Steeleye

Assignment

frontend

Aishwarya Kachru Naglot
4-23-2023

Question 1)

1. Explain what the simple `List` component does.

Answer:

A Simple List component is a common user interface element that displays a collection of items in a list format. We've probably seen it before in apps like your email or messaging app, where it displays a list of your emails or messages.

It's called a "simple" list component because it's a basic component that's easy to create and doesn't require any complicated code. It's also a great way to organize information and present it in a clean and concise manner.

In React, creating a Simple List component involves using a map function to loop through an array of data and render each item in the list. Each item is typically displayed as a separate row or column in the list, with each row or column containing some information about the item, such as its name or description.

The Simple List component is a popular way to display various types of information, such as contacts, products, tasks, or any other collection of related items. It can also be customized with additional features like sorting, filtering, and pagination to help users navigate through large collections of items.

Overall, the Simple List component is a straightforward and useful way to present information to users in an organized and easy-to-understand manner.

Question 2)

1. What problems / warnings are there with code?

Errors

1)

Error: setSelectedIndex is not a function in WrappedListComponent.

Explanation: setSelectedIndex is defined as a function returned from useState, but it is not called correctly in the code.

Solution: Change const [setSelectedIndex, selectedIndex] = useState(); to const [selectedIndex, setSelectedIndex] = useState(); in the WrappedListComponent function.

2)

Warning: isSelected is expected to be a boolean, but it is not defined correctly in SingleListItem.

Explanation: The isSelected prop in SingleListItem is defined as a boolean, but the selectedIndex state in WrappedListComponent is not initially set to a boolean.

Solution: Change const [selectedIndex, setSelectedIndex] = useState(); to const [selectedIndex, setSelectedIndex] = useState(false); in the WrappedListComponent function.

3)

Error: items prop in WrappedListComponent is expected to be an array of objects, but it is not defined correctly.

Explanation: The items prop in WrappedListComponent is defined as an array of objects, but the PropTypes declaration is not correct.

Solution: Change items: PropTypes.array(PropTypes.shapeOf({text: PropTypes.string.isRequired})) to items: PropTypes.arrayOf(PropTypes.shape({text: PropTypes.string.isRequired})) in the PropTypes declaration of WrappedListComponent.

Warnings

1)

Warning: setSelectedIndex is missing from the dependency array of useEffect in WrappedListComponent.

Explanation: setSelectedIndex is used inside the useEffect hook, but it is not included in the dependency array. This can cause unexpected behavior.

Solution: Add setSelectedIndex to the dependency array of useEffect in WrappedListComponent.

2)

Warning: index is missing from the dependency array of onClickHandler in SingleListItem.

Explanation: index is used inside the onClickHandler function, but it is not included in the dependency array. This can cause unexpected behavior.

Solution: Add index to the dependency array of onClickHandler in SingleListItem

3)

Warning: items is missing from the dependency array of useEffect in WrappedListComponent.

Explanation: items is used inside the useEffect hook, but it is not included in the dependency array. This can cause unexpected behavior.

Solution: Add items to the dependency array of useEffect in WrappedListComponent.

4)

Warning: onClickHandler is missing from the dependency array of SingleListItem.

Explanation: onClickHandler is used inside SingleListItem, but it is not included in the dependency array. This can cause unexpected behavior.

Solution: Add onClickHandler to the dependency array of SingleListItem.

5)

Warning: index is missing from the dependency array of the map function in WrappedListComponent.

Explanation: index is used inside the map function, but it is not included in the dependency array. This can cause unexpected behavior.

Solution: Add index to the dependency array of the map function in WrappedListComponent.

Question 3)

1. Please fix, optimize, and/or modify the component as much as you think is necessary.

Solution:

```
import React, { useState, useEffect, useCallback, memo } from 'react';
import PropTypes from 'prop-types';

// Single List Item
// memoizing the component to prevent unnecessary re-renders
const SingleListItem = memo(({ index, isSelected, onClickHandler, text }) => {
  // using a callback function to handle the click event to avoid creating a new
function on every render
  const handleClick = useCallback(() => {
    onClickHandler(index);
  }, [onClickHandler, index]);

  return (
    <li
```

```
      // dynamically setting background color based on whether or not the item is
selected
      style={{ backgroundColor: isSelected ? 'green' : 'red' }}
      onClick={handleClick} // using the callback function for click event
    >
      {text}
    </li>
  );
});

SingleListItem.propTypes = {
  index: PropTypes.number,
  isSelected: PropTypes.bool,
  onClickHandler: PropTypes.func.isRequired,
  text: PropTypes.string.isRequired,
};

// List Component
// memoizing the component to prevent unnecessary re-renders
const List = memo(({ items }) => {
  // using the state hook to store the index of the selected item
  const [selectedIndex, setSelectedIndex] = useState(null);

  // using the useEffect hook to reset the selected index when the items prop
changes
  useEffect(() => {
    setSelectedIndex(null);
  }, [items]);

  // using a callback function to handle the click event for each item to avoid
creating a new function on every render
  const handleClick = useCallback((index) => {
    setSelectedIndex(index);
  }, []);

  return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          key={index} // adding a key prop to prevent console warnings and
improve performance
          onClickHandler={handleClick} // passing the callback function to the
child component
          text={item.text}
          index={index}
```

```
          isSelected={selectedIndex === index} // checking if the current index
is the same as the selected index
        />
      ))}
    </ul>
  );
});

List.propTypes = {
  items: PropTypes.arrayOf(
    PropTypes.shape({
      text: PropTypes.string.isRequired,
    })
  ),
};

List.defaultProps = {
  items: null,
};

export default List;
```