

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 X

▶ ⌂ ...

```
LAB_w1 > lab1.1.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">
6      <title>Document</title>
7  </head>
8  <body>
9      <h1 id="main-title">Hello World</h1>
10     <button id="change-btn">Change Title</button>
11 </body>
12 <script>
13     const title = document.getElementById('main-title');
14     const button = document.getElementById('change-btn');
15
16     button.addEventListener('click', () => {
17         title.textContent = 'JavaScript is Fun!';
18     });
19 </script>
20 </html>
```



Document



127.0.0.1:5500/LAB_w1/lab1.1.html

S

Hello World

Change Title

The screenshot shows a code editor and a browser window. The code editor tab is labeled "lab1.1.html". The code itself is an HTML file with embedded JavaScript. It includes meta tags for charset and viewport, a title, and a body containing an h1 element with id "main-title" and a button with id "change-btn". The script part of the file uses document.getElementById to select these elements and adds an event listener to the button that changes the h1 element's text content to "JavaScript is Fun!". Below the code editor is a browser window. The address bar shows "Document" and the URL "127.0.0.1:5500/LAB_w1/lab1.1.html". The browser displays the rendered HTML with the updated text content.

```
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>Document</title>
7  </head>
8  <body>
9      <h1 id = "main-title">Hello World</h1>
10     <button id="change-btn">Change Title</button>
11  </body>
12  <script>
13      const title = document.getElementById('main-title');
14      const button = document.getElementById('change-btn');
15
16      button.addEventListener('click', () => {
17          title.textContent = 'JavaScript is Fun!';
18      });
19  </script>
20  </html>
```

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 X

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=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <div id="box" style="width: 100px; height: 100px; background-color: blue;">
10         <button id="color-btn">Turn Red</button>
11     </div>
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

A screenshot of a web browser window. The title bar says "Document" and the address bar shows the URL "127.0.0.1:5500/LAB_w1/lab1.2.html". The main content area displays a blue square with a white border and a button below it labeled "Turn Red".

lab1.2.html X

▶ ⌂ ...

```
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=1">
6      <title>Document</title>
7  </head>
8  <body>
9      <div id="box" style="width: 100px; height: 100px; background-color: black;>
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>
```

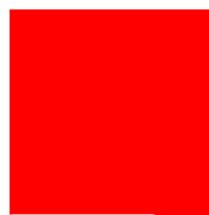


Document



127.0.0.1:5500/LAB_w1/lab1.2.html

Su



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 X ▶ ⌂ ⌂ ...

LAB_w1 > lab1.3.html > html > script

```
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">
6      <title>Document</title>
7  </head>
8  <body>
9      <input type="text" id="username" placeholder="Enter name">
10     <button id="greet-btn">Greet </button>
11     <p id="message"></p>
12 </body>
13 <script>
14     const nameInput = document.getElementById('username');
15     const greetBtn = document.getElementById('greet-btn');
16     const message = document.getElementById('message');
17
18     greetBtn.addEventListener('click', () => {
19         const name = nameInput.value;
20         message.textContent = `Hello, ${name}!`;
21     });
22
23
```

▼ Document × +

← ⌂ ⓘ 127.0.0.1:5500/LAB_w1/lab1.3.html

BooRei

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:
 - o Update the variable (`count++`).
 - o Call `render()` to update the UI.
 - o *Do NOT change the DOM directly inside the event listener.*

lab1.4.html X

LAB_w1 > lab1.4.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>Document</title>
7  </head>
8  <body>
9      <h2 id="counter-view">0</h2>
10     <button id="inc-btn">+1</button>
11  </body>
12 <script>
13     const counterView = document.getElementById('counter-view');
14     const incBtn = document.getElementById('inc-btn');
15
16     incBtn.addEventListener('click', () => {
17         counterView.textContent = parseInt(counterView.textContent) + 1;
18     });
19 </script>
20 </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:
 - o If `isLightOn` is true: set box background to white, text to "Lights are ON".
 - o 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 X

LAB_w1 > lab1.5.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>Document</title>
7  </head>
8  <body>
9      <div id="room" style="width: 200px; height: 200px; border: 1px solid black; background-color: white;">
```

lab1.5.html X

LAB_w1 > lab1.5.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>Document</title>
7  </head>
8  <body>
9      <div id="room" style="width: 200px; height: 200px; border: 1px solid black; background-color: white; position: relative; margin: auto; margin-top: 100px;">
10     <button id="switch-btn">Light Switch</button>
11     <p id="status">Lights are ON</p>
12 </body>
13 <script>
14     const room = document.getElementById('room');
15     const switchBtn = document.getElementById('switch-btn');
16     const status = document.getElementById('status');
17
18     switchBtn.addEventListener('click', () => {
19         if (room.style.backgroundColor === 'white') {
20             room.style.backgroundColor = 'black';
21             status.textContent = 'Lights are OFF';
22         } else {
23             room.style.backgroundColor = 'white';
24             status.textContent = 'Lights are ON';
25         }
26     });
27
28 </script>
```

Document 127.0.0.1:5500/LAB_w1/lab1.5.html

The screenshot shows a browser window with the URL '127.0.0.1:5500/LAB_w1/lab1.5.html'. The page content consists of a large black square with a white border, centered on the page. Below the square is a button labeled 'Light Switch'. To the right of the button is a paragraph element with the text 'Lights are OFF'. The browser's address bar and tabs are visible at the top.

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:
 - o Change `user.role` to "Admin". o
 - Call `render()`.

The screenshot shows a browser developer tools window with the following details:

- Title Bar:** lab1.6.html X
- DOM Tree:** LAB_w1 > lab1.6.html > html > script > promoteBtn.addEventListener('click') callback
- Script Content:**

```
2  <html lang="en">
3  <head>
4      <meta name="viewport" content="width=device-width, initial-scale=1">
5      <title>Document</title>
6  </head>
7  <body>
8      <div id="card">
9          <h3 id="username"></h3>
10         <p id="user-role"></p>
11     </div>
12     <button id="promote-btn">Promote to admin</button>
13 </body>
14 <script>
15     let user = {name:"Sok", role: "Student"};
16
17
18     const card = document.getElementById('card');
19     const username = document.getElementById('username');
20     const userRole = document.getElementById('user-role');
21     const promoteBtn = document.getElementById('promote-btn');
```
- Network Tab:** Document
- Address Bar:** 127.0.0.1:5500/LAB_w1/lab1.6.html
- Page Preview:** Shows the HTML structure with "Sok" and "Student" text and a "Promote to admin" button.

```
21 const promoteBtn = document.getElementById('promote-btn');
22
23 < Document >
24
25 ← ⌂ 127.0.0.1:5500/LAB_w1/lab1.6.html
26
27
28
29
30
31
```

Sok

Admin

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:
 - o Clears the current innerHTML of `fruit-list`.
 - o Loops through the `fruits` array.
 - o For each fruit, creates an `` element.
 - o Appends the `` to the ``.
3. Call the function once at the start to display the list.

lab1.7.html X

▶ ⌂ ...

LAB_w1 > lab1.7.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">
6  |   <title>Document</title>
7  </head>
8  <body>
9  |   <ul id="fruit-list">
10 </body>
11 <script>
12 |   let fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
13 |   const renderFruits = () => {
14 |       const fruitList = document.getElementById('fruit-list');
15 |       fruitList.innerHTML = '';
16 |       fruits.forEach(fruit => {
17 |           const li = document.createElement('li');
18 |           li.textContent = fruit;
19 |           fruitList.appendChild(li);
20 |
21 |       });
22 |   };
23 | 
```

Document

X +

← C ① 127.0.0.1:5500/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:
 - o Get the text from the input.
 - o `.push()` the new text into the `fruits` array.
 - o Call `renderFruits()`.

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 X

▶ ⏺ ...

LAB_w1 > lab1.8.html > html > script

```
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      <ul id="fruit-list"></ul>
10     <input type="text" id="new-fruit">
11     <button id="add-btn">Add Fruit</button>
12     <ul id="fruit-list"></ul>
13 </body>
14 <script>
15     let fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
16     const fruitList = document.getElementById('fruit-list');
17     const newFruitInput = document.getElementById('new-fruit');
18     const addBtn = document.getElementById('add-btn');
```

Windsurf: Refactor | Explain | Generate Function Comment | X

20 ▾ Document X +

21 ← ⌂ ① 127.0.0.1:5500/LAB_w1/lab1.8.html

- Apple
- Banana
- Orange
- Mango

30

31
32
33
34
35
36

The screenshot shows a browser window with the title "Document". The URL in the address bar is "127.0.0.1:5500/LAB_w1/lab1.8.html". The page content includes a list of fruits:

```
20
21
22
23
24
25     • Apple
26     • Banana
27     • Orange
28     • Mango
29     • Cherry
30
31       Add Fruit
32
33
34
35
36
```

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 X

```
LAB_w1 > lab1.9.html > html
 2   <html lang="en">
11   <script>
12     let tasks= [
13       {
14         title: "Do Homework",
15         isUrgent: true
16       },
17       {
18         title: "Wash Dishes",
19         isUrgent: false
20       }
21     ];
22
23     const tasksContainer = document.getElementById('tasks-container');
24     tasks.forEach(task => {
25       const taskDiv = document.createElement('div');
26       taskDiv.textContent = task.title;
27       if (task.isUrgent) {
28         taskDiv.style.backgroundColor = 'red';
29       }
30       tasksContainer.appendChild(taskDiv);
31     });
32   </script>
33 </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:
 - o Filter the items array so it only contains items that include the filterText.
 - o Render the filtered list to the .
4. Add an event listener to the input (event type: input or keyup).
5. On event: Call render(input.value).

The screenshot shows a code editor and a browser window. The code editor displays the file `lab1.10.html` with the following content:

```
lab1.10.html X
LAB_w1 > lab1.10.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4  |  <meta charset="UTF-8">
5  |  <title>lab1.10</title>
6  </head>
7  <body>
8
9  <h2>Filter Items</h2>
10 <input type="text" id="search" placeholder="Type to filter..." />
11 <ul id="list"></ul>
12
13 <script>
14 |  let items = ["Book", "Pen", "Pencil", "Paper", "Backpack"];
15 |
16 |  const ul = document.getElementById("list");
17 |  const input = document.getElementById("search");
Windsurf: Refactor | Explain | Generate Function | Comment | X
18 |  function render(filterText) {
19 |    ul.innerHTML = "";
20 |    const filteredItems = items.filter(item =>
21 |      item.includes(filterText))
```

The browser window below shows the rendered HTML. It has a title bar with `lab1.10`, a URL bar with `127.0.0.1:5500/LAB_w1/lab1.10.html`, and the content area with the heading **Filter Items** and a search input field labeled `Type to filter...`. Below the input is a list of items:

- Book
- Pen
- Pencil
- Paper
- Backpack

The screenshot shows a browser window with the title "lab1.10.html". The page content is as follows:

```
LAB_w1 > lab1.10.html > html > body > script
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>lab1.10</title>
6  </head>
7  <body>
8
9      <h2>Filter Items</h2>
10     <input type="text" id="search" placeholder="Type to filter..." />
11     <ul id="list"></ul>
12
13     <script>
14         let items = ["Book", "Pen", "Pencil", "Paper", "Backpack"];
15
16         const ul = document.getElementById("list");
17         const input = document.getElementById("search");
18         function render(filterText) {
19             ul.innerHTML = "";
20             const filteredItems = items.filter(item =>
21                 item.includes(filterText)

```

The browser's address bar shows "127.0.0.1:5500/LAB_w1/lab1.10.html". The page itself has a heading "Filter Items" and a search input field containing "Book". Below the input, a list contains a single item: "• Book".

===== 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

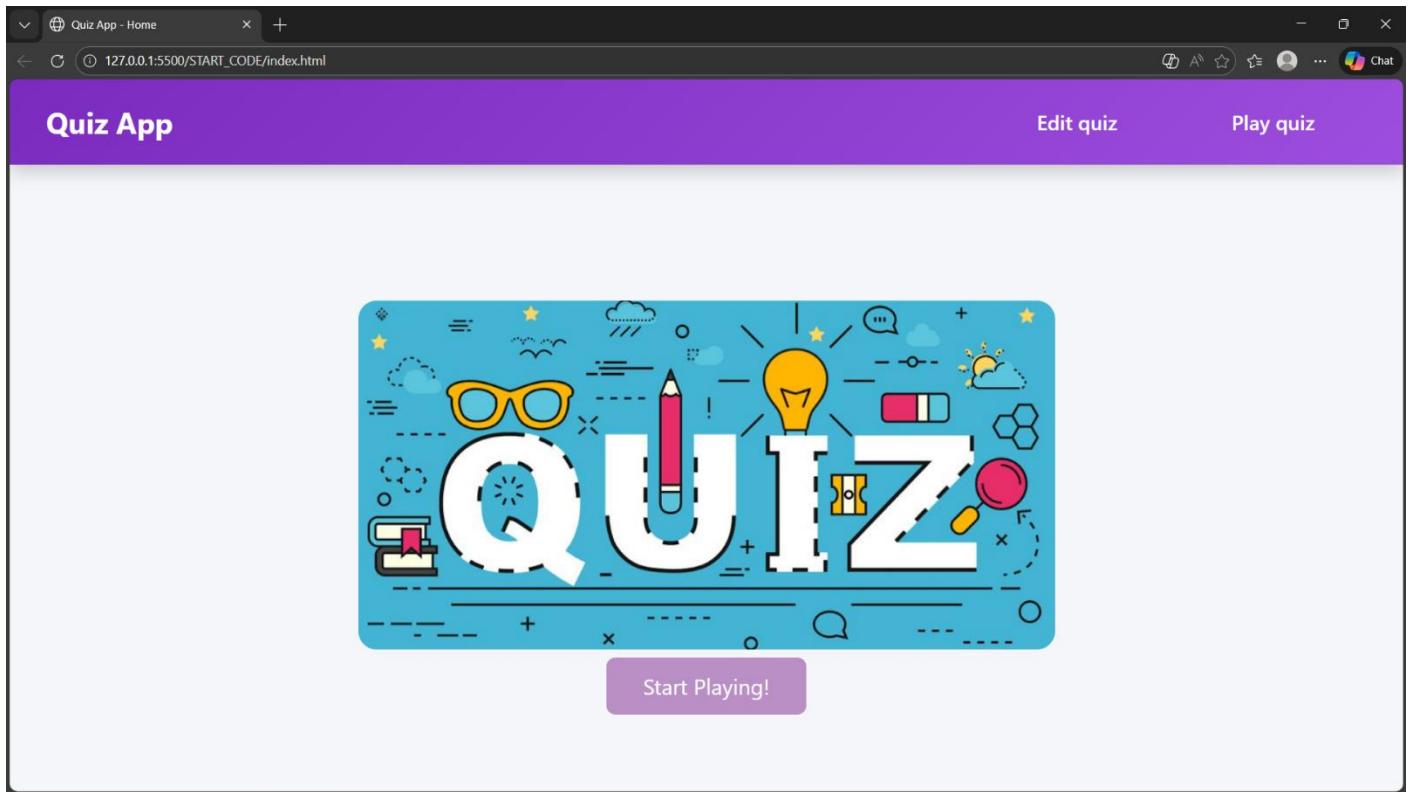
- View all quiz questions
- Edit existing questions
- Add new questions
- (Optional) Delete questions

Navigation

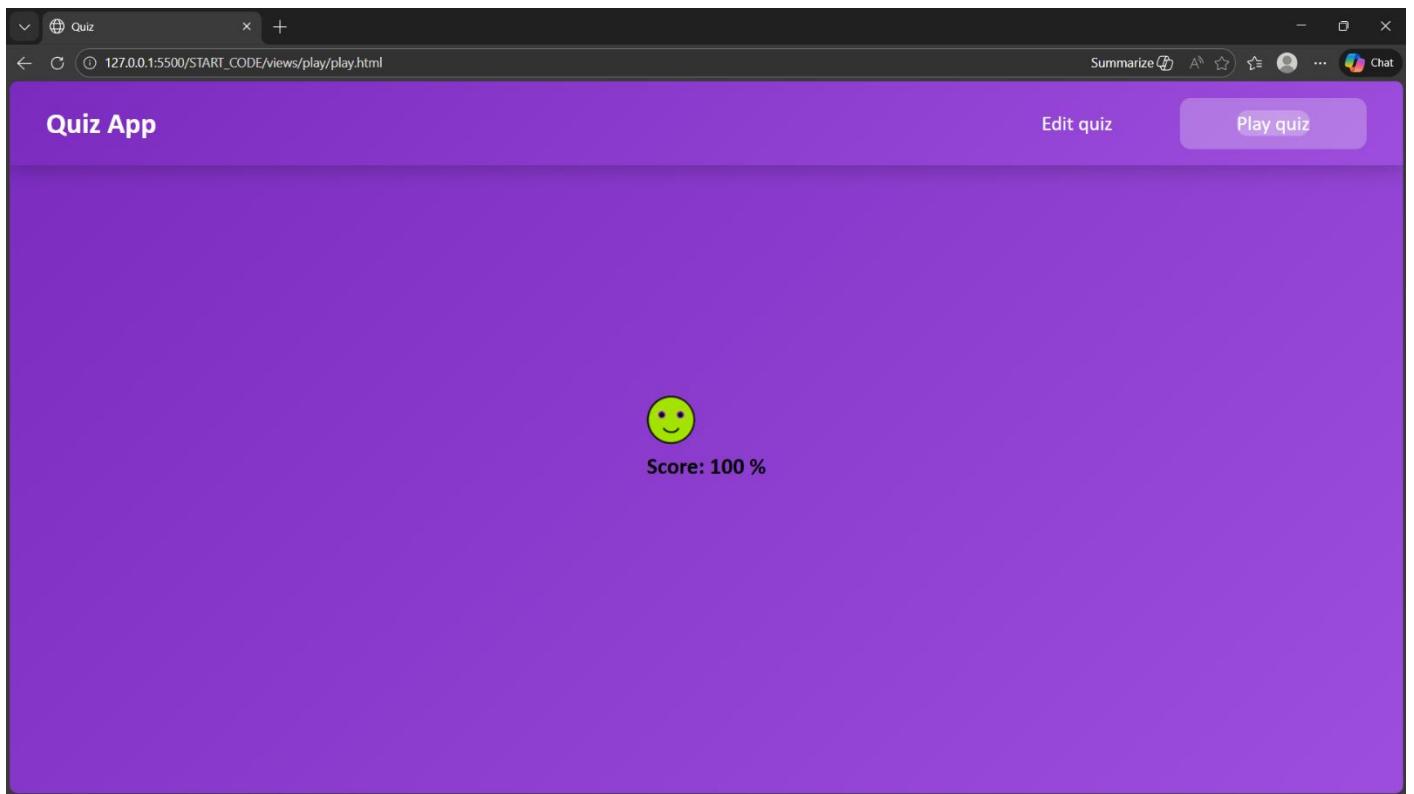
- Switch between views

The image displays four screenshots of a mobile application titled "Quiz App".

- Splash Screen:** Shows a cartoon brain character wearing glasses and holding books, with the text "Quiz App" at the top.
- Quiz View:** A question "What does CSS stand for?" with four options: "Cisco and Super Start", "Ci So Sa", "Cascading Style Sheets", and "I don't know!".
- Creation View:** A modal for creating a new question. It has a title field containing "What does HTML stand for?", an answers section with four entries: "Hi Thierry More Laugh", "How To move Left", "Ho Theary Missed the Laundry!", and "Hypertext Markup Language", and two buttons at the bottom: "Cancel" and "EDIT".
- List View:** A list of three questions: "What does HTML stand for?", "What does CSS stand for?", and "What does JS stand for?". Each question has a checked checkbox icon and a red circular icon next to it.



A screenshot of a web browser showing the Quiz App play page. The title bar says "Quiz". The main content displays a quiz question: "What does HTML stand for?". Four options are shown in boxes: "Hi Thierry More Laught", "How To move Left", "Ho Theory Missed the Laundry!", and "Hypertext Markup Language". A green progress bar is at the bottom of the screen.



A screenshot of a web browser window titled "Quiz - Edit Mode". The URL in the address bar is "127.0.0.1:5500/START_CODE/views/edit/edit.html". The page content includes a purple header with the text "Quiz App" on the left, "Edit quiz" in the middle, and "Play quiz" on the right. On the left side, there is a button labeled "Add Question". Below the header, there are three questions listed in a list-like format, each with a blue checkmark icon and a red trash can icon to its right.

- 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

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

The screenshot shows a web-based quiz application interface. At the top, there's a browser header with the URL '127.0.0.1:5500/START_CODE/views/edit/edit.html'. Below the header, the main area has a dark purple header bar with the text 'Quiz App'. On the left, there are three cards with questions: 'What does HTML stand for?', 'What does CSS stand for?', and 'What does JS stand for?'. A large central modal window is open, titled 'Edit question'. It contains a 'Title' field with the text 'What does HTML stand for?'. Under the 'Answers' section, four options are listed: 'Hi Thierry More Laught', 'How To move Left', 'Ho Theary Missed the Laundry !', and 'Hypertext Markup Language'. In the 'Correct Answer' section, a dropdown menu is open, showing the letter 'D' as the selected option. At the bottom of the modal are two buttons: 'Cancel' and 'Update'.