Rakuten Frontend Internship Questions: Explanation of Answers

Bheddar Zakaria

March 22, 2025

Question 1: Understanding Asynchronous Code Execution

Code Provided:

```
const main = function () {
  console.log("A");
  setTimeout(function () {
    console.log("B");
  }, 1000);
  console.log("C");
};
main();
```

Possible Answers:

1. A B C
2. B A C
3. A C
4. A C B

Correct Answer:

4. A C B

Explanation:

The code execution follows these steps:

1. The function `main()` is called.
2. `console.log("A")` is executed immediately, printing **A**.
3. `setTimeout` is encountered, which schedules the callback function to execute after 1000 milliseconds (1 second). However, this does not block the execution of the next line.
4. `console.log("C")` is executed immediately, printing **C**.
5. After approximately 1 second, the callback inside `setTimeout` is executed, printing **B**.

Therefore, the output is **A C B**.

Question 2: Understanding React's `useEffect` Hook

Code Provided:

```
import React, { useState, useEffect } from "react";

export default function App() {
  const [count, setCount] = useState(10);

  useEffect(
    function () {
      setCount(count * 0.5);
    },
    []
  );
  return <div className="hello-">Your counter is: {count}</div>;
}
```

Possible Answers:

1. Your counter is: 0
2. Your counter is: 10
3. Your counter is: 5
4. Your counter is: 20

Correct Answer:

3. Your counter is: 5

Explanation:

The `useEffect` hook in React runs after the component renders. In this case:

1. The initial state of `count` is set to 10 using `useState(10)`.
2. The `useEffect` hook is used with an empty dependency array (`[]`), which means it will only run once after the initial render.
3. Inside the `useEffect`, the state is updated using `setCount(count * 0.5)`. Since `count` is initially 10, this operation sets `count` to 5.
4. The component re-renders with the updated `count` value, displaying **Your counter is: 5**.

Therefore, the output is **Your counter is: 5**.