useContext

useContext用于在快速在函数组件中导入上下文。

```
import React, { useContext } from "react";
import { Context } from "../AppContext";

export default function UseContextPage() {
  const ctx = useContext(Context);
  const { name } = ctx.user;
  return (
    <div>
      <h1>UseContextPage</h1>
      <p>{name}</p>
    </div>
  );
}
```

useReducer

reducer 就是一个纯函数，接收旧的 state 和 action，返回新的 state。

useReducer是useState的可选项，常用于组件有复杂状态逻辑时，类似于redux中reducer概念。

商品列表状态维护

```
  import React, { useEffect, useReducer }
from "react";
  import FruitList from
"../components/FruitList";
  import FruitAdd from
"../components/FruitAdd";

  function fruitReducer(state = [], action)
{
    switch (action.type) {
      case "replace":
      case "init":
        return [...action.payload];
      case "add":
        return [...state, action.payload];
      default:
        return state;
    }
  }
  export default function UseReducerPage()
{
    const [fruits, dispatch] =
useReducer(fruitReducer, []);
    useEffect(() => {
      setTimeout(() => {
        dispatch({ type: "init", payload:
["apple", "banana"] });
      }, 1000);
      return () => {};
    }, []);
    return (
      <div>
        <h1>UseReducerPage</h1>
        <FruitAdd
          fruits={fruits}
          addFruit={name => dispatch({
type: "add", payload: name })}
        />
        <FruitList
          fruits={fruits}
          setFruits={newList => dispatch({
type: "init", payload: newList })}
        />
      </div>
    );
  }
```