```
/* ====== useReducer() for State Management ==========
   sometimes two or more state change at time for similer task
   like eteredPassword and chacking password at time that time useReucer() hook can be
   work and easy to mamage task.

   Sometimes, you have more complex state = for example if it got multiple states, multiple ways of
   changing it or dependencies to ohter states

   useState() then often becomes hard or error-prone to use - its easy to write bad, inefficient or buggy code in such scenarios

   useReducer() can be used as a replacement for useState() it you need "more powerful state management"

   **** two or more state or vairable related then we can combine them into one state by using useReducer()

   =============== useReducer() Theory and implementation design ===============

   const [state, dispatchFn] = useReducer(reducerFn, initialState, initFn);

   state => the State snapshot used in the component re-render/re-evaluation cycle

   dispatchFn => A function That can be used to dspatch a new action (i.e trigger an update of the state)

   reducerFn => (prevState, action) => newState
            A function that is triggered automatically once an acton is dispatched (via dspatchFn()) - it receives the latest
            state snapshot and should return the new, update state.



   ================= useState() vs useReducer() =========

   Generally, you will know when you need useReducer() ->(when using useState() becomes cumbersome or you
    are getting a lot of bugs / unintended behaviors)

    useState()

   1) the main state management "tool"
   2) Great for independent pieces of state / data
   3) Great if state updates are easy and limited to a few kinds of updates

   useReducer()

   1) Great if you need "more Power"
   2) Should be considered if you have related pieces of state / data
   3) can be helpful if you have more complex state updates



   Object distructuring and work with only isValid mail and call useEffect() eficiently

   when the dependencies change less the call the useEffect() hook less

      const {isValid:  emailIsValid} = emailState;
      const {isValid: passwordIsValid} = passwordState;

   useEffect( ()=>{
      const identifier = setTimeout( () => {

         setFormIsValid(emailIsValid && passwordIsValid);
      }, 500);

      return () => {
         clearTimeout(identfier);
      }

   }, [emailIsValid, passwordIsValid])
```

```
67
68
69  */
70
71  // ========================== Folow Exmple to understand the concepts useReducer() ==============
72
73  import React, { useState, useReducer } from 'react';
74
75  import Card from '../UI/Card/Card';
76  import classes from './Login.module.css';
77  import Button from '../UI/Button/Button';
78
79  // create outside function
80  const emailReducer = (state, action) => {
81      if(action.type === 'USER_INPUT'){
82          return {value: action.val, isValid: action.val.includes('@') };
83      }
84      if(action.type === 'USER_BLUR'){
85
86          // it will work only the last state snapshot...
87
88          return {value: action.value, isValid: state.value.includes('@')}
89      }
90      return {value: '', isValid: false};
91  }
92
93  const Login = (props) => {
94
95      // combine this two by using useReducer()
96
97  //   const [enteredEmail, setEnteredEmail] = useState('');
98  //   const [emailIsValid, setEmailIsValid] = useState();
99    const [enteredPassword, setEnteredPassword] = useState('');
100   const [passwordIsValid, setPasswordIsValid] = useState();
101   const [formIsValid, setFormIsValid] = useState(false);
102
103   // create useReducer here
104   const [emailState, dispatchEmail] = useReducer(emailReducer, {value: '', isValid: null},);
105
106
107   const emailChangeHandler = (event) => {
108     // setEnteredEmail(event.target.value);
109     dispatchEmail({type: 'USER_INPU', val: event.target.value});
110
111     setFormIsValid(
112     //   event.target.value.includes('@') && enteredPassword.trim().length > 6
113     emailState.value.includes('@') && enteredPassword.trim().length > 6
114     );
115   };
116
117   const passwordChangeHandler = (event) => {
118     setEnteredPassword(event.target.value);
119
120
121     setFormIsValid(
122     //   event.target.value.trim().length > 6 && enteredEmail.includes('@')
123     event.target.value.trim().length > 6 && emailState.value.includes('@')
124     );
125   };
126
127   const validateEmailHandler = () => {
128     // setEmailIsValid(enteredEmail.includes('@'));
129     dispatchEmail({type: 'USER_BLUR'});
130   };
131
132   const validatePasswordHandler = () => {
133     setPasswordIsValid(enteredPassword.trim().length > 6);
```

```
  };

  const submitHandler = (event) => {
    event.preventDefault();
    // props.onLogin(enteredEmail, enteredPassword);
    props.onLogin(emailState.value, enteredPassword);
  };

  return (
    <Card className={classes.login}>
      <form onSubmit={submitHandler}>
        <div
          className={`${classes.control} ${
            // emailIsValid === false ? classes.invalid : ''
            emailState.isValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="email">E-Mail</label>
          <input
            type="email"
            id="email"
            // value={enteredEmail}
            value={emailState.value}
            onChange={emailChangeHandler}
            onBlur={validateEmailHandler}
          />
        </div>
        <div
          className={`${classes.control} ${
            passwordIsValid === false ? classes.invalid : ''
          }`}
        >
          <label htmlFor="password">Password</label>
          <input
            type="password"
            id="password"
            value={enteredPassword}
            onChange={passwordChangeHandler}
            onBlur={validatePasswordHandler}
          />
        </div>
        <div className={classes.actions}>
          <Button type="submit" className={classes.btn} disabled={!formIsValid}>
            Login
          </Button>
        </div>
      </form>
    </Card>
  );
};

export default Login;
```