

W13-P1: PersonContext for T82_xx

The screenshot displays a web application running on localhost:3000. The browser window shows the title "Tutorial T81-210410626" and the subtitle "Context API / useContext". Below the title, there is a list of people: John and Peter, each with a "remove" button. The code editor on the left shows the implementation of the Context API. The `PersonContext` is created using `React.createContext()`. The `T81_26_ContextAPI` component uses `useState` to manage the state and `useContext` to access the context. The `List` component uses `useContext` to access the context and render the list of people. The Chrome DevTools component inspector on the right shows the component tree, highlighting the `Context.Provider` and `List` components. The hooks panel shows the `useContext` hook being used in the `List` component.

```
const PersonContext = React.createContext();

const T81_26_ContextAPI = () => {
  const [people, setPeople] = useState(data);
  const removePerson = (id) => {
    setPeople(people => {
      return people.filter((person) => person.id !== id);
    });
  };
  return (
    <PersonContext.Provider value={ {people, removePerson} }>
      <h3>context API / useContext</h3>
      <List/>
    </PersonContext.Provider>
  );
};

const List = () => {
  const mainData = useContext(PersonContext);
  console.log('mainData', mainData);
  return (
    <mainData.people.map((person) => {
      return (
        <SinglePerson
          key={person.id}
          {...person}
        />
      );
    })
  );
};
```

Context: {people: Array(2), removePerson: f removePerson() {}}

people: [{id: 1, name: "john"}, {id: 2, name: "peter"}]

W13-P2: Apply useReducer to add an item with modal alert

```
src > tutorial > 6-useReducer > JS T6_26-useReducer.js > [e] T6_26_useReducer
11 modalContent: ''
12 }
13
14 const T6_26_useReducer= () => {
15
16   const [name, setName] = useState('');
17   const [state, dispatch] = useReducer(reducer_26, defaultState);
18
19   // const [people, setPeople] = useState(data);
20   // const [isModalOpen, setIsModalOpen] = useState(false);
21   // const [modalContent, setModalContent] = useState('');
22
23
24   const handleSubmit = (e) => {
25     e.preventDefault();
26     if(name){
27       const newItem = { id: new Date().getTime().toString(), name};
28       dispatch({type: 'ADD_ITEM', payload: newItem});
29       // setPeople([...people, newItem]);
30       setName('');
31     }else{
32     }
33   }
34
35   const closeModal = () => {
36     dispatch({type: 'CLOSE_MODAL'});
37     // setIsModalOpen(false);
38   }
39
40   return <>
41     { state.isModalOpen && <T6_26_Modal modalContent={state.modalContent} closeModal={
42       <form className="form" onSubmit={handleSubmit}>
43         <div>
44           <input type="text" value={name} onChange={(e) => setName(e.target.value)} />
45         </div>
46         <button type="submit">add</button>
47       </form>
48       { state.people.map( (person) => {
49         return (
50           <div key={person.id} className="item">
51             <h4>{person.name}</h4>
52             <button>remove</button>
53           </div>
54         )
55       } ) }
56     }
57   </>;
58 }
```

```
src > tutorial > 6-useReducer > JS T6_26_reducer.js > [e] reducer_26
1 export const reducer_26 = (state, action) => {
2
3   if(action.type === 'ADD_ITEM') {
4     const newPeople = [...state.people, action.payload];
5     return {
6       ...state,
7       people: newPeople,
8       isModalOpen: true,
9       modalContent: 'item added'
10    }
11  }
12
13  if(action.type === 'CLOSE_MODAL') {
14    return {
15      ...state,
16      isModalOpen: false,
17      modalContent: ''
18    }
19  }
```

The image shows a web browser window displaying a tutorial application. The browser's address bar shows the URL `localhost:3000`. The page title is "Tutorial T6-210410626". The application interface includes a form with an "Add" button and a list of items, each with a "remove" button. The items listed are John, Peter, Susan, Weiyl, and Wei. The right sidebar shows the React DevTools component inspector, displaying the state of the application. The state is an object with `isModalOpen: true`, `isModalOpen: false`, `modalContent: "item added"`, and `isModalOpen: true`. The component inspector also shows the props and hooks for the `T6_26_useReducer` component.