This code is a browser-based multiple-choice quiz app that:

Loads quiz data from /api/quiz dynamically.

Displays one question at a time.

Lets the user select an answer, submit it, view feedback, and click Next.

Shows a final score with a restart button.

Set variables for quizData, currentQuestionIndex, correctCount

quizData: Holds the entire array of quiz questions fetched from the server.

currentQuestionIndex: Tracks which question to show.

correctCount: Keeps a running total of correct answers submitted by the user.

Create a Type Guard: isDiv(el)

This function checks if a DOM element is actually a . Why?

document.getElementById('quiz') might return null or a wrong type.

This prevents errors if something like container.innerHTML = '' runs on null.

Fetch Data from the Server

This code:

Makes a GET request to /api/quiz.

Converts the response into JavaScript using .json().

Stores it in quizData.

Calls showQuestion(0) to display the first question.

showQuestion(index)

This function displays a single question, with radio buttons and buttons for "Submit Answer" and “Next".

Step-by-step Breakdown:
1. Get and validate the quiz container-

Safely gets the container where the quiz will appear.

Clears the container before showing the next question.

2. Create and display the question-

q is the current question.

An <h5> heading shows the question text (e.g., "1. What is 2 + 2?").

3. **Render choices as radio buttons**

```
q.choices.forEach(choice => {
    const input = document.createElement('input');
    input.type = 'radio';
    input.name = `question-${index}`; // ensures only one selectable
    input.value = choice;
    input.classList.add('form-check-input');
    input.id = `q${index}-${choice}`;

    const label = document.createElement('label');
    label.classList.add('form-check-label');
    label.setAttribute('for', input.id);
    label.textContent = choice;

    const wrapper = document.createElement('div');
    wrapper.classList.add('form-check');
    wrapper.appendChild(input);
    wrapper.appendChild(label);

    questionDiv.appendChild(wrapper);
});
```

Each answer is a radio input inside a Bootstrap .form-check wrapper.

input.name = "question-0" groups them together.

input.id and label.for link the label and input.

Helps screen readers and boosts accessibility.

4. **Add hidden explanation text**

```
const explanation = document.createElement('p');
explanation.id = `explanation-${index}`;
explanation.classList.add('text-info');
explanation.style.display = 'none';
questionDiv.appendChild(explanation);
```

Placeholder for showing feedback (correct answer + explanation) after submission.

text-info gives it blue Bootstrap text style.

5. **Add Submit Button**

On click:

Finds the selected radio input.

If no answer is selected, it does nothing.

Otherwise, it checks if the answer is correct.

Hint:

```
const userAnswer = selected.value;
const isCorrect = userAnswer === q.answer;
```

If correct:

Increments correctCount.

Highlights the wrapper with green (#d4edda).

If wrong:

Highlights with red (#f8d7da).

Explanation is shown with a link to learn more.

Submit button is disabled after submission.

Next button is now clickable.

6. **Create a Next Button -**

   Increments the index.

   Loads the next question or shows the final score.

7. **showResults()**

   Clears the quiz.

   Shows the score.

   Adds a Restart Quiz button that reloads the page to start over.

8. Convert these to TypeScript
9. Add JSDOC/TSDOC info
10. Write a unit test with Jest for the js/ts
11. Write and E2E test for the form