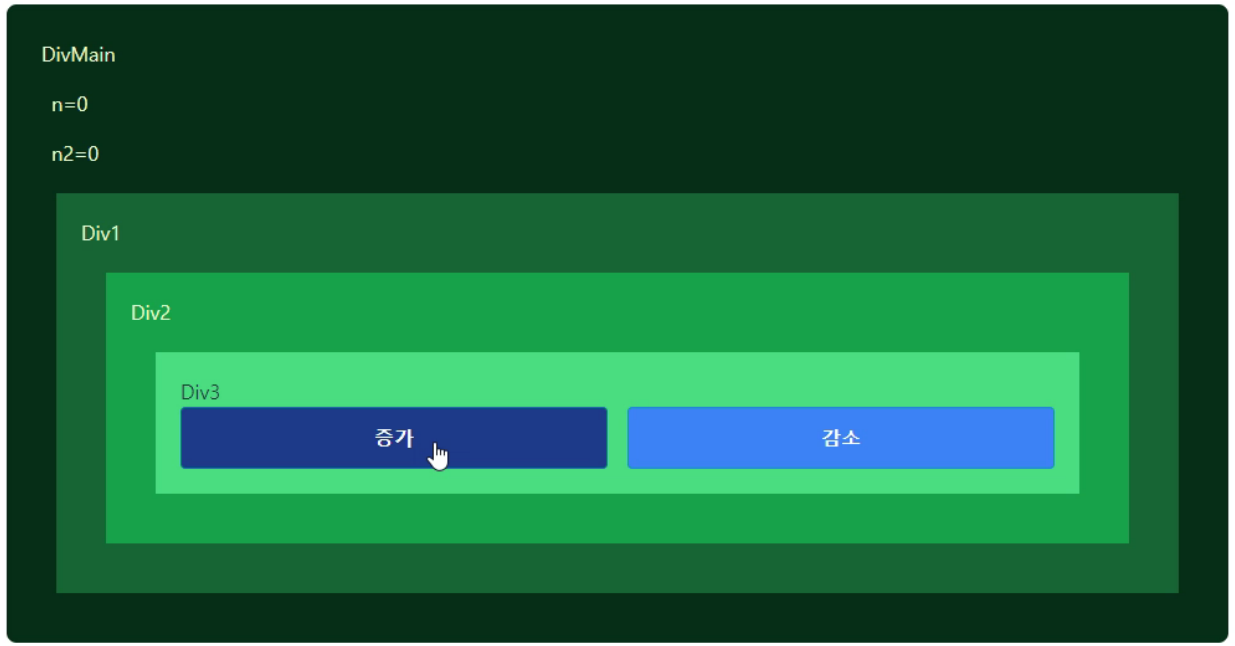
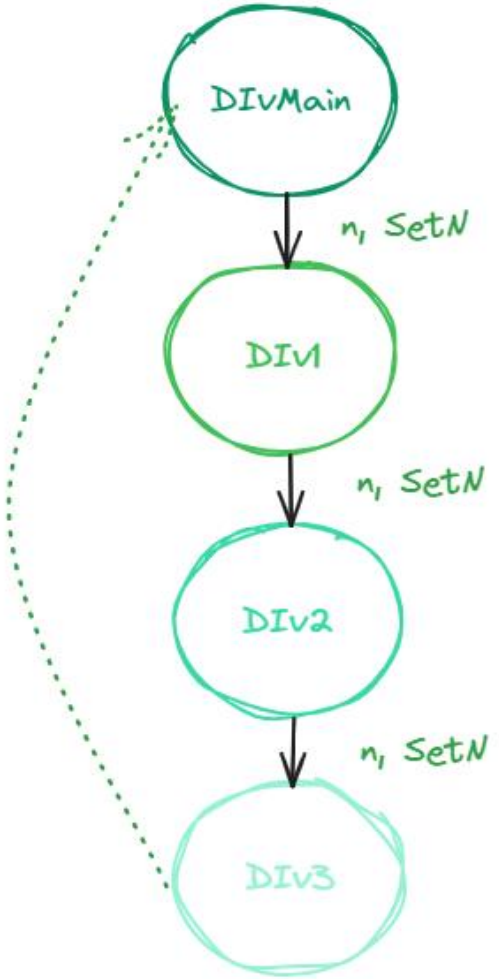
리액트(React) 기초

상태 관리

김경민

Probs를 활용한 상태 변경





JavaScript React JS

jotai

- 리액트를 위한 미니멀하고 유연한 상태 관리 라이브러리
  - Daishi Kato와 Poimandres 팀이 만든 오픈소스 프로젝트
  - Recoil에서 영감을 받았지만, 더 간단한 API와 작은 용량을 지향
    - 상태를 atom 단위로 관리하는 방식으로 Recoil이 호환되지 않는 React 버전(예: React 19) 사용
- 핵심개념 : 아톰 (Atom)
  - 상태의 최소 단위상태의 최소 단위
  - React의 useState와 비슷하지만, 컴포넌트 트리에 묶이지 않고 애플리케이션 전역에서 접근 가능
  - 각 아톰은 독립적인 상태 조각을 가짐
- 설치
  - npm install jotai

jotai 사용법

JotaiCnt.jsx

Jotai 전역 상태관리

count : 0

double count : 0

JotaiCnt.jsx

증가 감소

atoms.js

```jsx
import { atom } from "jotai";

//기본 Atom
export const cntAtom = atom(0) ;

//파생 Atom
export const dbCntAtom = atom((get) => get(cntAtom) * 2) ;
```

JotaiCnt.jsx

```jsx
import JotaiCntBt from "./JotaiCntBt"
import { useAtomValue } from "jotai"
import { cntAtom, dbCntAtom } from "./atoms"

export default function JotaiCnt() {
  const cnt = useAtomValue(cntAtom) ;
  const dbCnt = useAtomValue(dbCntAtom) ;
  return (
    <div className='w-full flex flex-col
                    justify-start items-center'>
      <h1 className='text-2xl font-bold p-5'>
        Jotai 전역 상태관리
      </h1>
      <div className="w-md border bg-amber-50 border-amber-200
                    rounded-xl flex flex-col m-5 p-5 justify-start items-start">
        <p className="text-xl font-bold text-blue-500">count : {cnt}</p>
        <p className="text-xl font-bold">double count : {dbCnt} </p>
      </div>
      <JotaiCntBt />
    </div>
  )
}
```

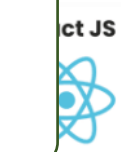
- useAtomValue 훅
  상태 값만 읽기 (읽기 전용)

JotaiCnt.jsx

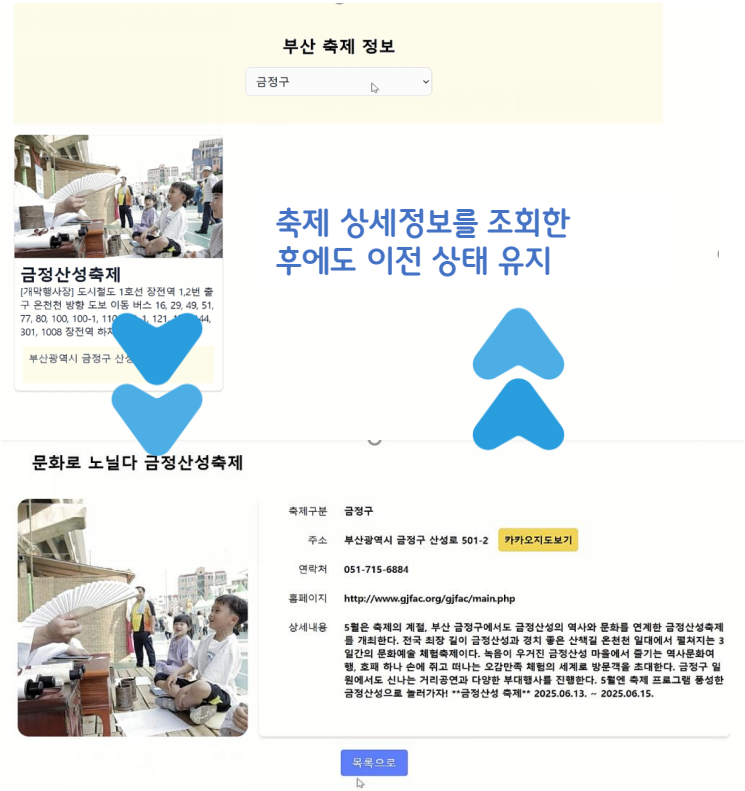
```jsx
import TailButton from "../components/TailButton"
import { useAtom } from "jotai"
import { cntAtom } from "./atoms"

export default function JotaiCntBt() {
  const [cnt , setCnt] = useAtom(cntAtom) ;

  return (
    <div>
      <TailButton color="blue" caption="증가"
                  onHandle={() => setCnt(cnt + 1)} />
      <TailButton color="orange" caption="감소"
                  onHandle={() => setCnt(cnt -1)} />
    </div>
  )
}
```

- useAtom 훅
  상태 값만 읽고 쓰기

JotaiCnt.jsx



Jotai atom에서 fetch



Jotai atom에서 fetch

atomsF.js

```jsx
import { atom } from "jotai";

export const selGuAtom = atom(null) ;

export const festivalFetchAtom = atom(async () => {
    const apikey = import.meta.env.VITE_API_KEY ;
    const baseUrl = 'https://apis.data.go.kr/6260000/FestivalService/getFestivalKr?' ;
    let url = `${baseUrl}serviceKey=${apikey}`;
    url = `${url}&pageNo=1&numOfRows=45&resultType=json`;

    const resp = await fetch(url) ;
    const data = await resp.json() ;
    return data.getFestivalKr.item
}) ;
```

1. FestivalContent 컴포넌트가 렌더링되면서 useAtom(festivalDataAtom)을 호출
2. festivalDataAtom은 아직 데이터를 가져오는 중이므로(Promise가 pending 상태), atom은 이 Promise 자체를 throw
3. React는 이 던져진 Promise를 감지하고, 가장 가까운 부모 트리에 있는 <Suspense> 컴포넌트를 찾음
4. <Suspense>는 이 상황을 처리하기 위해 자신의 fallback prop(예: Loading...)을 렌더링
5. 시간이 지나 데이터 로딩이 완료되면(Promise가 resolved 상태), React는 이 사실을 알고 FestivalContent 컴포넌트의 렌더링을 재개
6. 이때 useAtom은 실제 데이터를 반환하고, 정상적인 UI가 화면에 그려짐

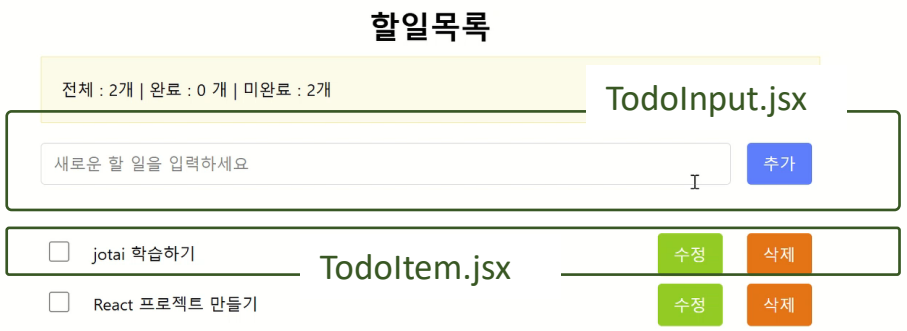
```jsx
import { selGuAtom, festivalFetchAtom } from "./atomsF";
import { Suspense } from "react";

export default function Festival() {
  return (
    <Suspense fallback={<div>Loading...</div>}>
      <FestivalContent />
    </Suspense>
  )
}

function FestivalContent() {
  const [tdata] = useAtom(festivalFetchAtom) ;
  const [gu, setGu] = useAtom(selGuAtom) ;
  const [area, setArea] = useState([]) ;
  const [areaFestival, setAreaFestival] = useState([]) ;

  const selRef = useRef() ;

  const handleChange = () => {
    setGu(selRef.current.value) ;
  }
  useEffect(() => {
    if (!gu) {
      setAreaFestival([]) ;
    } else {
      let tm = tdata.filter(item => item.GUGUN_NM == gu) ;
      setAreaFestival(tm) ;
    }
  }, [gu, tdata])
```

<Suspense> : 컴포넌트가 아직 렌더링할 준비가 되지 않았음을 리액트에게 알리고, 준비가 될 때까지 대체 UI(fallback)를 보여주도록 하는 선언적인 메커니즘

Festival.jsx

실습문제 : TodoList

TodoList.jsx



atomsTodo.js

```js
import { atom } from "jotai";

export const todoAtom = atom([
  { id: 1, text: 'jotai 학습하기', completed: false },
  { id: 2, text: 'React 프로젝트 만들기', completed: false },
])

export const compltedAtom = atom((get) => {
  const todos = get(todoAtom);
  return todos.filter(todo => todo.completed).length;
})
```