Let's break down your JavaScript code line-by-line to understand how the quiz works. I'll explain it in simple terms focusing on data flow and logic:

1. \*\*Quiz Data Storage\*\* (Lines 11-95)

const quizData = [/\* questions array \*/]

```

- This is where all your questions, options and correct answers are stored

- Each question is an object in this array

2. \*\*State Variables\*\* (Lines 100-110)

let currentQuestionIndex = 0; // Tracks which question we're on

let score = 0; // Total correct answers

let timer; // Reference to the timer

let timeLeft = 300; // 5 minutes in seconds

const userAnswers = new Array(quizData.length).fill(null); // User's answers

```

- These variables track the quiz's current state

3. \*\*DOM Elements\*\* (Lines 115-126)

// These connect to HTML elements

const startScreen = document.getElementById("start-screen");

const quizScreen = document.getElementById("quiz-screen");

// ... and other elements ...

```

- These variables are like handles to control different parts of the webpage

4. \*\*Event Listeners\*\* (Lines 131-135)

```javascript

startButton.addEventListener("click", startQuiz);

// ... others ...

```

- These set up what happens when buttons are clicked

5. \*\*Starting the Quiz\*\* (startQuiz function)

function startQuiz() {

startScreen.style.display = "none";

quizScreen.style.display = "block";

displayQuestion();

startTimer();

}

```

- Hides start screen

- Shows quiz screen

- Shows first question

- Starts countdown timer

6. \*\*Displaying Questions\*\* (displayQuestion function)

function displayQuestion() {

// Gets current question data

const questionData = quizData[currentQuestionIndex];

// Creates HTML for question and options

questionContainer.innerHTML = `

<h3>${currentQuestionIndex + 1}. ${questionData.question}</h3>

${questionData.options.map((option, index) => `

<div class="option ${userAnswers[currentQuestionIndex] === option ? 'selected' : ''}"

onclick="selectAnswer('${option}')">${option}</div>

`).join("")}

`;

// Controls button visibility

prevButton.style.display = currentQuestionIndex > 0 ? "inline-block" : "none";

nextButton.style.display = currentQuestionIndex < quizData.length - 1 ? "inline-block" : "none";

submitButton.style.display = currentQuestionIndex === quizData.length - 1 ? "inline-block" : "none";

}

```

- Builds the question display using template strings

- Highlights selected answers using CSS classes

- Shows/hides navigation buttons based on position

7. \*\*Answer Selection\*\* (selectAnswer function)

function selectAnswer(answer) {

userAnswers[currentQuestionIndex] = answer; // Store answer

displayQuestion(); // Refresh display to show selection

}

```

- When you click an option, it saves your answer

- Redraws the question to show your selection

8. \*\*Navigation Between Questions\*\*

function nextQuestion() {

if (userAnswers[currentQuestionIndex] === null) {

alert("Please select an answer before proceeding.");

return;

}

currentQuestionIndex++;

displayQuestion();

}

function prevQuestion() {

currentQuestionIndex--;

displayQuestion();

}

```

- Next button checks if answer is selected first

- Both buttons update currentQuestionIndex and redraw

9. \*\*Timer System\*\* (startTimer function)

function startTimer() {

timer = setInterval(() => {

timeLeft--;

// Update display...

if (timeLeft <= 0) {

clearInterval(timer);

submitQuiz();

}

}, 1000);

}

```

- Updates every second

- Formats time as MM:SS

- Auto-submits when time runs out

10. \*\*Submitting Results\*\* (submitQuiz and calculateScore)

function submitQuiz() {

// Check all answers

clearInterval(timer);

calculateScore();

// Show results screen

}

function calculateScore() {

// Compare user answers with correct answers

score = userAnswers.filter((answer, index) =>

answer === quizData[index].correctAnswer).length;

// Update score display

}

```

- Validates all questions are answered

- Calculates final score by comparing answers

- Shows result screen

\*\*Key Data Flow:\*\*

1. User clicks Start → initialize quiz state

2. Timer starts counting down

3. Questions are loaded from quizData

4. User selections stored in userAnswers array

5. Navigation between questions updates currentQuestionIndex

6. On submit:

- Timer stops

- Answers compared to quizData's correct answers

- Final score calculated and displayed

\*\*Visual Flow:\*\*

Start Screen → Quiz Screen (with questions) → Result Screen

\*\*Important Logic:\*\*

- The userAnswers array tracks selections for each question

- currentQuestionIndex determines which question to show

- Timer forces automatic submission when time expires

- Navigation buttons are shown/hidden based on position

- Answer validation prevents skipping questions

This structure keeps everything organized and makes it easy to add more questions or modify the behavior. Each part has a clear responsibility, making the code easier to maintain and understand.