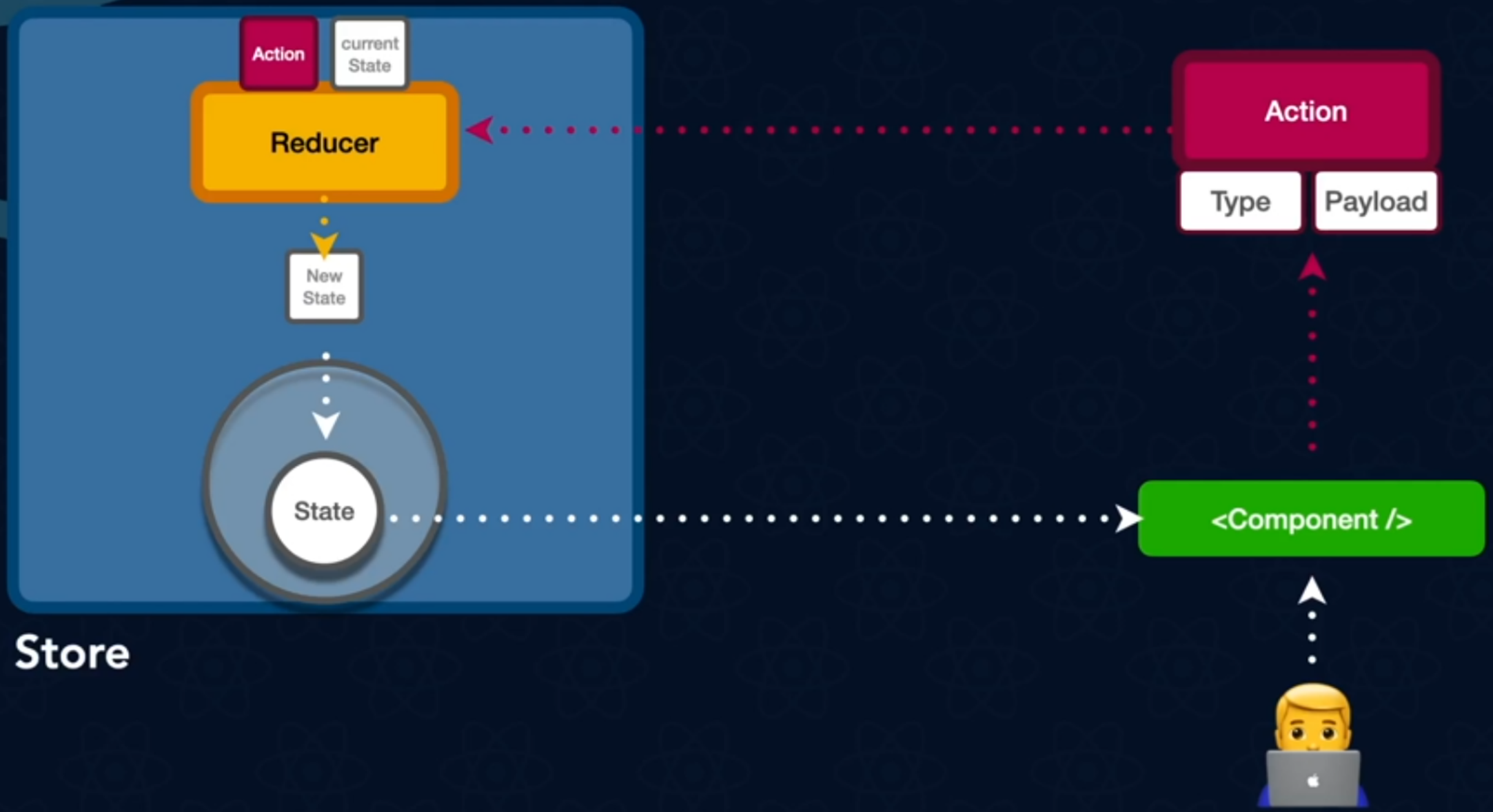
#114

مطابقة الشرح النظري و التطبيق العملي

126. مطابقة الشرح النظري و التطبيق العملي



```
import { configureStore } from '@reduxjs/toolkit'
import resultReducer from '../features/calc/calcSlice'
export const store = configureStore({
  reducer: {
  calc:resultReducer,
  },
})
```

```
import { createSlice } from "@reduxjs/toolkit";

const initialStat = {
    result: 0,
};

export const calcSlice = createSlice({
    // the slice name :
    name: "calc",
    // the stat var :
    initialState: initialStat,

    reducers: {
        sum: (currentStat, action) => {
            const { firstInputValue, secondInputValue } = action.payload;
            currentStat.result = firstInputValue + secondInputValue;
        },
        subtract: (currentStat, action) => {
            const { firstInputValue, secondInputValue } = action.payload;
            currentStat.result = firstInputValue - secondInputValue;
        },
        multiply: (currentStat, action) => {
            const { firstInputValue, secondInputValue } = action.payload;
            currentStat.result = firstInputValue * secondInputValue;
        },
        divide: (currentStat, action) => {
            const { firstInputValue, secondInputValue } = action.payload;
            currentStat.result = firstInputValue / (secondInputValue === 0 ? 1:secondInputValue );
        },
    },
});

export const { sum ,divide,multiply,subtract} = calcSlice.actions;

export default calcSlice.reducer;

```

```jsx
import "./App.css";

import { TextField, Button, Box, Stack } from "@mui/material";
import { useRef } from "react";
import { useSelector, useDispatch } from "react-redux";
import { sum, subtract, divide, multiply } from "./features/calc/calcSlice";
function App() {
  const result = useSelector((state) => state.calc.result);
  const dispatch = useDispatch();

  const firstInputRef = useRef("");
  const secondInputRef = useRef("");

  function handleAddition() {
    const firstInputValue = +firstInputRef.current.value || 0;
    const secondInputValue = +secondInputRef.current.value || 0;
    dispatch(sum({ firstInputValue, secondInputValue }));
  }
  function handleSubtraction() {
    const firstInputValue = +firstInputRef.current.value || 0;
    const secondInputValue = +secondInputRef.current.value || 0;

    dispatch(subtract({ firstInputValue, secondInputValue }));
  }
  function handleMultiplication() {
    const firstInputValue = +firstInputRef.current.value || 0;
    const secondInputValue = +secondInputRef.current.value || 0;

    dispatch(multiply({ firstInputValue, secondInputValue }));
  }
  function handleDivision() {
    const firstInputValue = +firstInputRef.current.value || 0;
    const secondInputValue = +secondInputRef.current.value || 1;

    dispatch(divide({ firstInputValue, secondInputValue }));
  }

  return (
    <div className="App">
      <h1 id="result">{result.toFixed(2)}</h1>
      <Box component="form">
        <Stack alignItems="center" spacing="20px">
          <TextField required inputRef={firstInputRef} autoFocus type="number" sx={{ width: "70%" }} placeholder="The First number" name="firstNumber" label="number 1" variant="outlined"></TextField>
          <TextField required inputRef={secondInputRef} type="number" sx={{ width: "70%" }} placeholder="The Second number" name="secondNumber" label="number 2" variant="outlined"></TextField>

          <Button variant="outlined" sx={{ width: "70%" }} color="primary" onClick={handleAddition}>
            Add
          </Button>
          <Button variant="outlined" sx={{ width: "70%" }} color="primary" onClick={handleSubtraction}>
            Subtract
          </Button>
          <Button variant="outlined" sx={{ width: "70%" }} color="primary" onClick={handleMultiplication}>
            Multiply
          </Button>
          <Button variant="outlined" sx={{ width: "70%" }} color="primary" onClick={handleDivision}>
            Divide
          </Button>
        </Stack>
      </Box>
    </div>
  );
}

export default App;

```