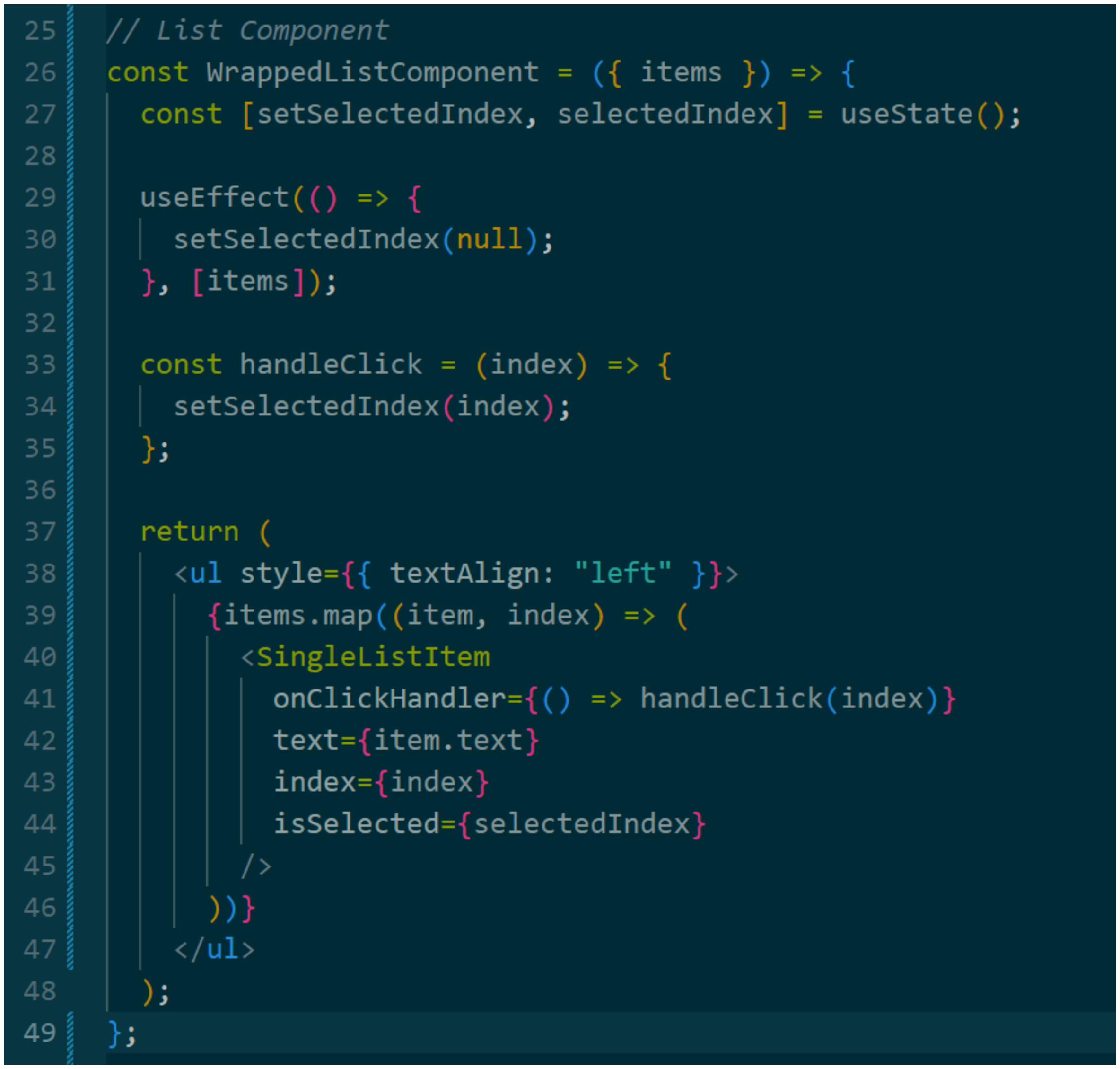
1. **Explain what the simple List component does ?**

List Component is the memoized version of `WrappedListComponent` which is getting a prop “items” .



```
// 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>
  );
};
```

Further it maps `SingleListItem` according to “items”.

SingleListItem is Also a memoized version of `WrappedSingleListItem` which returns a list element.

`WrappedSingleListItem` has an event Listener attached to it on Click. Basically it changes its background color according to the condition.

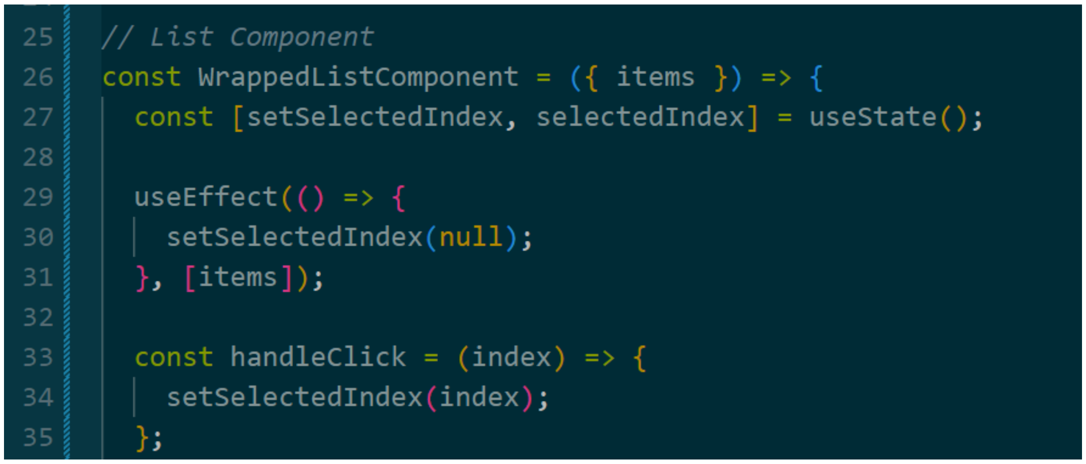
2. What problems / warnings are there with code?

The problems that I figured out were :

1. There is An Error in destructuring the result of useState hook.

   As we know, the useState() hook returns a pair of values : The first one is the current state and Second value is a function that updates it.

   This is how we write
   `const [selectedIndex,setSelectedIndex] = useState().`

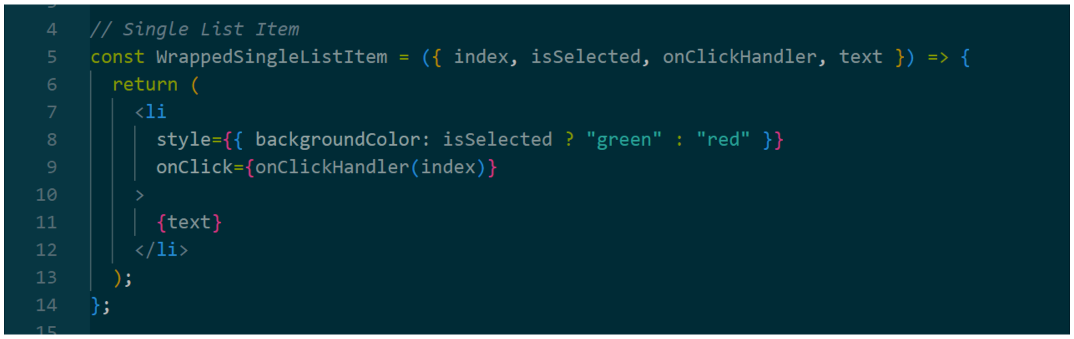
   ```
   25  // List Component
   26  const WrappedListComponent = ({ items }) => {
   27    const [setSelectedIndex, selectedIndex] = useState();
   28
   29    useEffect(() => {
   30      setSelectedIndex(null);
   31    }, [items]);
   32
   33    const handleClick = (index) => {
   34      setSelectedIndex(index);
   35    };
   ```

   But here , On line no. 27 in above image the return value of the useState function is treated oppositely, the state value is being treated as function and the second value is a function which is treated as State value. Which will cause An error :

   ```
   Uncaught TypeError: setSelectedIndex is not a function.
   ```

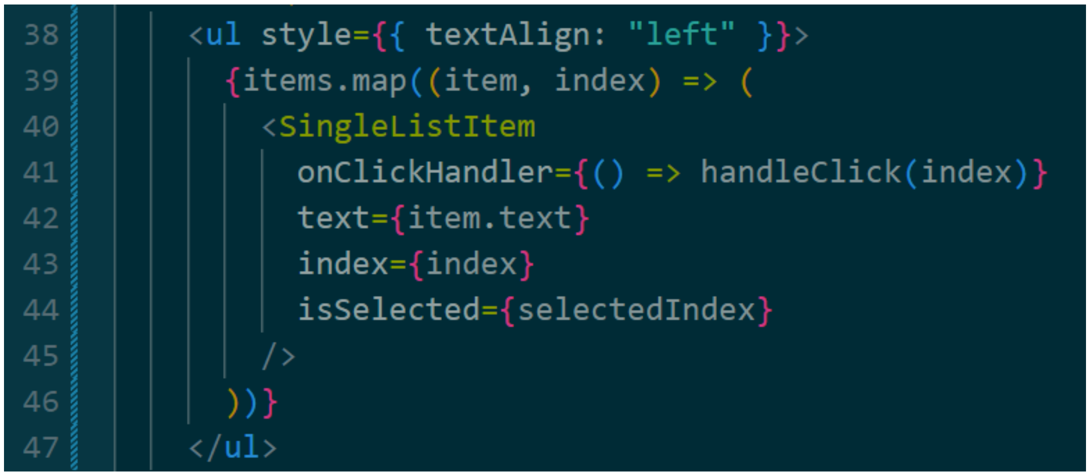
2. Invalid PropType for isSelected. It accepts bool value , but selectedIndex (number) is passed to it.

3. Incorrectly onClickHandler is passed to li element in line no. 9. This is causing an error : `Warning: Cannot update a component (`WrappedListComponent`) while rendering a different component (`WrappedSingleListItem`). To locate the bad setState() call inside `WrappedSingleListItem`.`

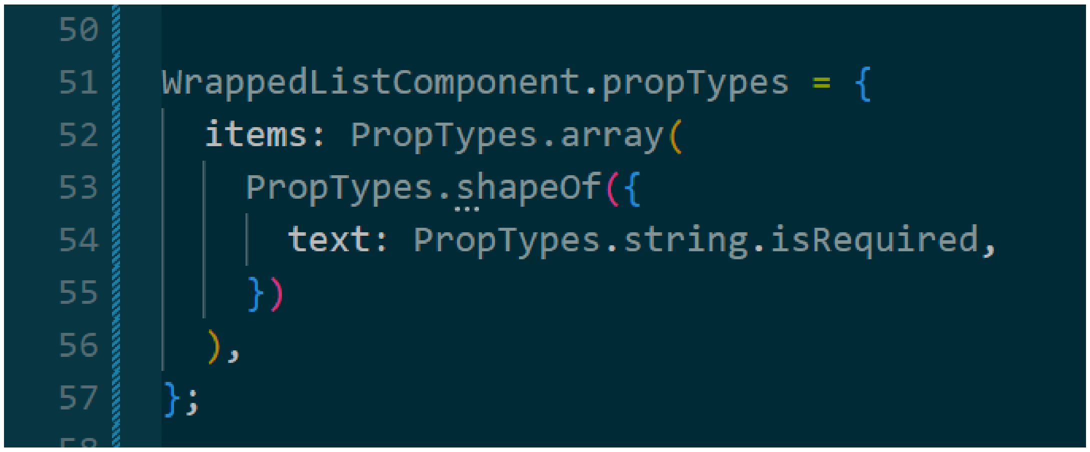

```
// Single List Item
const WrappedSingleListItem = ({ index, isSelected, onClickHandler, text }) => {
  return (
    <li
      style={{ backgroundColor: isSelected ? "green" : "red" }}
      onClick={onClickHandler(index)}
    >
      {text}
    </li>
  );
};
```

4. NULL is passed as Default prop to the **`WrappedListComponent`**

5. Key attribute is not passed while mapping SingleListItem.


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

6. TypeChecking is Done incorrectly.


```

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

In Line no. 52, it should be `PropTypes.arrayOf` instead of `PropTypes.array` and then in line 53, instead of `PropTypes.shapeOf its should be PropTypes.shape`

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

1. First and foremost I would like to make separate files for the **`WrappedListComponent`** and **`WrappedSingleListItem`** so that the code becomes more readable and organized.

2. As defaultProp value of Items is declared as a **null**, so if by chance items is not passed as prop it will assign it null and inside **`WrappedListComponent`**
When it will try to map through the value of items it will through an error : **`Uncaught TypeError: Cannot read properties of null (reading 'map')`**

Fix : Instead of setting its default value to null , we should set it to an empty array [].

```
WrappedListComponent.defaultProps = {
  Items:[], // setting default to an empty array.
};

Alternative Solution BUT with different Approach :
```

Before mapping we can check that if we have some value inside items then only we will map through it otherwise we won't. We can achieve this by conditional statement

```
Alternative Solution I :

{
      items ? ( items.map((item, index) => (
        <SingleListItem
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex}
        />
      ))) : ( <div> data not found or something else </div>
)
}
```

```
Alternative Solution II :

{
      Items && items.map((item, index) => (
        <SingleListItem
          onClickHandler={() => handleClick(index)}
          text={item.text}
          index={index}
          isSelected={selectedIndex}
        />
      ))
}
```

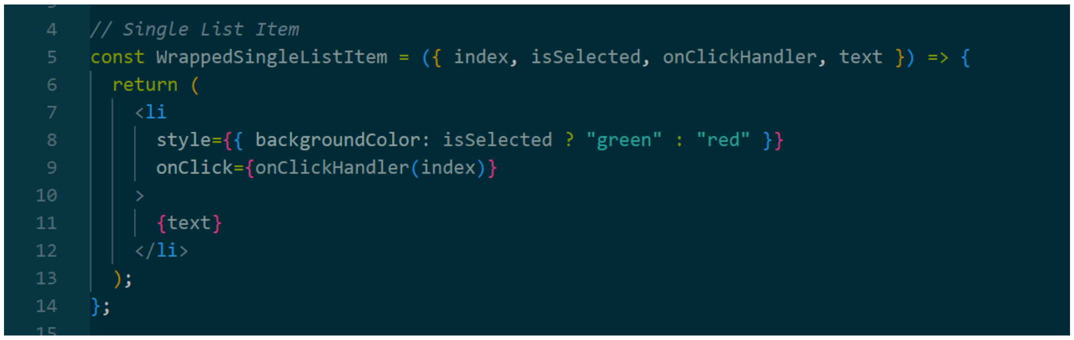
3. Fixing syntactical bugs like :

1. useState hook :

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

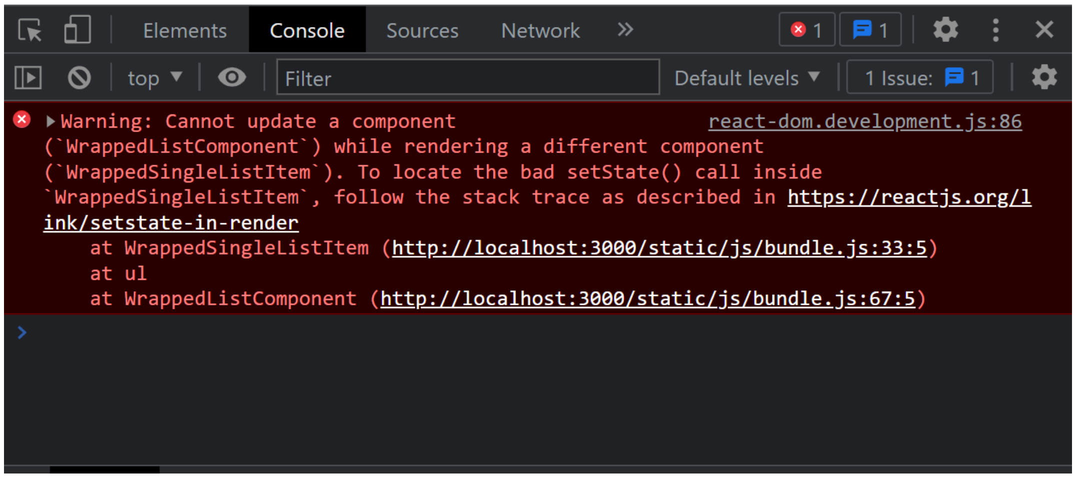
2. TypeCheck :
   Before :
   ```
         WrappedListComponent.propTypes = {
           items: PropTypes.array(
             PropTypes.shapeOf({
               text: PropTypes.string.isRequired,
             })
           ),
         };
   ```
   After :
   ```
         WrappedListComponent.propTypes = {
           items: PropTypes.arrayOf(
             PropTypes.shape({
               text: PropTypes.string.isRequired,
             })
           ),
         };
   ```

3. Passing onClickHandler properly :



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

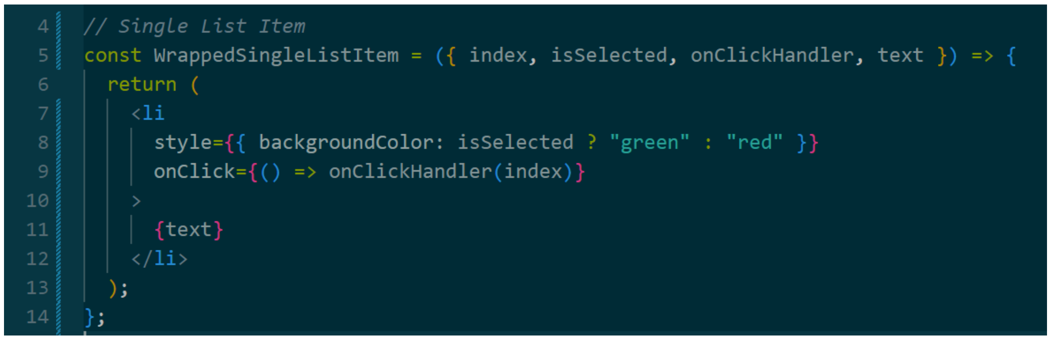
Here the expression inside onClick is going to be executed on the mount. This is going to delete all the bills in the list, as soon as the component is started.



```
▸Warning: Cannot update a component                    react-dom.development.js:86
(`WrappedListComponent`) while rendering a different component
(`WrappedSingleListItem`). To locate the bad setState() call inside
`WrappedSingleListItem`, follow the stack trace as described in https://reactjs.org/l
ink/setstate-in-render
    at WrappedSingleListItem (http://localhost:3000/static/js/bundle.js:33:5)
    at ul
    at WrappedListComponent (http://localhost:3000/static/js/bundle.js:67:5)
```

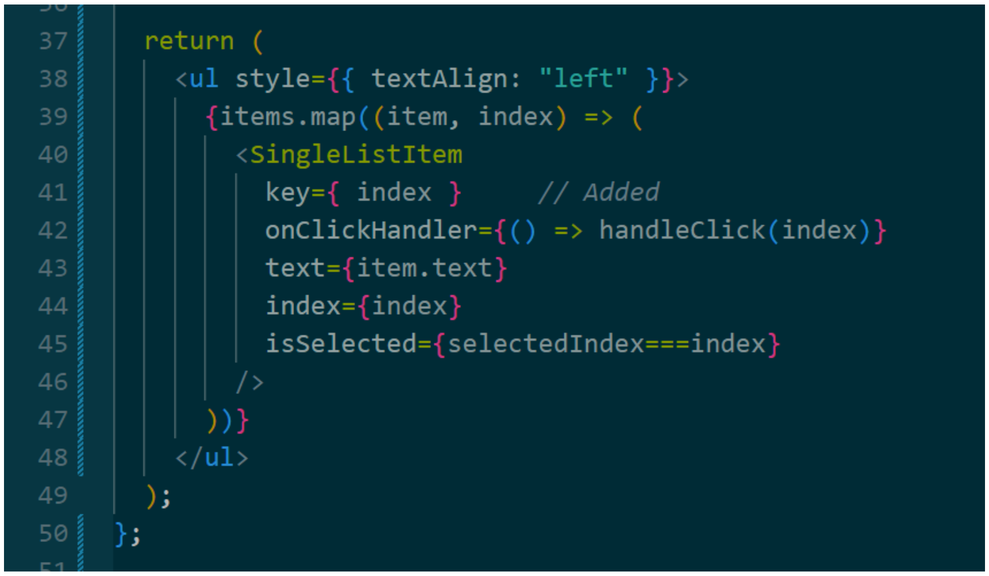
FIx :

When we use an arrow function, the event handler is automatically bound to the component instance similarly as we do in the class component to bind it in the constructor using **This** keyword.



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

4. Adding Key attribute While mapping :



```
37    return (
38      <ul style={{ textAlign: "left" }}>
39        {items.map((item, index) => (
40          <SingleListItem
41            key={ index }     // Added
42            onClickHandler={() => handleClick(index)}
43            text={item.text}
44            index={index}
45            isSelected={selectedIndex===index}
46          />
47        ))}
48      </ul>
49    );
50  };
```

5. As isSelected accepts a bool value but previously number was passed to it. We can check the current index against the selectedIndex and provide that value to isSelected As done in above picture in line no 45.

After All this changes the code looks like :

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

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

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

  return (
    <ul style={{ textAlign: "left" }}>
      {items.map((item, index) => (
        <SingleListItem
          key={ index }      // Added
          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;
```