Workshop Coding Instructions

Exercise (1)

In App.tsx, Create a Navbar component from scratch

Create the file /src/components/Navbar.tsx .

Contents of the file should look like this:

Navbar.tsx

```
import React from 'react';

export default function Navbar() {
    return (
        <nav className='navbar'>
            <h3>To Do</h3>
        </nav>
    );
}
```

Add the component to App.tsx.

App.tsx

```
import React from 'react';
import ListContainer from './components/ListContainer';
import Navbar from './components/Navbar';

export default function App() {
  return (
        <>
            <Navbar />
            <ListContainer />
        </>
  );
}
```

Exercise (2)

In ListContainer.tsx, Add a required 'title' prop to ListContainer

Initial signature:

ListContainer.tsx

```
export default function ListContainer()
```

Add interface and props parameter:

ListContainer.tsx

```
interface ListContainerProps {
    title: string
}
export default function ListContainer(props: ListContainerProps)
```

Reference props in return statement:

ListContainer.tsx

```
return (
    <div className='list-container'>
        <h1>{props.title}</h1>
    ...
)
```

Add prop value to component in App.tsx:

App.tsx

```
return (
    <>
        <Navbar />
        <ListContainer
            title="School Work"
        />
    </>
  );
```

Exercise (3)

In ListContainer.tsx, Render ListItem components from an array

Initial return:

ListContainer.tsx

```
<div className="list-container__list">
    <ListItem index={0} item={schoolItems[0]} setItems={setItems} />
    <ListItem index={1} item={schoolItems[1]} setItems={setItems} />
    <ListItem index={2} item={schoolItems[2]} setItems={setItems} />
    <ListItem index={3} item={schoolItems[3]} setItems={setItems} />
</div>
```

Add the following above the return statement:

ListContainer.tsx

```
const listItems = schoolItems.map((elem, index) =>
    <ListItem
        key={index}
        index={index}
        item={elem}
        setItems={setItems}
    />
  );
```

Change the return statement to render `listItems` instead of the raw components:

ListContainer.tsx

```
<div className="list-container__list">
    {listItems}
</div>
```

Exercise (4)

In ListItem.tsx, Create an `isCrossedOut` state

Add the following hook underneath the existing hooks

ListItem.tsx

```
const [isCrossedOut, setIsCrossedOut] = React.useState(false);
```

Create an event handler to go with it

ListItem.tsx

```
const goToggleIsCrossedOut = () => {
    setIsCrossedOut((prev) => !prev);
}
```

Reference the state ant the event handler in the return statement

ListItem.tsx

```
return (
    <div className='list-item'>
        <div
            className={isCrossedOut
                ? 'list-item__text list-item__text--done'
                : 'list-item__text'}
            onClick={goToggleIsCrossedOut}
        >
)
```

Exercise (5)

In ListItem.tsx, Set up conditional rendering to render the input box

End of file looks like this:

ListItem.tsx

```
const inputBox = <form className='list-item__form' onSubmit={goToggleEdit}>
    <input
      type="text"
      name="itemName"
      id="itemName"
      className="list-item__input"
      value={props.item}
      onChange={goChangeItem}
      autoFocus
      onFocus={(e) => e.target.select()}
      onBlur={goToggleEdit}
    />
    <button
      className="material-symbols-outlined list-item__button done"
      type='submit'
    >
      done
    </button>
  </form>

  return (
    <div className='list-item'>
      <div
        className={ isCrossedOut
          ? 'list-item__text list-item__text--done'
          : 'list-item__text'}
      >
        {props.item}
      </div>
      <div
        className="material-symbols-outlined list-item__button edit"
        onClick={goToggleEdit}
      >
        edit
      </div>
      <div
        className="material-symbols-outlined list-item__button delete"
        onClick={goDelete}
      >
        delete
      </div>
    </div>
  );
```

Return statement should look something like this when done:

ListItem.tsx

```
  if (isEditing) {
      return (
        <div className="list-item">
          <form className='list-item__form' onSubmit={goToggleEdit}>
            <input
              type="text"
              name="itemName"
              id="itemName"
              className="list-item__input"
              value={props.item}
              onChange={goChangeItem}
              autoFocus
              onFocus={(e) => e.target.select()}
              onBlur={goToggleEdit}
            />
            <button
              className="material-symbols-outlined list-item__button done"
            type='submit'
          >
            done
          </button>
        </form>
      </div>
    );
  } else {
    return (
      <div className='list-item'>
        <div
          className={ isCrossedOut
            ? 'list-item__text list-item__text--done'
            : 'list-item__text'}
        >
          {props.item}
        </div>
        <div
          className="material-symbols-outlined list-item__button edit"
          onClick={goToggleEdit}
        >
          edit
        </div>
        <div
          className="material-symbols-outlined list-item__button delete"
          onClick={goDelete}
        >
          delete
        </div>
      </div>
    );
  }
```

There may be alternatives that work.

Exercise (6)

In ListContainer.tsx, Add state for the form values

Go back and change the component array to use state:

ListContainer.tsx

```
const listItems = items.map((elem, index) =>
    <ListItem
        key={index}
        index={index}
        item={elem}
        setItems={setItems}
    />
);
```

Input element currently looks like this:

ListContainer.tsx

```
<input
    className='list-container__input'
    type="text"
    name="itemName"
    id="itemName"
/>
```

Change input element to look like this:

ListContainer.tsx

```
<input
    className='list-container__input'
    type="text"
    name="itemName"
    id="itemName"
    value={inputString}
    onChange={(e) => setInputString(e.target.value)}
/>
```