React-3

agenda

- input and controlled components
- thinking in react
- shopping cart application

Controlled Components in React

Controlled components are components in which form data is handled by the component's state. This makes React the single source of truth for the form data. Let's explore this concept using the `InputBox` component you provided. Controlled components provide **Single Source of Truth:**- The form data is stored in the component's state, ensuring consistency.

`InputBox` Component

Here's the `InputBox` component:

```
import { useState } from "react";

function InputBox() {
    const [content, setContent] = useState("");

    const handleAlert = () => {
        alert(content);
        setContent("");
    }

    const handleChange = (e) => {
        const updatedValue = e.target.value;
        setContent(updatedValue);
    }

    return (
        <div>
```

```
            <input type="text" value={content} onChange={handleChange} />
            <button onClick={handleAlert}>Alert content</button>
        </div>
    );
}

export default InputBox;
```

Explanation

1. **State Initialization:**
   - The component uses the `useState` hook to create a state variable `content` and a function `setContent` to update it.
   - `const [content, setContent] = useState("");`
   - Initially, `content` is an empty string.
2. **Handling Input Change:**
   - The `handleChange` function updates the state with the current value of the input field.
   - `const handleChange = (e) => { const updatedValue = e.target.value; setContent(updatedValue); }`
   - The `onChange` event on the `input` element triggers `handleChange`, ensuring the state `content` is updated with every keystroke.
3. **Alert and Reset:**
   - The `handleAlert` function alerts the current value of `content` and then resets it to an empty string.
   - `const handleAlert = () => { alert(content); setContent(""); }`
   - This function is triggered by clicking the `button`.
4. **Input Element:**
   - The `value` attribute of the `input` element is set to `content`.
   - `<input type="text" value={content} onChange={handleChange} />`
   - This makes the `input` a controlled component because its value is controlled by the state `content`.

Revision Notes: Shopping Cart Component

Steps to Create the Component

1. **Create Static HTML**
2. **Add Event Listener**
3. **Identify Dynamic Data** (task and list of tasks)

Step-by-Step Explanation

Step 1: Create Static HTML

Initial HTML Structure:

- Begin by writing the static HTML structure. This includes a text input, a button, and a placeholder for the list of tasks.

```
import React from 'react';

function Shopping() {
    return (
        <div className='Shopping-list'>
            <div className="input-box">
                <input type="text" />
                <button>Add Item</button>
            </div>
            <h2>Shopping Cart</h2>
            <ul className="list">
                {/* List items will go here */}
            </ul>
        </div>
    );
}

export default Shopping;
```

- This static structure helps visualize the layout and plan for dynamic data integration.

Step 2: Add Event Listener

Add Event Listeners for User Interactions:

- Implement event listeners for the input field and button to handle user actions.

Code:

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

function Shopping() {
    const [content, setContent] = useState("");

    const handleInput = (e) => {
        setContent(e.target.value);
    }

    const handleAddItem = () => {
        // Logic for adding item will go here
    }

    return (
        <div className='Shopping-list'>
            <div className="input-box">
                <input type="text" onChange={handleInput} value={content}
/>
                <button onClick={handleAddItem}>Add Item</button>
            </div>
            <h2>Shopping Cart</h2>
            <ul className="list">
                {/* List items will go here */}
            </ul>
        </div>
    );
}

export default Shopping;
```

- The `handleInput` function updates the state with the current input value.

- The `handleAddItem` function will handle adding the new item to the list.

Step 3: Identify Dynamic Data

Manage Dynamic Data with State:

- Use state to manage the input value (`content`) and the list of tasks (`tasks`).

Code:

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

function Shopping() {
    const [content, setContent] = useState("");
    const [tasks, setTasks] = useState([]);

    const handleInput = (e) => {
        setContent(e.target.value);
    }

    const handleAddItem = () => {
        const newItem = content;
        const newArr = [...tasks];
        newArr.push(newItem);
        setTasks(newArr);
        setContent("");
    }

    return (
        <div className='Shopping-list'>
            <div className="input-box">
                <input type="text" onChange={handleInput} value={content}
/>
                <button onClick={handleAddItem}>Add Item</button>
            </div>
            <h2>Shopping Cart</h2>
            <ul className="list">
                {tasks.map((item, index) => (
                    <li key={index}>
```

```
                        <span>{item}</span>
                        <button>Delete</button>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default Shopping;
```

- `content`: Stores the current value of the input field.
- `tasks`: Stores the list of tasks.

Step 4: Implement Delete Functionality

Add Functionality to Delete Items from the List:

- Implement the `handleDelete` function to remove items from the list.

Code:

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

function Shopping() {
    const [content, setContent] = useState("");
    const [tasks, setTasks] = useState([]);

    const handleInput = (e) => {
        setContent(e.target.value);
    }

    const handleAddItem = () => {
        const newItem = content;
        const newArr = [...tasks];
        newArr.push(newItem);
        setTasks(newArr);
        setContent("");
    }
```

```
    const handleDelete = (index) => {
        const filteredArr = tasks.filter((_, i) => i !== index);
        setTasks(filteredArr);
    }

    return (
        <div className='Shopping-list'>
            <div className="input-box">
                <input type="text" onChange={handleInput} value={content}
/>
                <button onClick={handleAddItem}>Add Item</button>
            </div>
            <h2>Shopping Cart</h2>
            <ul className="list">
                {tasks.map((item, index) => (
                    <li key={index}>
                        <span>{item}</span>
                        <button onClick={() =>
handleDelete(index)}>Delete</button>
                    </li>
                ))}
            </ul>
        </div>
    );
}

export default Shopping;
```

- `handleDelete`: Removes the item at the specified index from the list.

Step 5: Extract into Independent Components

Refactor to Separate Components:

- Create `InputBox` and `ListItem` components.
- Lift the state to the parent component (`Shopping`) and pass the necessary props.

`InputBox` Component:

```
function InputBox({ handleInput, handleAddItem, content }) {
    return (
        <div className="input-box">
            <input type="text" onChange={handleInput} value={content} />
            <button onClick={handleAddItem}>Add Item</button>
        </div>
    );
}

export default InputBox;
```

`ListItem` Component:

```
function ListItem({ tasks, handleDelete }) {
    return (
        <ul className="list">
            {tasks.map((item, index) => (
                <li key={index}>
                    <span>{item}</span>
                    <button onClick={() =>
handleDelete(index)}>Delete</button>
                </li>
            ))}
        </ul>
    );
}

export default ListItem;
```

Updated `Shopping` Component:

```
import React, { useState } from 'react';
import InputBox from './InputBox';
import ListItem from './ListItem';

function Shopping() {
    const [content, setContent] = useState("");
    const [tasks, setTasks] = useState([]);

    const handleInput = (e) => {
```

```
        setContent(e.target.value);
    }

    const handleAddItem = () => {
        const newItem = content;
        const newArr = [...tasks];
        newArr.push(newItem);
        setTasks(newArr);
        setContent("");
    }

    const handleDelete = (index) => {
        const filteredArr = tasks.filter((_, i) => i !== index);
        setTasks(filteredArr);
    }

    return (
        <div className='Shopping-list'>
            <InputBox
                handleInput={handleInput}
                handleAddItem={handleAddItem}
                content={content}
            />
            <h2>Shopping Cart</h2>
            <ListItem
                tasks={tasks}
                handleDelete={handleDelete}
            />
        </div>
    );
}

export default Shopping;
```