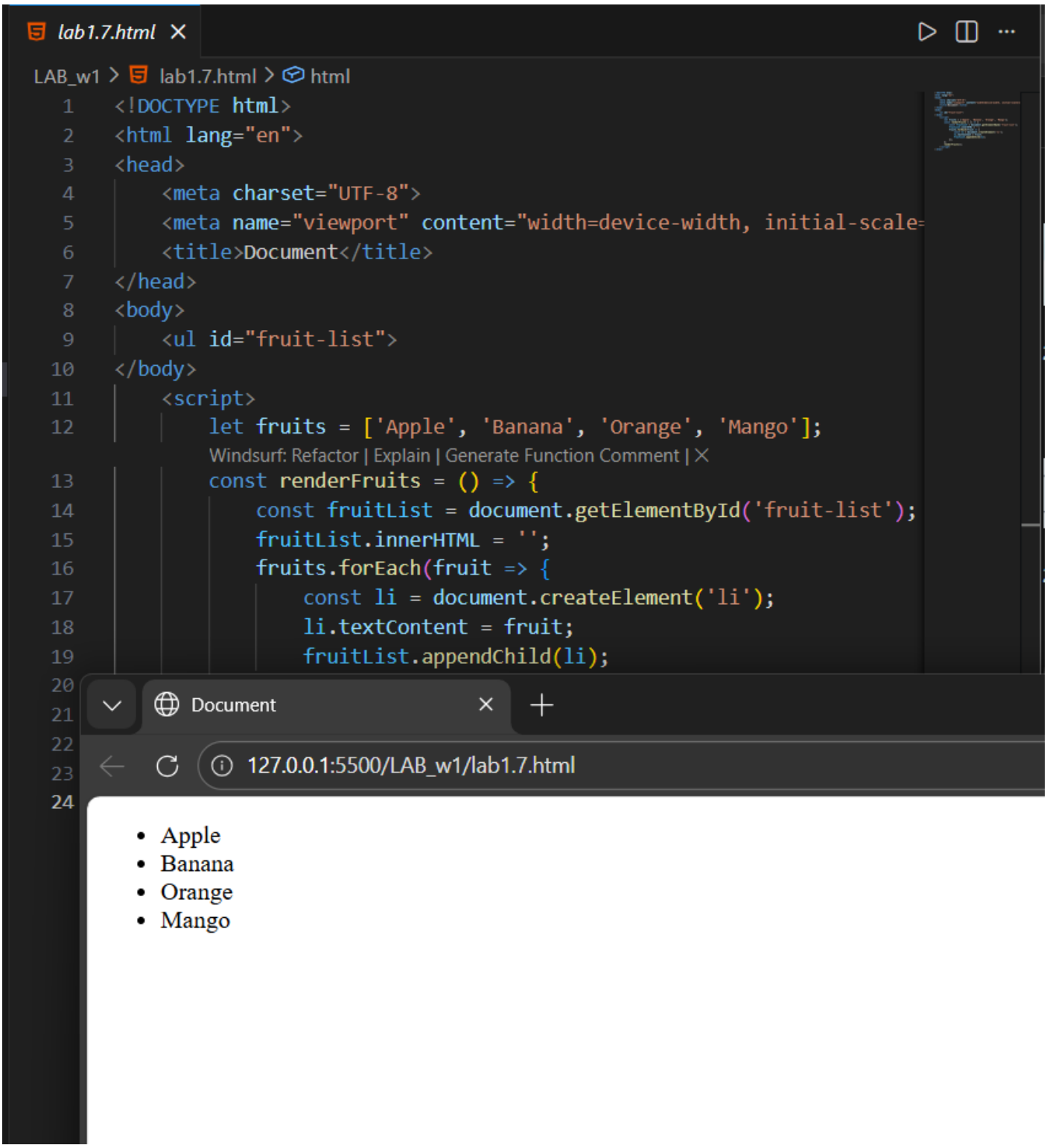
Lab1.7

Rendering a List (Static)

```
<ul id="fruit-list"></ul>
```

Todo:

1. Start with this data: `let fruits = ["Apple", "Banana", "Orange", "Mango"];` 2. Write a function `renderFruits()` that:
    - Clears the current innerHTML of `fruit-list`. o Loops through the `fruits` array.
    - For each fruit, creates an `<li>` element. o Appends the `<li>` to the `<ul>`.
3. Call the function once at the start to display the list.


lab1.7.html
LAB_w1 > lab1.7.html > html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<ul id="fruit-list">
</body>
<script>
let fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
Windsurf: Refactor | Explain | Generate Function Comment |
const renderFruits = () => {
const fruitList = document.getElementById('fruit-list');
fruitList.innerHTML = '';
fruits.forEach(fruit => {
const li = document.createElement('li');
li.textContent = fruit;
fruitList.appendChild(li);
Document
127.0.0.1:5500/LAB_w1/lab1.7.html
Apple
Banana
Orange
Mango

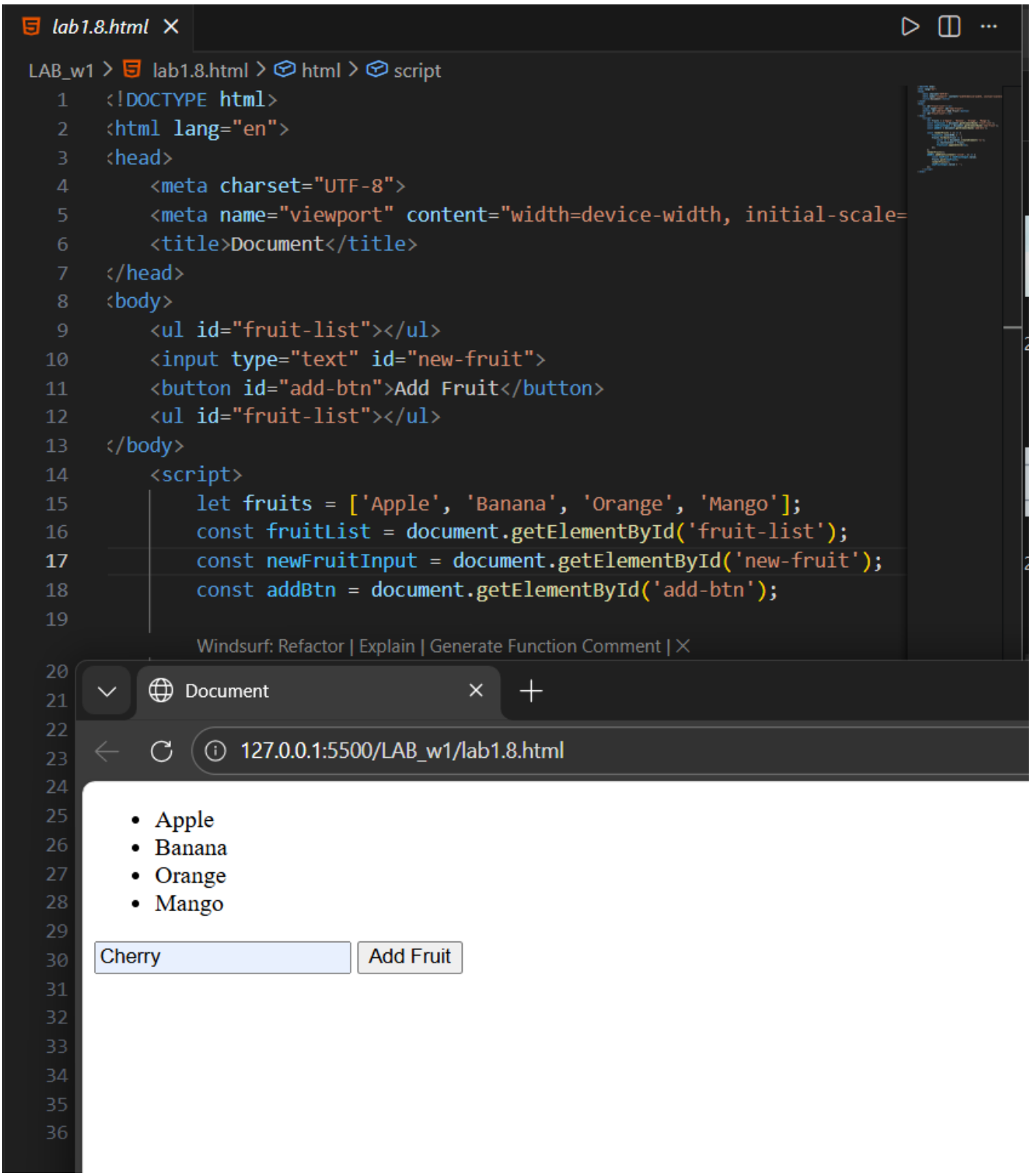
Lab1.8

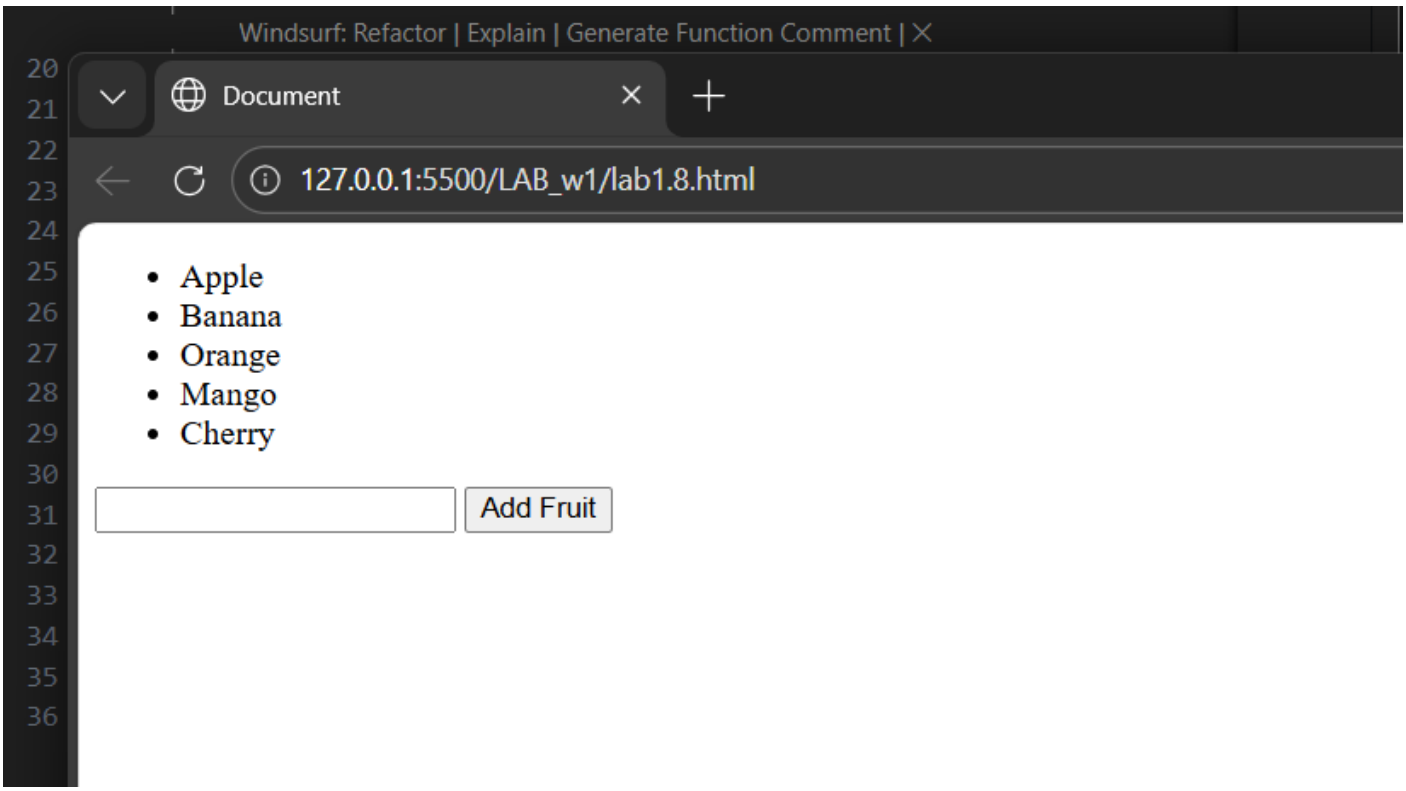
Adding to a List (Dynamic)

```
<input type="text" id="new-fruit">
<button id="add-btn">Add Fruit</button>
<ul id="fruit-list"></ul>
```

Todo:

1. Use the code from Exercise 7.
2. When the "Add Fruit" button is clicked:
    - Get the text from the input.
    - `.push()` the new text into the `fruits` array.
    - Call `renderFruits()`.o *Observation: Notice how you don't need to manually create the new DOM element in the click event? The render function handles the whole list.*


lab1.8.html
LAB_w1 > lab1.8.html > html > script
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=
<title>Document</title>
</head>
<body>
<ul id="fruit-list">
<input type="text" id="new-fruit">
<button id="add-btn">Add Fruit</button>
<ul id="fruit-list">
</body>
<script>
let fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
const fruitList = document.getElementById('fruit-list');
const newFruitInput = document.getElementById('new-fruit');
const addBtn = document.getElementById('add-btn');
Windsurf: Refactor | Explain | Generate Function Comment |
Document
127.0.0.1:5500/LAB_w1/lab1.8.html
Apple
Banana
Orange
Mango
Cherry
Add Fruit



Lab1.9

Smart List (Objects + Style)

```
<div id="tasks-container"></div>
```

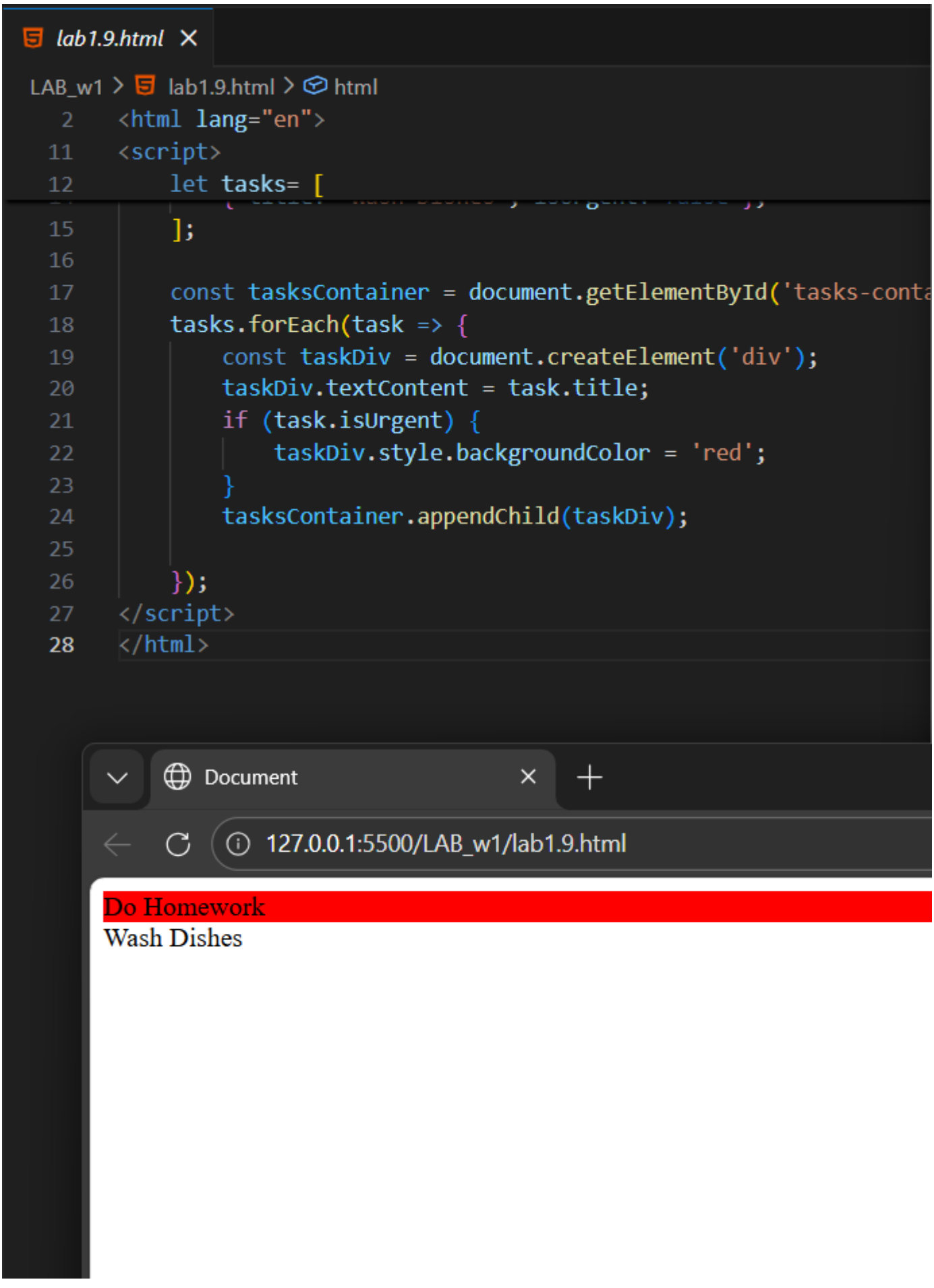
Task:

1. Data:

   JavaScript

   ```
   let tasks = [
     { title: "Do Homework", isUrgent: true },
     { title: "Wash dishes", isUrgent: false } ];
   ```

2. Create a `renderTasks()` function.
3. Loop through tasks and create `<div>` elements for each.
4. **Logic:** If `isUrgent` is `true`, set the text color to **red**. If `false`, set it to **black**.
5. Append them to the container.

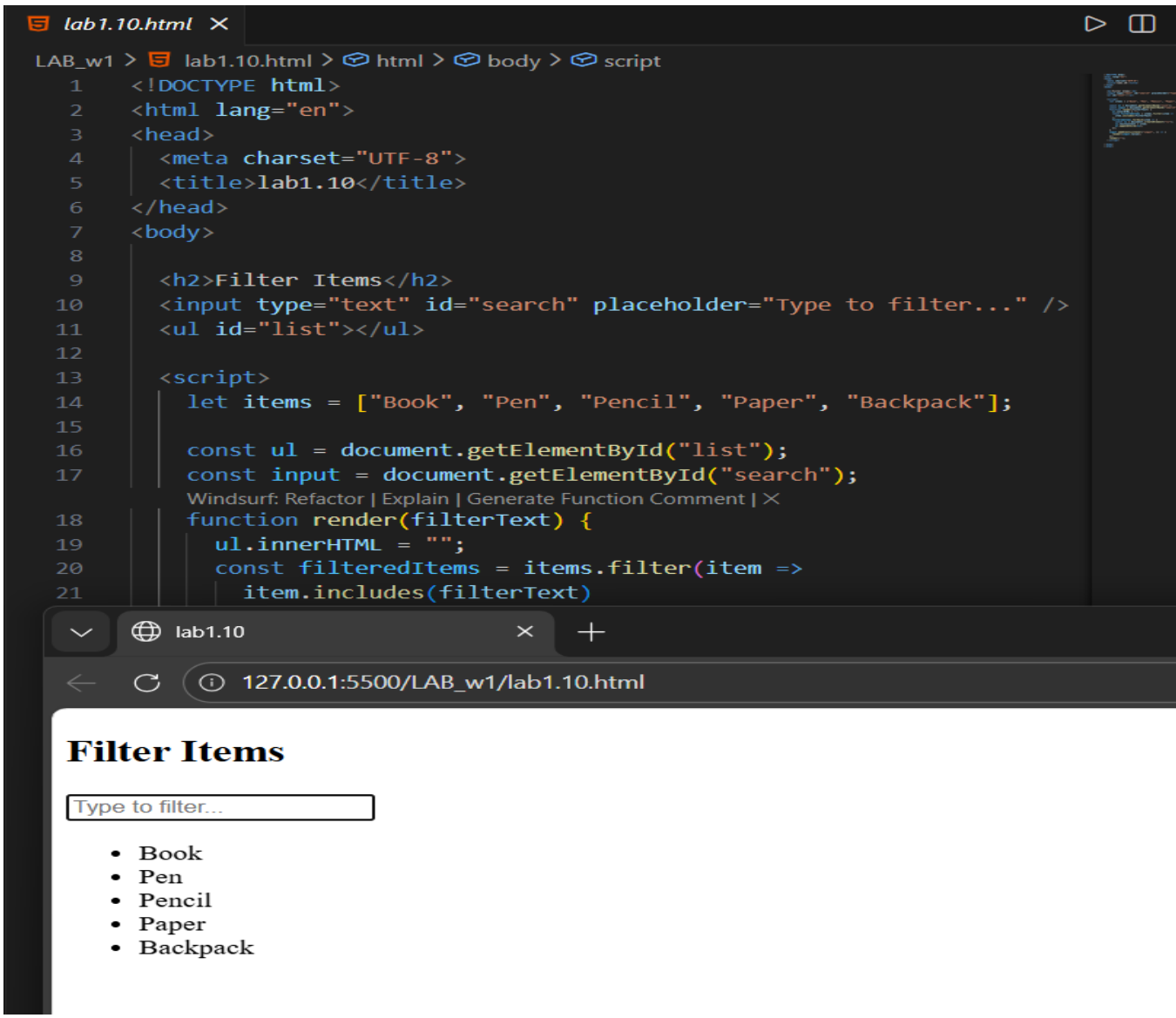

lab1.9.html
LAB_w1 > lab1.9.html > html
2 <html lang="en">
11 <script>
12 let tasks= [
15];
16
17 const tasksContainer = document.getElementById('tasks-conta
18 tasks.forEach(task => {
19 const taskDiv = document.createElement('div');
20 taskDiv.textContent = task.title;
21 if (task.isUrgent) {
22 taskDiv.style.backgroundColor = 'red';
23 }
24 tasksContainer.appendChild(taskDiv);
25
26 });
27 </script>
28 </html>
Document
127.0.0.1:5500/LAB_w1/lab1.9.html
Do Homework
Wash Dishes

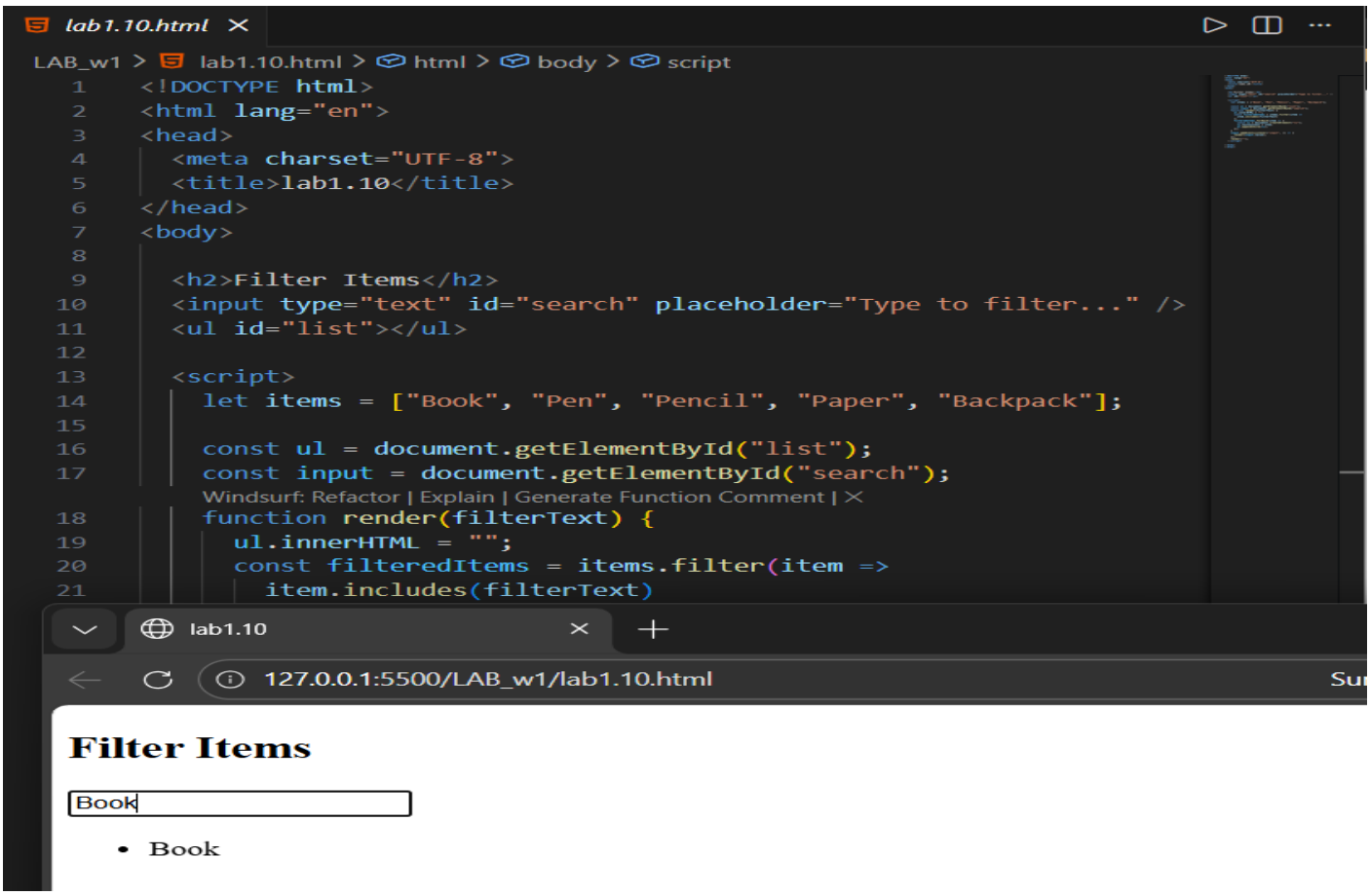
Lab1.10

The Search Filter

```
<input type="text" id="search-input" placeholder="Search item...">
<ul id="items-list"></ul>
```

Task:

1. Data: `let items = ["Book", "Pen", "Pencil", "Paper", "Backpack"];` 2. Write a function `render(filterText)` that receives a filter argument.
3. Inside the function:
    - Filter the `items` array so it only contains items that include the `filterText`. o Render the filtered list to the `<ul>`.
4. Add an event listener to the input (event type: `input` or `keyup`).
5. On event: Call `render(input.value)`.
6. 



======== THE QUIZ APP ======

During this practice you will implement a **QUIZ APP**, following the bellow requirements:

Quiz Player

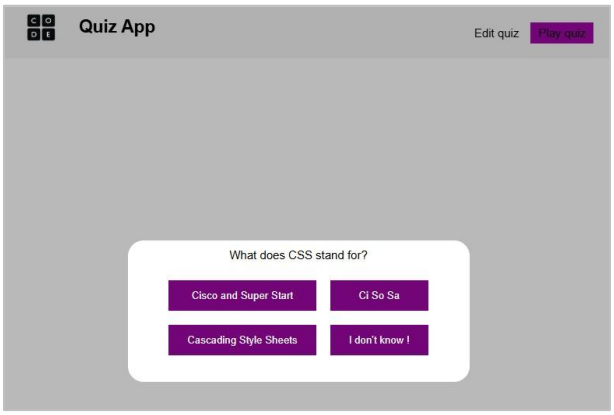
- Start a quiz
- Answer multiple questions
- Navigate through questions
- See the final score at the end

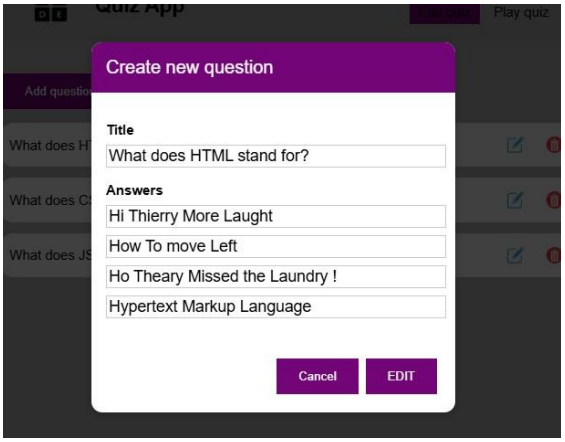
Quiz Editor

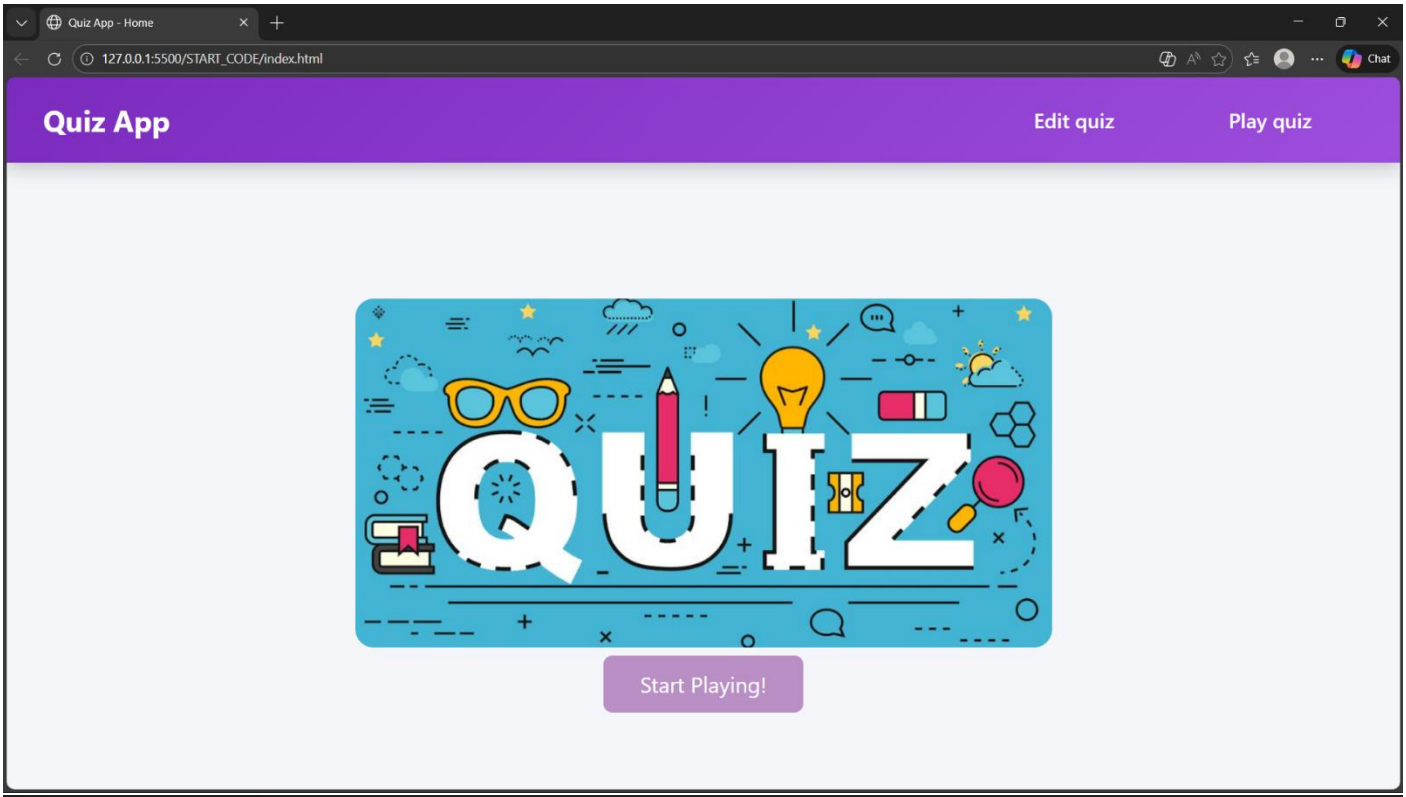
- View all quiz questions
- Edit existing questions
- Add new questions
- (Optional) Delete questions

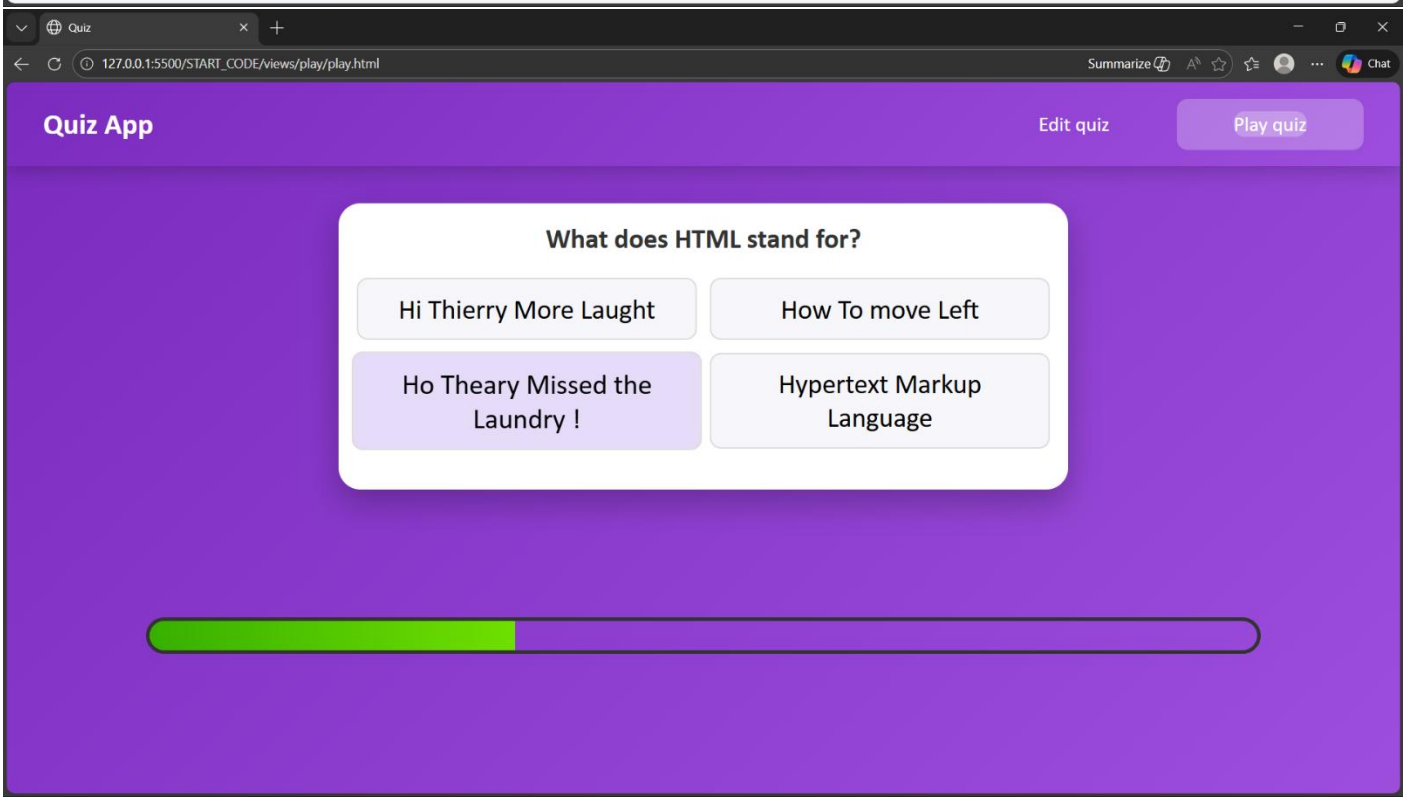
Navigation

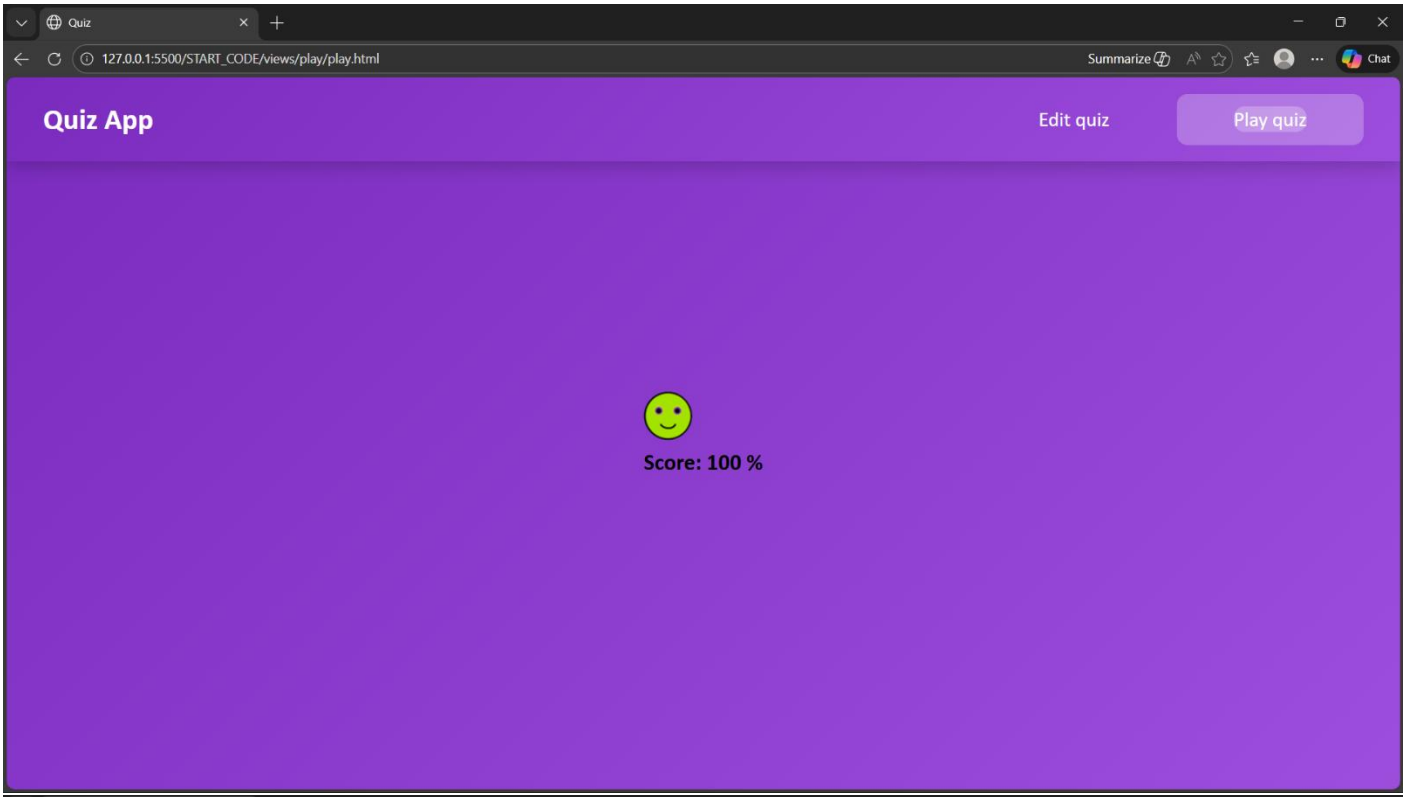
- Switch between views

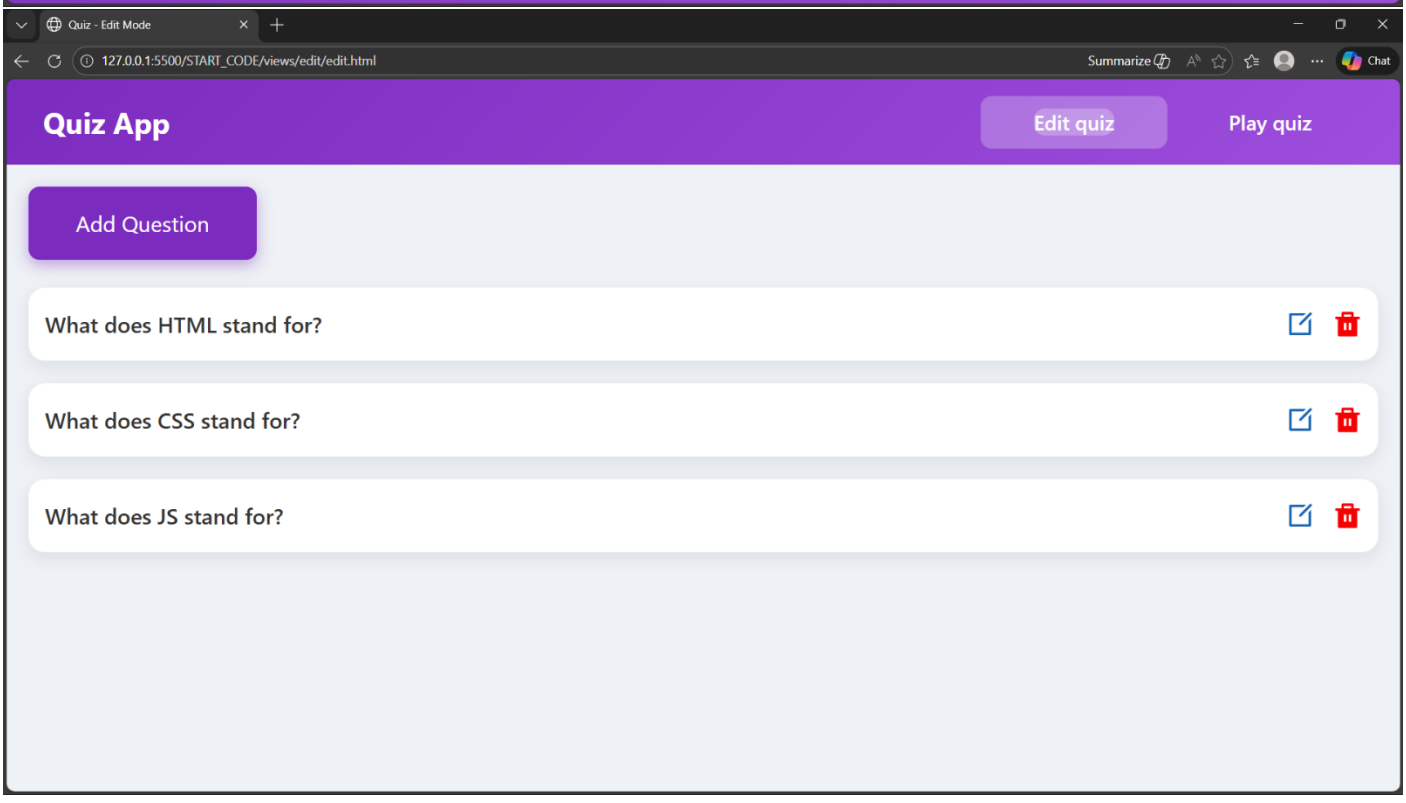


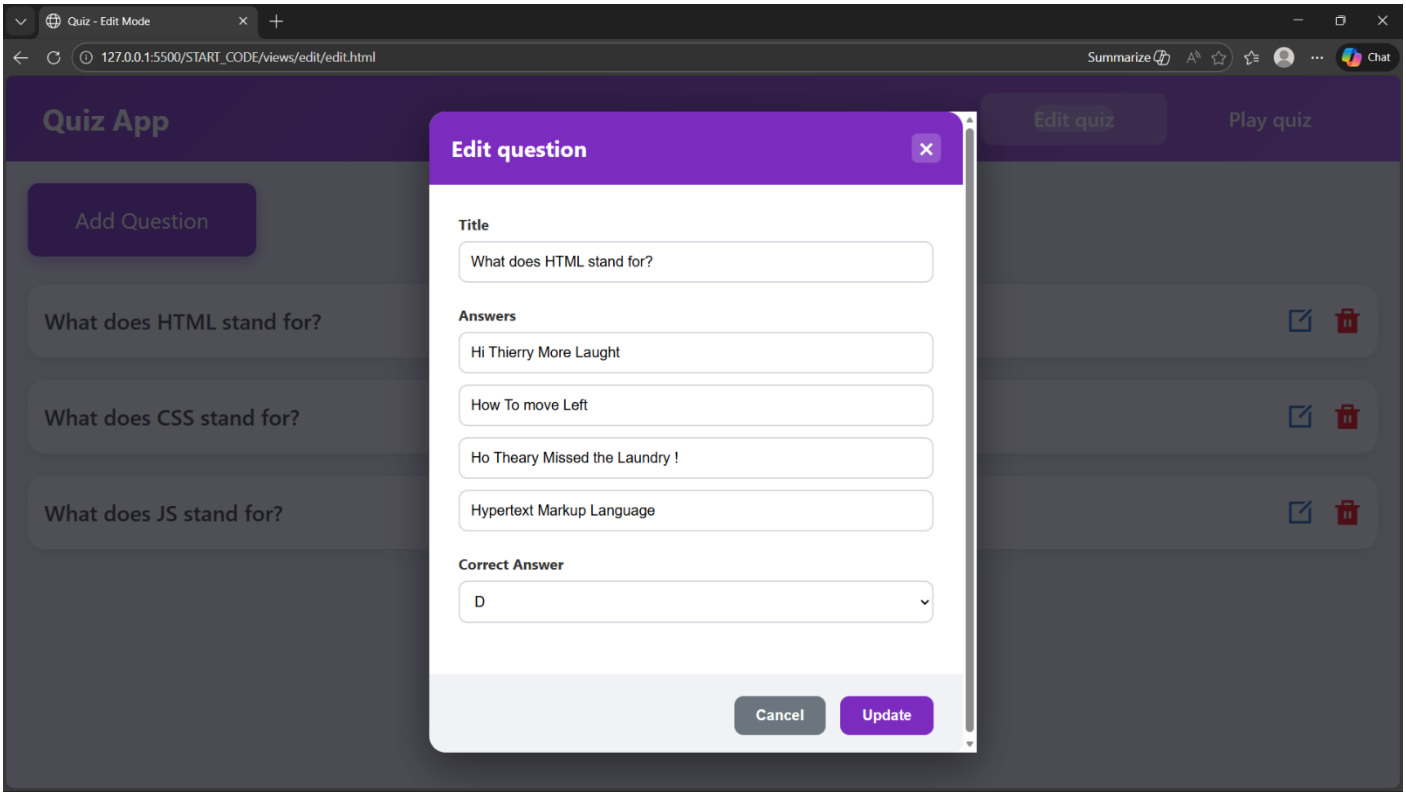



Quiz App - Home
127.0.0.1:5500/START_CODE/index.html
Chat
Quiz App
Edit quiz
Play quiz
QUIZ
Start Playing!


Quiz
127.0.0.1:5500/START_CODE/views/play/play.html
Summarize
Chat
Quiz App
Edit quiz
Play quiz
What does HTML stand for?
Hi Thierry More Laught
How To move Left
Ho Theary Missed the Laundry !
Hypertext Markup Language


Quiz
127.0.0.1:5500/START_CODE/views/play/play.html
Summarize
Chat
Quiz App
Edit quiz
Play quiz
Score: 100 %


Quiz - Edit Mode
127.0.0.1:5500/START_CODE/views/edit/edit.html
Summarize
Chat
Quiz App
Edit quiz
Play quiz
Add Question
What does HTML stand for?
What does CSS stand for?
What does JS stand for?


Quiz - Edit Mode
127.0.0.1:5500/START_CODE/views/edit/edit.html
Summarize
Chat
Quiz App
Edit quiz
Play quiz
Add Question
What does HTML stand for?
What does CSS stand for?
What does JS stand for?
Edit question
Title
What does HTML stand for?
Answers
Hi Thierry More Laught
How To move Left
Ho Theary Missed the Laundry !
Hypertext Markup Language
Correct Answer
D
Cancel
Update