


# REACT IQ

Interviewquestion-149

---



Follow on  **@Duvvuru Kishore**



**InterviewQuestion(IQ):** What issue might arise when adding items to the list, and how can you resolve it to ensure that the list updates correctly each time the button is clicked?

```
import React, { useState } from 'react';

function ItemList() {
  const [items, setItems] = useState([]);

  const addItem = () => {
    items.push(`Item ${items.length + 1}`);
    setItems(items);
  };

  return (
    <div>
      <button onClick={addItem}>Add Item</button>
      <ul>
        {items.map((item, index) => (
          <li key={index}>{item}</li>
        ))}
      </ul>
    </div>
  );
}

export default ItemList;
```

## Expected Answer:

The main issue here is that the component **will not re-render** when an item is added.

When you **click on addItem** button normally it should add **item1** on first click then **item2** on second click and so on but here the **items are not added**

A rectangular button with rounded corners, a light gray background, and a thin dark gray border. The text "Add Item" is centered on the button in a black, sans-serif font.

Add Item

In React, when you're working with state, it's important to understand that **state updates depend on immutability**.

This means that **instead of directly modifying the existing state—like changing an array(**items.push()**) or object directly—you should always create a new version of that state with your changes applied.**

By doing this, **you provide React with a new reference to the state.** React uses these references to detect changes and decide when to **re-render components.** If you simply mutate the original state, React won't detect a change because the reference stays the same, which can lead to UI updates not happening as expected.

## Correct Code Implementation:

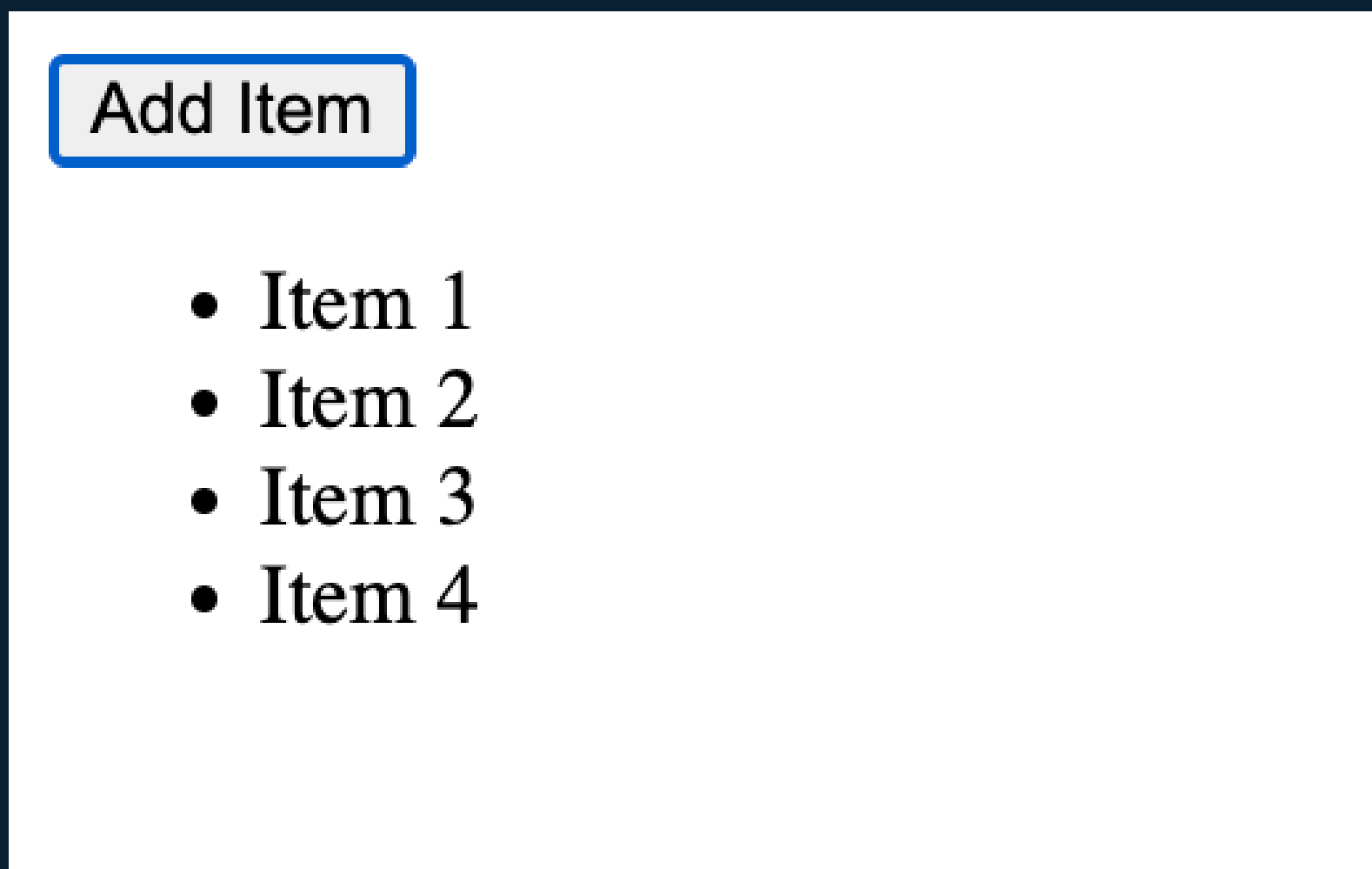
```
const addItem = () => {  
  setItems([...items, `Item ${items.length + 1}`]);  
};
```

## Explanation of the Code:

- The function `addItem` is responsible for adding a new item to the list of items in the state.
- Instead of **directly modifying the existing items array** (which would be a mutable operation), it **creates a new array**.

- The new array is created using the spread operator `[...items]`, which copies all existing items into a new array.
- A new item, labeled as Item `${items.length + 1}`, is added to this array.
- This operation returns a completely new array, providing a new reference for React to recognize.
- `setItems` is called with this new array, updating the state with a new reference.

- **React detects this new reference, recognizes that the state has changed, and triggers a re-render of the component.**
- **As a result, the UI updates correctly, showing the newly added item each time the button is clicked.**





**By using this approach, you ensure that the items are added correctly, and the component re-renders as expected, reflecting the updated state in the UI. This highlights the importance of immutability in managing state changes in React effectively.**

**Exciting News!** I'm launching a comprehensive PDF of interview questions on HTML, CSS, and JavaScript, soon expanding to React, Redux, TypeScript, and Next.js. Designed to provide clear insights and understanding, this resource compiles everything you need in one place. Stay tuned on LinkedIn for updates and get ready to elevate your preparation!



Follow on   
**@Duvvuru Kishore**

