main.jsx 공통

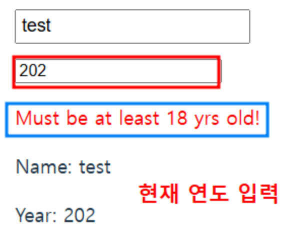
```jsx
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

App.jsx

```jsx
import './App.css'
import { useState } from 'react'

function App() {
  const [name, setName] = useState('')
  const [year, setYear] = useState('')
  const [warning, setWarning] = useState('')

  const handleNameChange = (newName) => {
    const formattedName = newName.trim().toLowerCase()
    setName(formattedName)
  }

  const handleYearChange = (newYear) => {
    const age = new Date().getFullYear() - newYear
    if (age < 18) {
      setWarning('Must be at least 18 yrs old!')
    } else {
      setWarning('')
      setYear(newYear)
    }
  }

  return (
    <div>
      <input
        type="text"
        placeholder="Enter name"
        value={name}
        onChange={
          (e) => handleNameChange(e.target.value)}
      />
      <input
        type="number"
        placeholder="Enter birth year"
        value={year}
        onChange={
          (e) => handleYearChange(e.target.value)}
      />
      {
        warning && (
          <p style={{ color: 'red' }}>{warning}</p>
        )
      }
```

```
        <p>Name: {name}</p>
        <p>Year: {year}</p>
      </div>
    )
  }
export default App
```

결과



```
test
202
Must be at least 18 yrs old!
Name: test
Year: 202
```

1. useReducer

- 여러 컴포넌트에서 상태를 공유할 때 사용
- reducer는 "상태"와 "액션"을 받아서 "새로운 상태"를 반환하는 함수이다

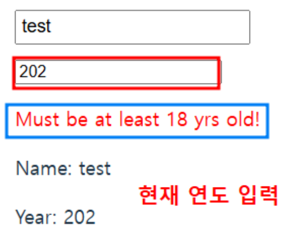
App.jsx

```jsx
import './App.css'
import { useState } from 'react'

// 1) useReducer
import React, { useReducer } from 'react' // {userReducer} 리엑트에서 제공되는 기능
import { userReducer, initialState } from './reducers/userReducer'

function App() {
  /*
   * 1) useReducer - 여러 컴포넌트에서 상태를 공유할 때 사용
   *    reducer는 "상태"와 "액션"을 받아서 "새로운 상태"를 반환하는 함수이다
   */
   const [state, dispatch] // 대부분 state, dispatch 라고 명시 (변수 이름은 변경가능)
    = useReducer(userReducer, initialState) // useReducer(함수, 초기값)

  return (
    <div>
      <input
        type="text" placeholder="Enter name"
        value={state.name}
        onChange={(e) => dispatch({
          // userReducer.js 에서
          // type은 userReducer(state, action)에서 action값으로 입력됨
          // payload는 userReducer(state, action)에서  return문의 ation.payload로 입력됨
          type: 'SET_NAME',  payload: e.target.value })}
      />
      <input
        type="number" placeholder="Enter birth year"
        value={state.year}
        onChange={(e) => dispatch({
          type: 'SET_YEAR', payload: e.target.value })}
      />
      {state.warning
       && <p style={{ color: 'red' }}>{state.warning}</p>}
      <p>Name: {state.name}</p>
      <p>Year: {state.year}</p>
    </div>
  )
}
export default App
```

userReducer.js

```
  /*
   * 1) useReducer - 여러 컴포넌트에서 상태를 공유할 때 사용
   *    reducer는 "상태"와 "액션"을 받아서 "새로운 상태"를 반환하는 함수이다
   */

  // 반드시 reducers 폴더에 넣어야 되지 않지만, 보통 폴더를 만들어 관리한다
  export const initialState = {
      name: '',
      year: '',
      warning: ''
    }

    export function userReducer(state, action) {
      switch (action.type) {
        case 'SET_NAME':
          return {
             ...state,
            name: action.payload.trim().toLowerCase() }
        case 'SET_YEAR':
          const age
           = new Date().getFullYear() - action.payload
          if (age < 18) {
            return {
               ...state,
              warning: 'Must be at least 18 yrs old!' }
          }
          return {
             ...state,
            year: action.payload,
            warning: '' }

        default:
          throw new Error('Unknown action type')
      }
    }
```

결과


test

202

Must be at least 18 yrs old!

Name: test

Year: 202

---

2. const [state,dispatch] = useReducer (userReducer,data,init) // data.js

App.jsx

```
  import './App.css'
```

```
import { useState } from 'react'

// 2) data.js
import React, { useReducer } from 'react'
import { userReducer, init } from './reducers/userReducer'
import data from './data'

function App() {
    // 2) data.js
    const [state,dispatch]
    // userReducer.js에 init은 init함수를 호출, data는 파라미터 값으로 들어감
    = useReducer (userReducer,data,init)

  return (
    <div>
      <input
        type="text" placeholder="Enter name"
        value={state.name}
        onChange={(e) => dispatch({
          // userReducer.js 에서
          // type은 userReducer(state, action)에서 action값으로 입력됨
          // payload는 userReducer(state, action)에서  return문의 ation.payload로 입력됨
          type: 'SET_NAME',  payload: e.target.value })}
      />
      <input
        type="number" placeholder="Enter birth year"
        value={state.year}
        onChange={(e) => dispatch({
          type: 'SET_YEAR', payload: e.target.value })}
      />
      {state.warning
       && <p style={{ color: 'red' }}>{state.warning}</p>}
      <p>Name: {state.name}</p>
      <p>Year: {state.year}</p>
      {/* 2) data.js */}
      <button onClick={
        () => dispatch({ type: 'RESET', payload: data })}>
        Reset
      </button>

    </div>
  )
}
export default App
```

userReducer.js

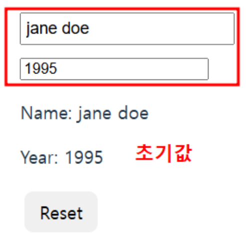
```js
/*
 * 1) useReducer - 여러 컴포넌트에서 상태를 공유할 때 사용
 *    reducer는 "상태"와 "액션"을 받아서 "새로운 상태"를 반환하는 함수이다
 */

// 반드시 reducers 폴더에 넣어야 되지 않지만, 보통 폴더를 만들어 관리한다
export const initialState = {
    name: '',
    year: '',
    warning: ''
  }

  export function userReducer(state, action) {
    switch (action.type) {
      case 'SET_NAME':
        return {
```

```
        ...state,
        name: action.payload.trim().toLowerCase() }
    case 'SET_YEAR':
      const age
       = new Date().getFullYear() - action.payload
      if (age < 18) {
        return {
          ...state,
          warning: 'Must be at least 18 yrs old!' }
      }
      return {
        ...state,
        year: action.payload,
        warning: '' }

    //  //2) data.jk
    case 'RESET':
    return init(action.payload)

    default:
      throw new Error('Unknown action type')
  }
}

// 2) data.js
export function init(externalData) {
  return {
    ...initialState,
    name: externalData.name,
    year: externalData.year
  }
}
```

data.js

```js
// 2) data.js
const externalData = {
    name: 'jane doe',
    year: 1995
  }
  export default externalData
```

결과



jane doe

1995

Name: jane doe

Year: 1995

---

3. countReducer.js

App.jsx

```jsx
import './App.css'
import { useState } from 'react'
```

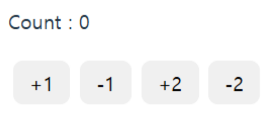
```

// 3) countReducer.js
import React, { useReducer } from 'react'
import { countReducer, initialState } from './reducers/countReducer'


function App() {
  // 3) countReducer.js
  const [state, dispatch]
  = useReducer(countReducer, initialState)

  const handleClick = (type, value, event) => {
    // clientX, clientY는 브라우저가 자동으로 제공하는 이벤트 객체의 속성
    const {clientX:x, clientY: y} = event
    dispatch({
      type, payload:{value}, meta:{x,y}
    })
  }

  return(
    <>
      <p>Count : {state.count}</p>
      <button onClick={(e) => handleClick('INC',1,e)}>+1</button>
      <button onClick={(e) => handleClick('DEC',1,e)}>-1</button>
      <button onClick={(e) => handleClick('INC',2,e)}>+2</button>
      <button onClick={(e) => handleClick('DEC',2,e)}>-2</button>
    </>
  )

}

export default App
```

countReducer.js

JS

```js
// 3) countReducer.js
export const initialState = {
    count: 0
  }

  export function countReducer(state, action) {
    const { value } = action.payload
    const { x, y } = action.meta
    switch (action.type) {
      case 'INC':
        console.log(`Click: (${x}, ${y})`)
        return {
           ...state,
          count: state.count + value
        }
      case 'DEC':
        console.log(`Click: (${x}, ${y})`)
        return {
           ...state,
          count: state.count - value
        }
      default:
        throw new Error('Unknown action type')
    }
  }
```

결과



Count : 0