STEELEYE LIMITED

Assessment

Name-Sunaina

Registration Number-12007753

LPU

Q1. Explain what the simple List component does in below code

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

// Single List Item
const WrappedSingleListItem = ({
  index,
  isSelected,
  onClickHandler,
  text,
}) => {
  return (
    <li
      style={{ backgroundColor: isSelected ? 'green' : 'red'}}
      onClick={onClickHandler(index)}
    >
      {text}
    </li>
  );
};

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

const SingleListItem = memo(WrappedSingleListItem);

// List Component
const WrappedListComponent = ({
  items,
}) => {
  const [setSelectedIndex, selectedIndex] = useState();
```

```
  useEffect(() => {
    setSelectedIndex(null);
  }, [items]);

  const handleClick = index => {
    setSelectedIndex(index);
  };

  return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex}
        />
      ))}
    </ul>
  )
};

WrappedListComponent.propTypes = {
  items: PropTypes.array(PropTypes.shapeOf({
    text: PropTypes.string.isRequired,
  })),
};

WrappedListComponent.defaultProps = {
  items: null,
};

const List = memo(WrappedListComponent);

export default List;
```

Solution 1

The code defines a React component called "List" that renders a list of items with the ability to select one item at a time. Breakdown of what each component does is given below:

SingleListItem component:

- This is a functional component that renders a single list item.
- It receives four props: "index", "isSelected", "onClickHandler", and "text".
- It renders an <li> element with a background color that changes depending on whether it is selected or not.
- When the <li> element is clicked, it calls the "onClickHandler" function with the "index" prop as an argument.

List component:

- This is a functional component that renders a list of items.
- It receives a single prop: "items", which is an array of objects that have a "text" property.
- It uses the useState and useEffect hooks to manage the selected index state.
- When the "items" prop changes, it resets the selected index to null using the useEffect hook.
- It defines a handleClick function that updates the selected index state when an item is clicked.
- It maps over the "items" array and renders a SingleListItem component for each item.
- Each SingleListItem component is passed the necessary props, including the onClickHandler function and the isSelected state.

Memoization:

- The SingleListItem and List components are wrapped in the memo function to optimize performance by preventing unnecessary re-renders.
- In summary, the List component renders a list of items with the ability to select one item at a time. It uses the SingleListItem component to render each item and manages the selected index state using the useState and useEffect hooks. The memo function is used to optimize performance by preventing unnecessary re-renders of the components.

Now The Detailed Explanation ->

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

- The first line imports the necessary React components and hooks for creating the List component.
- The second line imports the PropTypes library for type checking the component's props.

```
const WrappedSingleListItem = ({
  index,
  isSelected,
  onClickHandler,
  text,
}) => {
```

- This line defines a functional component called "WrappedSingleListItem" that takes in four props: "index", "isSelected", "onClickHandler", and "text".

```
  return (
    <li
      style={{ backgroundColor: isSelected ? 'green' : 'red'}}
      onClick={onClickHandler(index)}
    >
      {text}
    </li>
  );
};
```

- This block of code defines what the WrappedSingleListItem component will render.

- It returns an <li> element with a background color that changes based on whether it is selected or not.
- When the <li> element is clicked, it calls the "onClickHandler" function with the "index" prop as an argument.

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

- This block of code defines the PropTypes for the WrappedSingleListItem component to ensure that the correct props are passed to the component.

```
const SingleListItem = memo(WrappedSingleListItem);
```

- This line creates a new component called "SingleListItem" that is a memoized version of the "WrappedSingleListItem" component.
- Memoization optimizes performance by preventing unnecessary re-renders of the component.

```
const WrappedListComponent = ({
  items,
}) => {
```

- This line defines a functional component called "WrappedListComponent" that takes in one prop called "items".

```
const [setSelectedIndex, selectedIndex] = useState();
```

- This line uses the useState hook to define a "selectedIndex" state variable and a "setSelectedIndex" function for updating the state.
- The initial value of the "selectedIndex" variable is undefined.

```
useEffect(() => {
```

```
    setSelectedIndex(null);
  }, [items]);

  const handleClick = index => {
    setSelectedIndex(index);
  };
```

- This block of code defines an effect that resets the "selectedIndex" state to null whenever the "items" prop changes.
- This ensures that the selected index is reset whenever the list items change.

```
return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex}
        />
      ))}
    </ul>
  )
};
```

- This block of code defines what the WrappedListComponent will render.
- It returns an <ul> element with a textAlign property set to 'left'.
- It maps over the "items" array and renders a SingleListItem component for each item.
- Each SingleListItem component is passed the necessary props, including the onClickHandler function and the isSelected state.

```
WrappedListComponent.propTypes = {
  items: PropTypes.array(PropTypes.shapeOf({
    text: PropTypes.string.isRequired,
  })),
};

WrappedListComponent.defaultProps = {
  items: null,
};
```

```
const List = memo(WrappedListComponent);

export default List;
```

- This block of code defines the PropTypes for the WrappedListComponent to ensure that the correct props are passed to the component.It also sets a default prop value of null.

Q2. What problems / warnings are there with code?

Solution 2:

There are several problems or warnings with the code, which are listed below:

1. Unused import: The memo function is imported, but it is not used in the code. This can be removed to avoid unnecessary code clutter.

2. Missing prop type: The WrappedListComponent component has a state variable called selectedIndex, which is not assigned an initial value. This can lead to unexpected behavior and should have a prop type specified.

3. Incorrect prop type: In the WrappedListComponent component, the items prop is expected to be an array of objects with a text property. However, the PropTypes.array function should be replaced with PropTypes.arrayOf and the PropTypes.shapeOf function should be replaced with PropTypes.shape.

4. Missing key prop: In the items.map method, the SingleListItem component is rendered without a key prop. This can cause performance issues and should be corrected by assigning a unique key prop to each rendered component.

5. Incorrect onClickHandler: In the WrappedSingleListItem component, the onClickHandler function is called with an argument of index. However, it should be wrapped in an anonymous arrow function or use the bind method to pass the index as the second argument.

To correct these issues, the following changes can be made:

1. Remove the unused memo import statement.

2. Change the selectedIndex state declaration in WrappedListComponent to:

```
const [selectedIndex, setSelectedIndex] = useState(null);
```

3. Change the prop type declaration in WrappedListComponent to:

```
items: PropTypes.arrayOf(
    PropTypes.shape({
      text: PropTypes.string.isRequired,
    })
  ).isRequired,
```

4. Add a key prop to the SingleListItem component:

```
  <SingleListItem
  key={index}
  onClickHandler={() => handleClick(index)}
  text={item.text}
  index={index}
  isSelected={selectedIndex}
/>


//Change the onClickHandler prop in WrappedSingleListItem to:


onClick={() => onClickHandler(index)}
```

Q3. Please fix, optimize, and/or modify the component as much as you think is necessary.

Solution 3:

There are several issues with the code that can be improved. Below are the modifications that can be made to optimize the component:

1.In the WrappedSingleListItem component, we can change the onClick function to pass a function reference instead of invoking the function. This way, the function will only execute when the li element is clicked. Here's the modified code:

```
const WrappedSingleListItem = ({
  index,
  isSelected,
  onClickHandler,
  text,
}) => {
  const handleClick = () => {
    onClickHandler(index);
  };

  return (
    <li
      style={{ backgroundColor: isSelected ? 'green' : 'red'}}
      onClick={handleClick}
    >
      {text}
    </li>
  );
};
```

2.In the WrappedListComponent, we can set the initial state of selectedIndex to -1 instead of null. This way, we can check whether an item is selected or not by checking if selectedIndex is greater than or equal to 0. Here's the modified code:

```
const WrappedListComponent = ({
  items,
}) => {
  const [selectedIndex, setSelectedIndex] = useState(-1);

  useEffect(() => {
    setSelectedIndex(-1);
  }, [items]);

  const handleClick = index => {
```

```
    setSelectedIndex(index);
  };

  return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          key={index}
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex === index}
        />
      ))}
    </ul>
  )
};
```

3. We can also add a default value for the items prop to avoid the need to check for null in the component. Here's the modified code:

```
WrappedListComponent.defaultProps = {
  items: [],
};
```

Final code component

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

// Single List Item
const WrappedSingleListItem = ({
  index,
  isSelected,
  onClickHandler,
  text,
}) => {
  const handleClick = () => {
    onClickHandler(index);
  };

  return (
```

```
    <li
      style={{ backgroundColor: isSelected ? 'green' : 'red'}}
      onClick={handleClick}
    >
      {text}
    </li>
  );
};

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

const SingleListItem = memo(WrappedSingleListItem);

// List Component
const WrappedListComponent = ({
  items,
}) => {
  const [selectedIndex, setSelectedIndex] = useState(-1);

  useEffect(() => {
    setSelectedIndex(-1);
  }, [items]);

  const handleClick = index => {
    setSelectedIndex(index);
  };

  return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          key={index}
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex === index}
        />
      ))}
    </ul>
  )
};
```

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

WrappedListComponent.defaultProps = {
  items: [],
};

const List = memo(WrappedListComponent);

export default List;
```

By making these changes, I have optimized the component by avoiding unnecessary renders and making the code more readable.