
Side Project

오늘의 날씨

2024.01

Portfolio

김현서

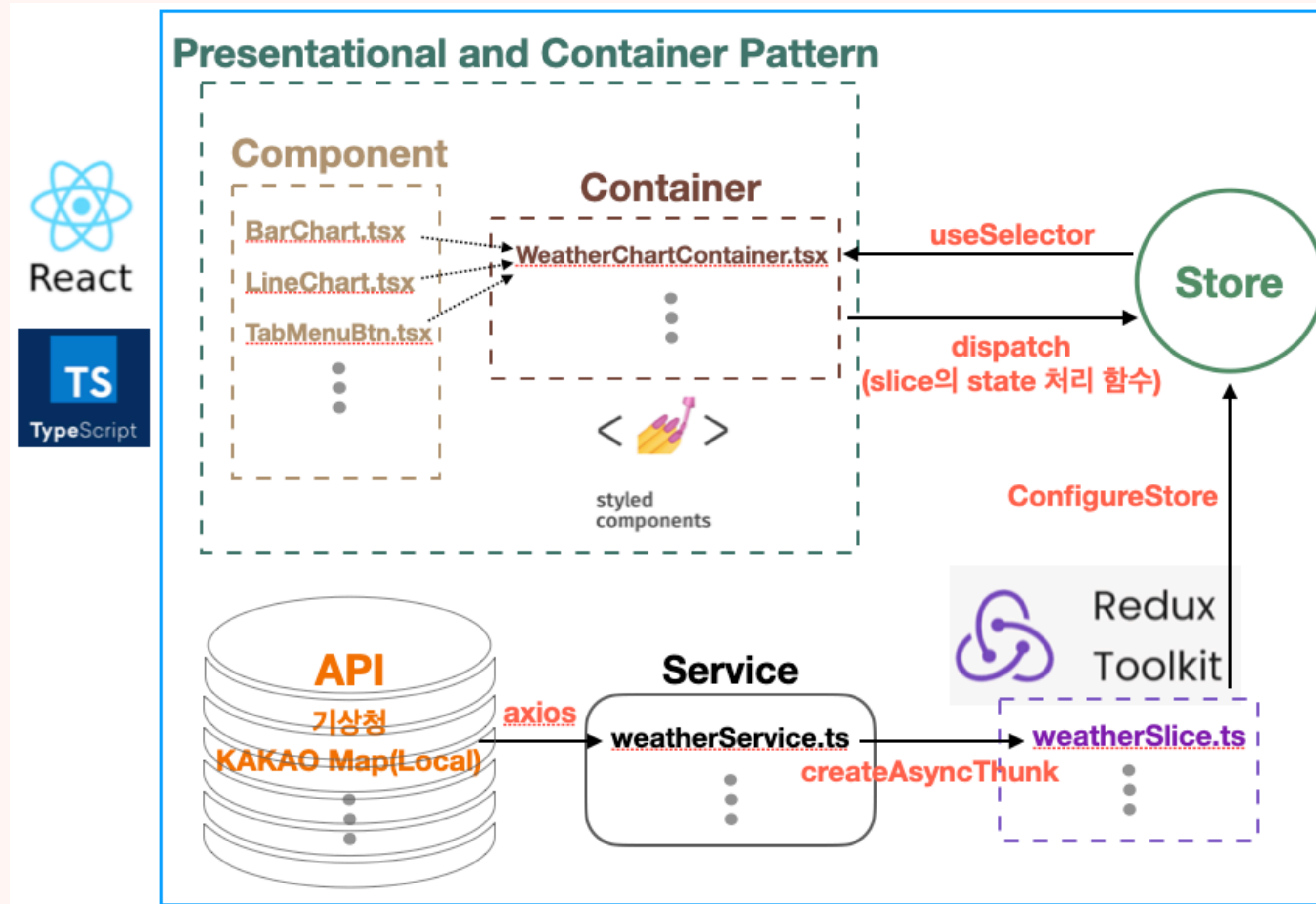
목차

- 사용기술
 - 아키텍처
 - 프로젝트 설명
 - 폴더 구조
 - 개발 기능
 - 기능 별 코드
-

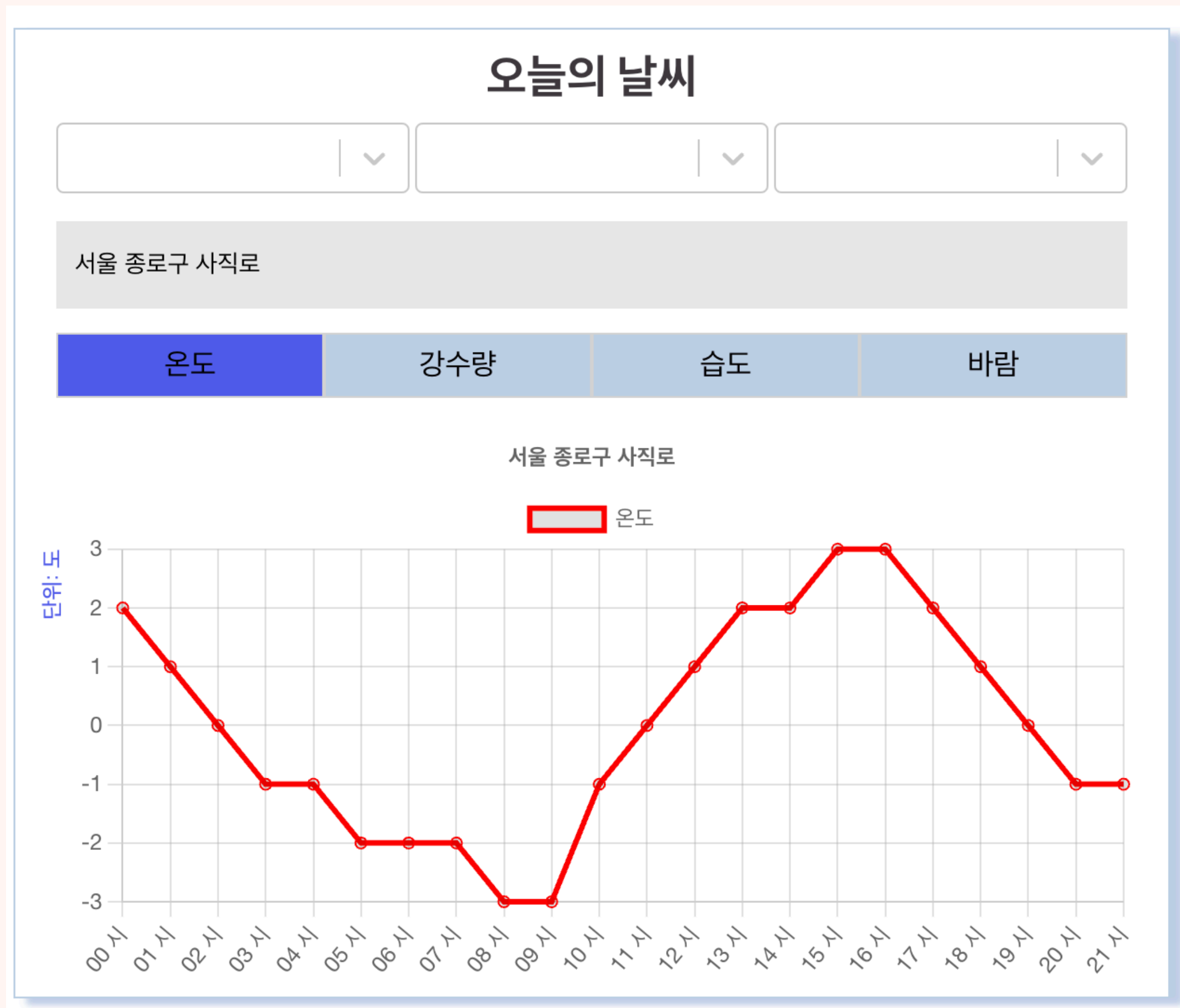
사용 기술

- TypeScript
 - React
 - Presentational and Container Pattern
 - Redux-toolkit
 - Axios
 - React-Chart
 - Styled-component
 - KAKAO Map(Local) API
-

아키텍처



프로젝트 설명



➤ 시, 도 / 구 / 동을 선택하거나 주소 입력을 해서 오늘의 날씨(온도, 강수량, 습도, 바람)를 그래프로 보여줍니다.

폴더 구조

Root Folder

- 전체 폴더

```
└─ today-weather
  ├── .git
  ├── .history
  ├── node_modules
  ├── public
  └─ src
      ├── components
      ├── containers
      ├── redux
      ├── services
      ├── theme
      ├── type
      ├── # App.css
      ├── App.test.tsx
      ├── TS App.tsx
      ├── # index.css
      └── TS index.tsx
```

Components

- 함수형 컴포넌트로 구현

```
└─ components
  ├── TS BarChart.tsx
  ├── TS CitySearch.tsx
  ├── TS CitySelect.tsx
  ├── TS LineChart.tsx
  ├── TS TabMenuButton.tsx
  └── TS Title.tsx
```

Containers

- 데이터를 다루어 Component에게 전달해 화면 구성

```
└─ containers
  ├── TS CitySearchContainer.tsx
  ├── TS CitySelectSet.tsx
  ├── TS MainContainer.tsx
  └── TS WeatherChartContainer.tsx
```

Redux

- RTK를 이용한 상태관리

```
└─ redux
  ├── city
  │   └── TS citySlice.ts
  ├── weather
  │   └── TS weatherSlice.ts
  └── TS store.ts
```

Services

- axios 을 이용한 API 호출

```
└─ services
  ├── TS citySearchService.ts
  └── TS weatherService.ts
```

Theme

- Css Theme 저장

```
└─ theme
  └── TS weatherTheme.ts
```

Type

- 공통 Data Type 저장

```
└─ type
  ├── TS cityType.ts
  └── TS weatherType.ts
```

개발 기능

➤ **Presentational and Container Pattern 사용**

1. Component, props 에 따라 독립적으로 동작하도록 구현(css 포함)
2. Container Component를 사용해서 layout 구성 및 데이터를 state와 주고 받아(useSelector 사용) Component에 뿌려줌

➤ **Redux Toolkit 사용**

1. createSlice, reducer, extraReducer, createAsyncThunk를 이용해 상태관리 및 비동기관리 구현

➤ **주소 선택**

1. 시, 도 / 구 / 동 의 상하위 개념의 Select 구현
2. 주소 검색

➤ **선택한 주소의 날씨 데이터 그래프로 표현**

1. 온도 - 라인 차트로 구현
2. 강수량, 습도, 바람 - 바 차트로 구현

➤ **탭 변경**

1. react-router-dom을 통해 탭 변경 구현
-

기능 별 코드(PRESENTATIONAL & CONTAINER PATTERN)

CitySelect.tsx

```
import React from 'react'
import Select, { PropsValue } from 'react-select'
import { Option } from '../type/cityType'
import styled from 'styled-components'

const SelectContainer = styled.div`
  width: 33%;
`

interface CitySelectType {
  value: PropsValue<Option | Option[]> | undefined
  options: Option[]
  onChange: (selections: PropsValue<Option | Option[]>) => void
}

const CitySelect = ({ value, options, onChange }: CitySelectType) => {
  return (
    <SelectContainer>
      <Select value={value} options={options} onChange={onChange} />
    </SelectContainer>
  )
}

export default CitySelect
```

CitySelectSet.tsx

```
return (
  <SelectContainer>
    <CitySelect
      value={firstSelectValue}
      options={firstSelectOptions}
      onChange={handleFirstSelectChange}
    />
    <CitySelect
      value={secondSelectValue}
      options={secondSelectOptions}
      onChange={handleSecondSelectChange}
    />
    <CitySelect
      value={thirdSelectValue}
      options={thirdSelectOptions}
      onChange={handleThirdSelectChange}
    />
  </SelectContainer>
)
```

Container인 CitySelectSet.tsx에서 Component인 CitySelect 사용, CitySelect는 전달받은 Props를 통해 독립적으로 동작하고 필요한 css요소를 styled-component로 가짐
CitySelectSet.tsx는 useSelector를 통해 데이터 전달

기능 별 코드 (REDUX TOOLKIT 사용)

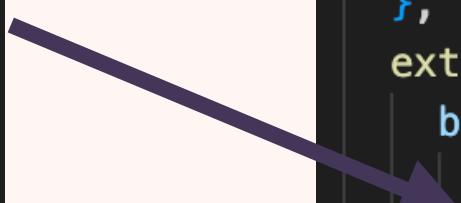
citySlice.ts

Action 생성

```
export const fetchFirstSelectOptions = createAsyncThunk(
  'city/fetchFirstSelectOptions',
  async () => {
    const newOptions: any = await getMainCityAddress()
    const newData: Option[] = newOptions.regcodes.map((item: any) => ({
      value: item.code,
      label: item.name,
    }))
    return newData
  }
)
```

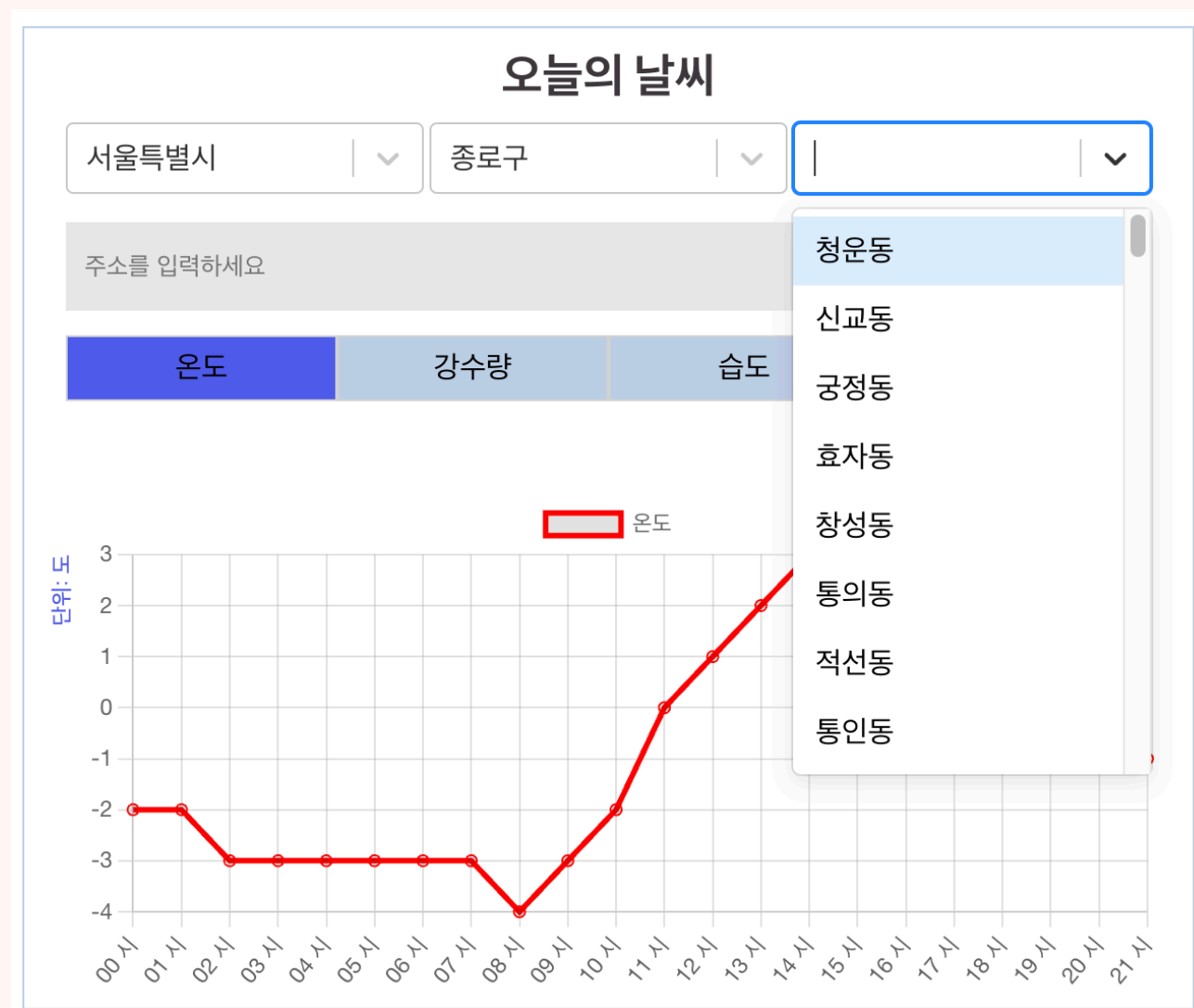
extraReducer를 통해 비동기 상태관리

```
const citySlice = createSlice({
  name: 'city',
  initialState,
  reducers: { ...
},
  extraReducers: (builder) => {
    builder
      .addCase(fetchFirstSelectOptions.fulfilled, (state, action) => {
        state.firstSelectOptions = action.payload
      })
      .addCase(fetchSecondSelectOptions.fulfilled, (state, action) => { ...
      })
      .addCase(fetchThirdSelectOptions.fulfilled, (state, action) => { ...
      })
      .addCase(fetchXYCoordinateBySearch.fulfilled, fetchXYCoordinateFulfilled)
      .addCase(fetchXYCoordinateBySelect.fulfilled, fetchXYCoordinateFulfilled)
      .addCase(fetchCityItems.fulfilled, (state, action) => { ...
      })
  },
})
```



기능 별 코드(시, 도/ 구 / 동 선택)

CitySelectSet.tsx



```
useEffect(() => {
  dispatch(fetchFirstSelectOptions())
}, [dispatch])

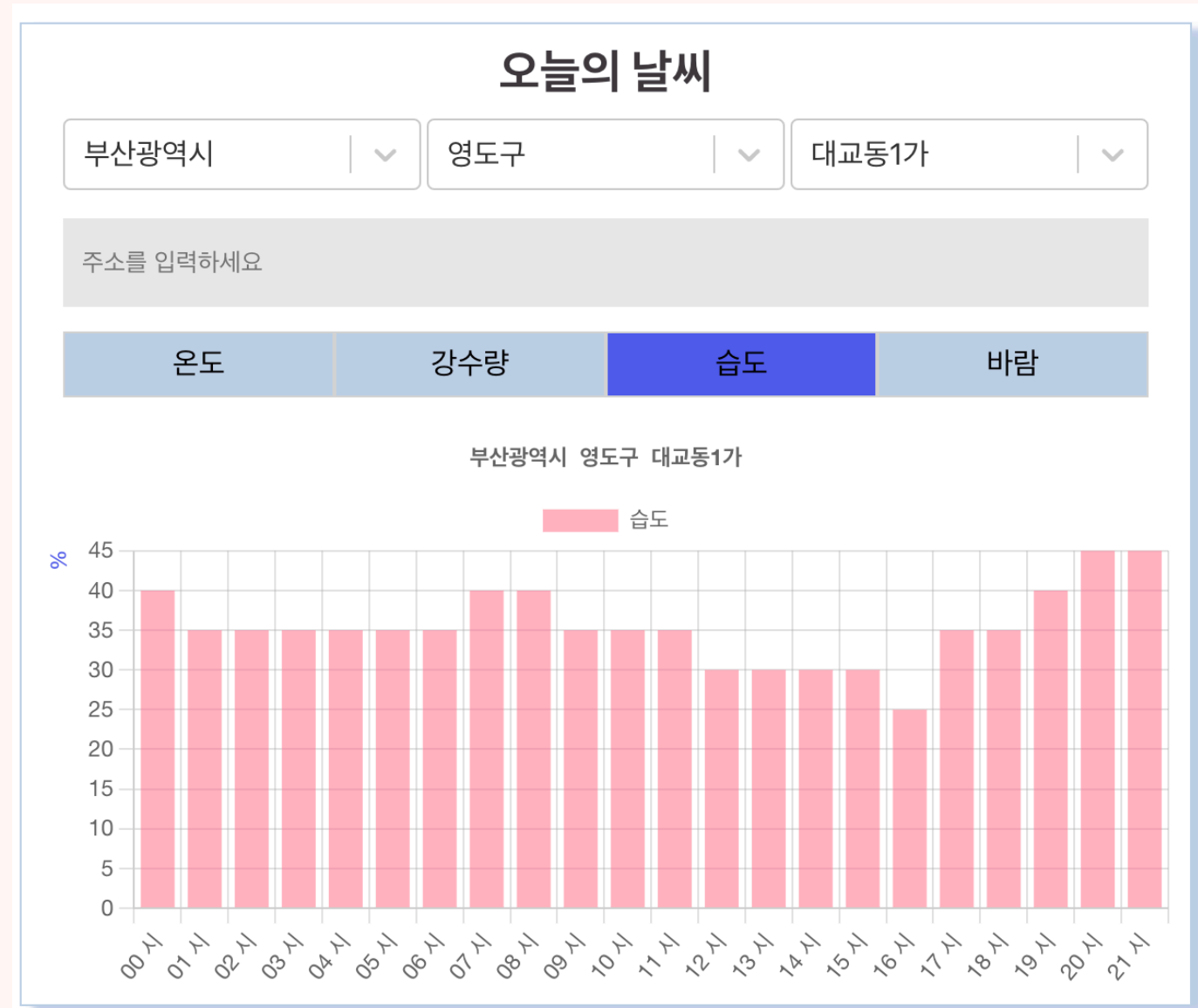
useEffect(() => {
  // Initial fetch or generate options for the second select
  if (firstSelectValue.value !== '') {
    dispatch(fetchSecondSelectOptions(firstSelectValue))
  }
}, [dispatch, firstSelectValue])

useEffect(() => {
  // Initial fetch or generate options for the third select
  if (secondSelectValue.value !== '') {
    dispatch(fetchThirdSelectOptions([firstSelectValue, secondSelectValue]))
  }
}, [dispatch, secondSelectValue])
```

```
return (
  <SelectContainer>
    <CitySelect
      value={firstSelectValue}
      options={firstSelectOptions}
      onChange={handleFirstSelectChange}
    />
    <CitySelect
      value={secondSelectValue}
      options={secondSelectOptions}
      onChange={handleSecondSelectChange}
    />
    <CitySelect
      value={thirdSelectValue}
      options={thirdSelectOptions}
      onChange={handleThirdSelectChange}
    />
  </SelectContainer>
)
```

dispatch를 통해 citySlice.ts에서 해당되는 단위의 리스트를 가져와 상태 처리 후 useSelector를 통한 반환

기능 별 코드(선택한 주소의 날씨 데이터 그래프로 표현)



MainContainer.tsx

```
useEffect(() => {
  if (xyCoordinate.x !== 0 && xyCoordinate.y !== 0) {
    const [x, y] = changeXY(xyCoordinate)
    setX(x)
    setY(y)
  }, [dispatch, xyCoordinate])

return (
  <MainContainerLayout>
    <MainLayout>
      <Title value="오늘의 날씨" />
      <CitySelectSet
        firstSelectValue={firstSelectValue}
        secondSelectValue={secondSelectValue}
        thirdSelectValue={thirdSelectValue}
      />
      <CitySearchContainer />
      <WeatherChartContainer x={x} y={y} />
    </MainLayout>
  </MainContainerLayout>
)
```

WeatherChartContainer.tsx

```
useEffect(() => {
  dispatch(fetchWeatherInfo({ x, y }))
}, [dispatch, x, y])

<Routes>
  <Route
    path="/"
    element={
      <LineChart
        labels={temperatureLabels}
        data={temperatureData}
        labelTitle="온도"
        address={address}
      />
    }
  />
</Routes>
```

weatherSlice.ts에서 state값(날씨정보) 변화 후 useSelector를 통해 값 전달

기능 별 코드(탭 변경)

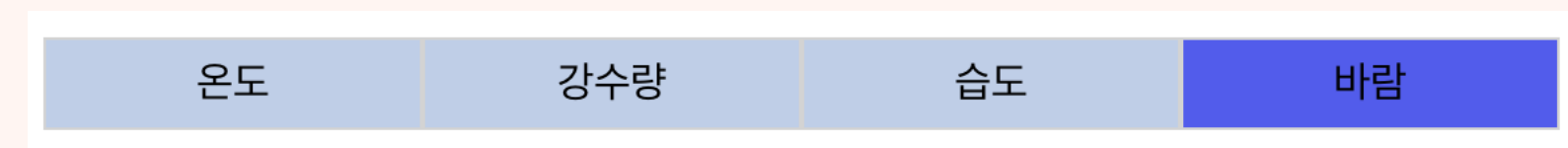
```
<Router>
  <TabContainer>
    <TabMenuButton
      url="/"
      title="온도"
      selected={selected}
      setSelected={setSelected}
    />
    <TabMenuButton
      url="/humiditybarchart"
      title="강수량"
      selected={selected}
      setSelected={setSelected}
    />
    <TabMenuButton
      url="/precipitationbarchart"
      title="습도"
      selected={selected}
      setSelected={setSelected}
    />
    <TabMenuButton
      url="/windbarchart"
      title="바람"
      selected={selected}
      setSelected={setSelected}
    />
  </TabContainer>
</Router>
```

```
<Routes>
  <Route
    path="/"
    element={
      <LineChart
        labels={temperatureLabels}
        data={temperatureData}
        labelTitle="온도"
        address={address}
      />
    }
  />
  <Route
    path="/humiditybarchart"
    element={
      <BarChart
        labels={humidityLabels}
        data={humidityData}
        labelTitle="강수량"
        address={address}
        unit="mm"
      />
    }
  />
</Routes>
```

TabButton 선택 시 라우터 이동

```
const TabMenuButton = ({ url, title, selected, setSelected }: TabMenuType) => {
  const TabButton = styled.button`
    width: 100%;
    height: 100%;
    font-size: 16px;
    background-color: ${({ theme }) =>
      url === selected
        ? theme.color.tabMenu.selected
        : theme.color.tabMenu.default};
    border: 1px solid lightgray;
  `

  return (
    <TabMenuButtonContainer onClick={() => setSelected(url)}>
      <StyledLink to={url}>
        <TabButton>{title}</TabButton>
      </StyledLink>
    </TabMenuButtonContainer>
  )
}
```



선택된 url과 자신의 props url이 일치할 경우 배경색 변경