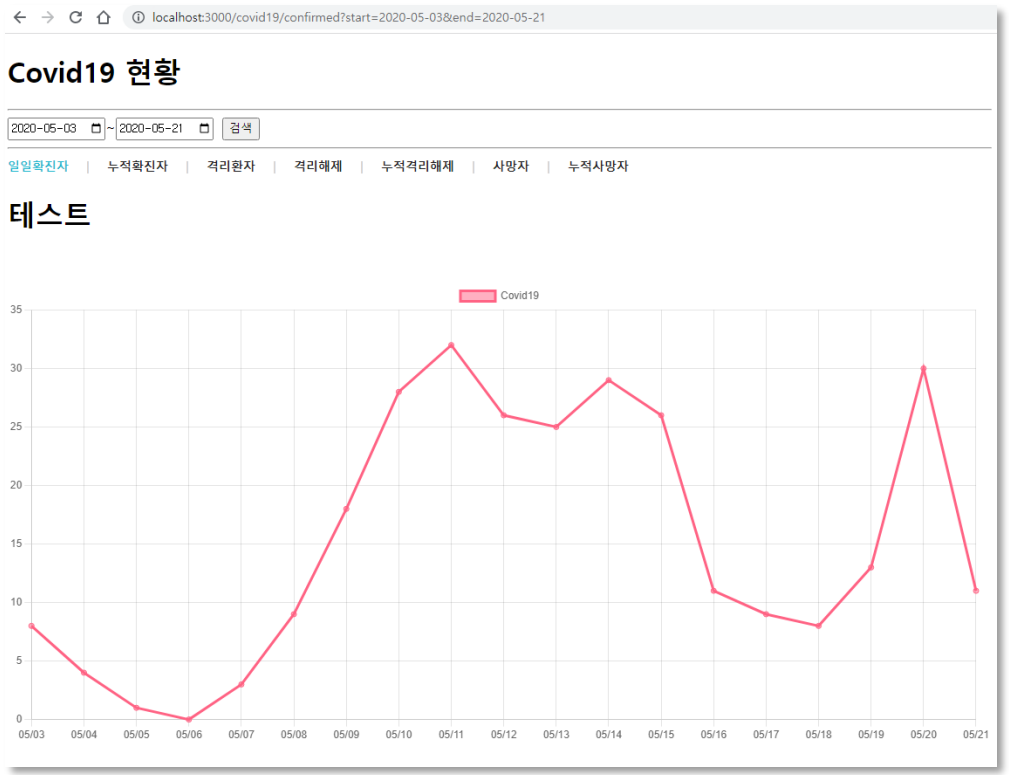
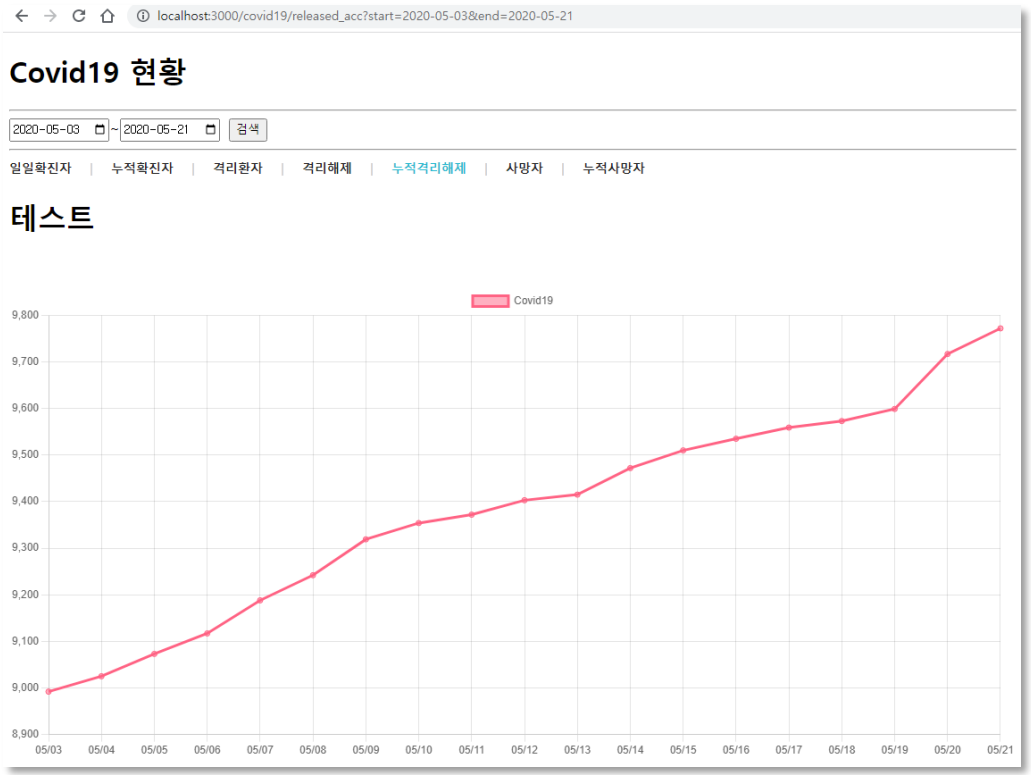
React 중간평가 과제

방미소

1. 최종과제 시연


localhost:3000/covid19/confirmed?start=2020-05-03&end=2020-05-21
Covid19 현황
2020-05-03 ~ 2020-05-21 검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
테스트
Covid19
35
30
25
20
15
10
5
0
05/03 05/04 05/05 05/06 05/07 05/08 05/09 05/10 05/11 05/12 05/13 05/14 05/15 05/16 05/17 05/18 05/19 05/20 05/21

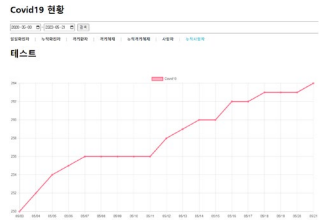

localhost:3000/covid19/released_acc?start=2020-05-03&end=2020-05-21
Covid19 현황
2020-05-03 ~ 2020-05-21 검색
일일확진자 | 누적확진자 | 격리환자 | 격리해제 | 누적격리해제 | 사망자 | 누적사망자
테스트
Covid19
9,800
9,700
9,600
9,500
9,400
9,300
9,200
9,100
9,000
8,900
05/03 05/04 05/05 05/06 05/07 05/08 05/09 05/10 05/11 05/12 05/13 05/14 05/15 05/16 05/17 05/18 05/19 05/20 05/21


Covid19 현황
테스트

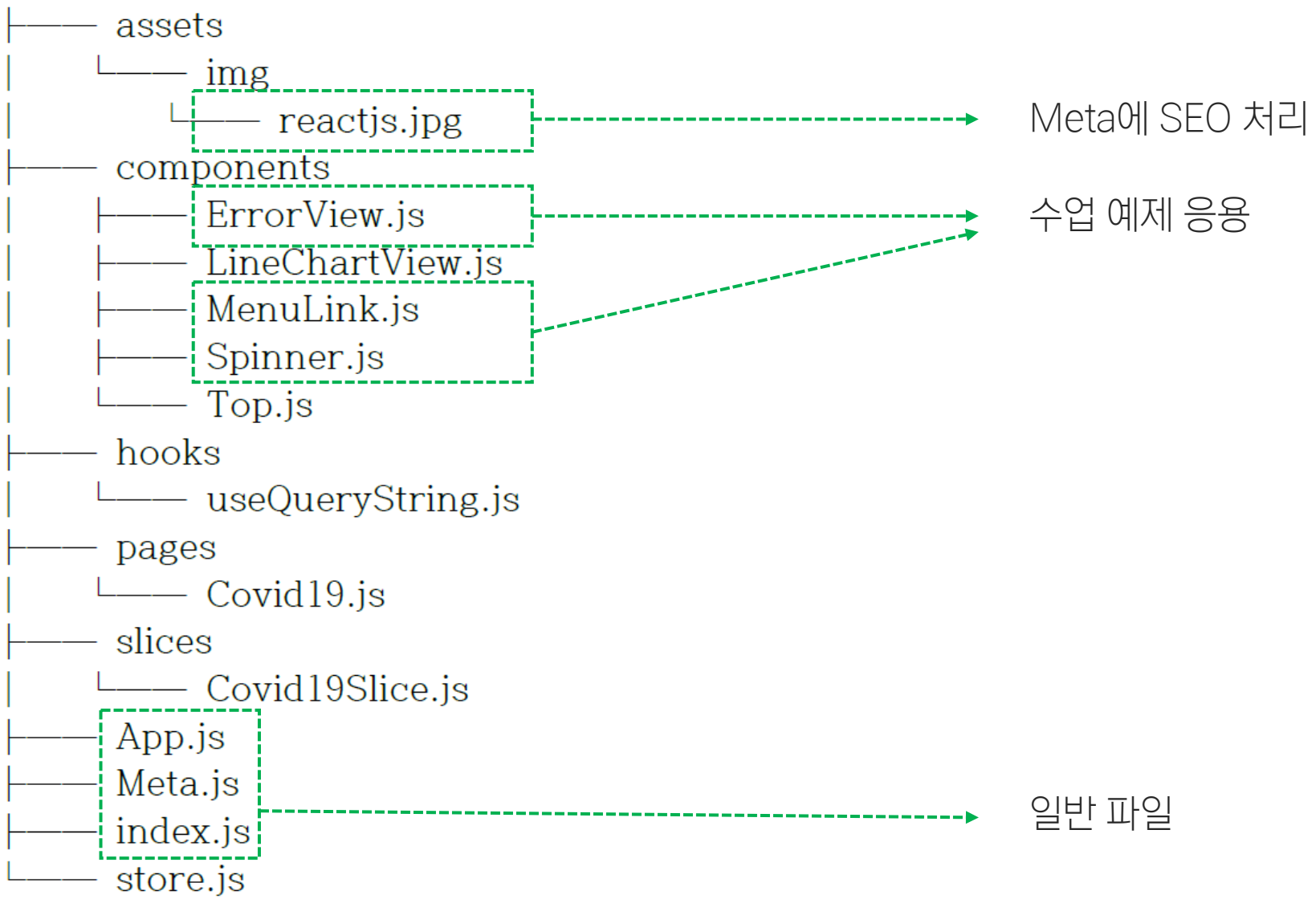

Covid19 현황
테스트


Covid19 현황
테스트


Covid19 현황
테스트


Covid19 현황
테스트

2. 설계


assets
img
reactjs.jpg
components
ErrorView.js
LineChartView.js
MenuLink.js
Spinner.js
Top.js
hooks
useQueryString.js
pages
Covid19.js
slices
Covid19Slice.js
App.js
Meta.js
index.js
store.js
Meta에 SEO 처리
수업 예제 응용
일반 파일

2. 설계

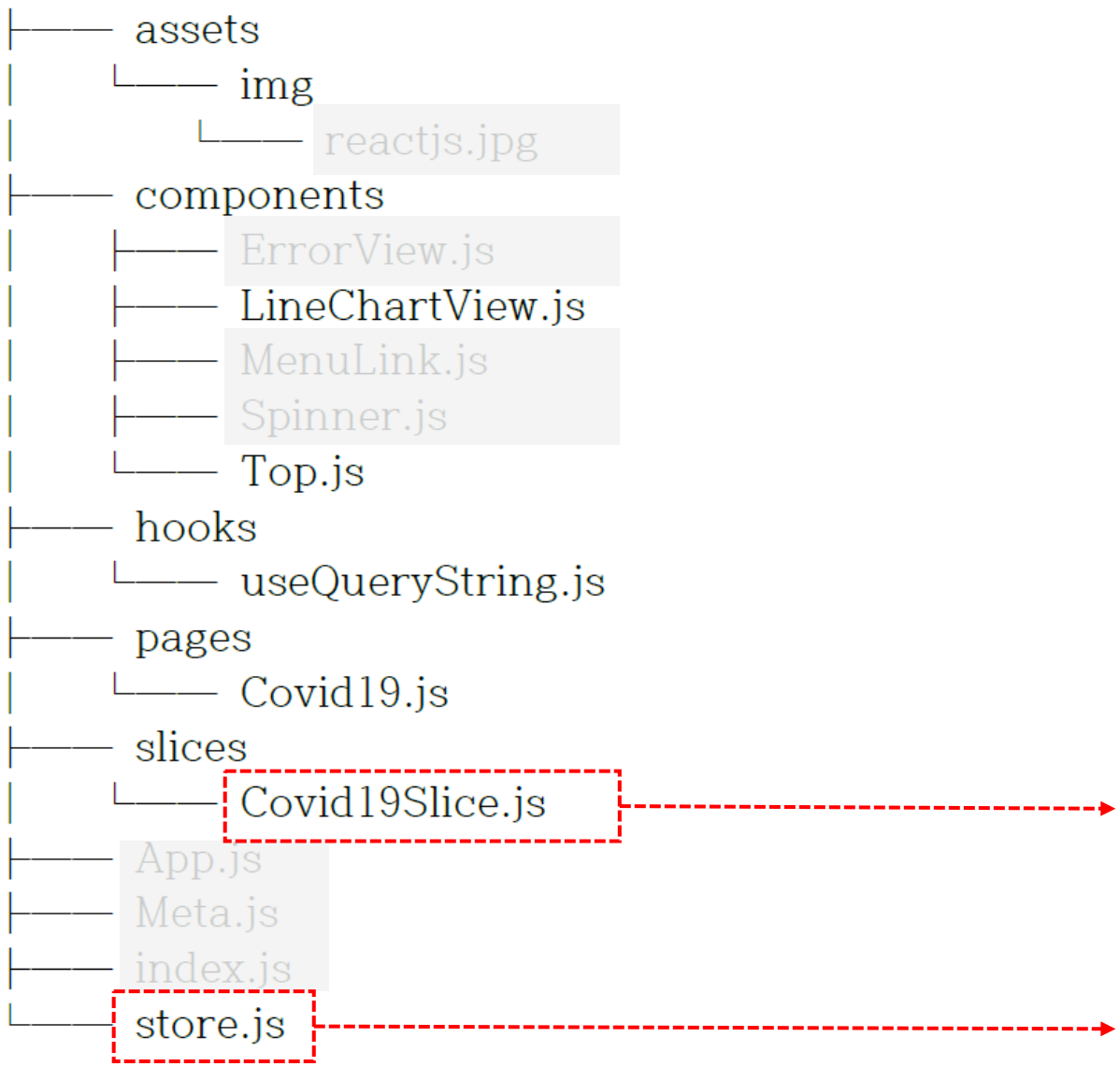
index.js

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";

import App from "./App";

import { Provider } from "react-redux";
import store from "./store";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <BrowserRouter>
        <App />
      </BrowserRouter>
    </Provider>
  </React.StrictMode>
);
```

App.js

```
import Top from "./components/Top";
import Covid19 from "./pages/Covid19";

import useQueryString from "./hooks/useQueryString";

function App() {
  useQueryString();
  return (
    <div className="App">
      <Top />
      <Covid19 />
    </div>
  );
}

export default App;
```

2. 설계



URL의 쿼리부분에 날짜를 반영하여
코로나19 데이터 디스패치

리덕스 스토어

2. 설계

store.js

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

import Covid19Slice from "./slices/Covid19Slice";

const store = configureStore({
  reducer: {
    covid19List: Covid19Slice,
  },

  middleware: (getDefaultMiddleware) => getDefaultMiddleware({ serializableCheck: false }),
  devTools: true,
});

export default store;
```

2. 설계

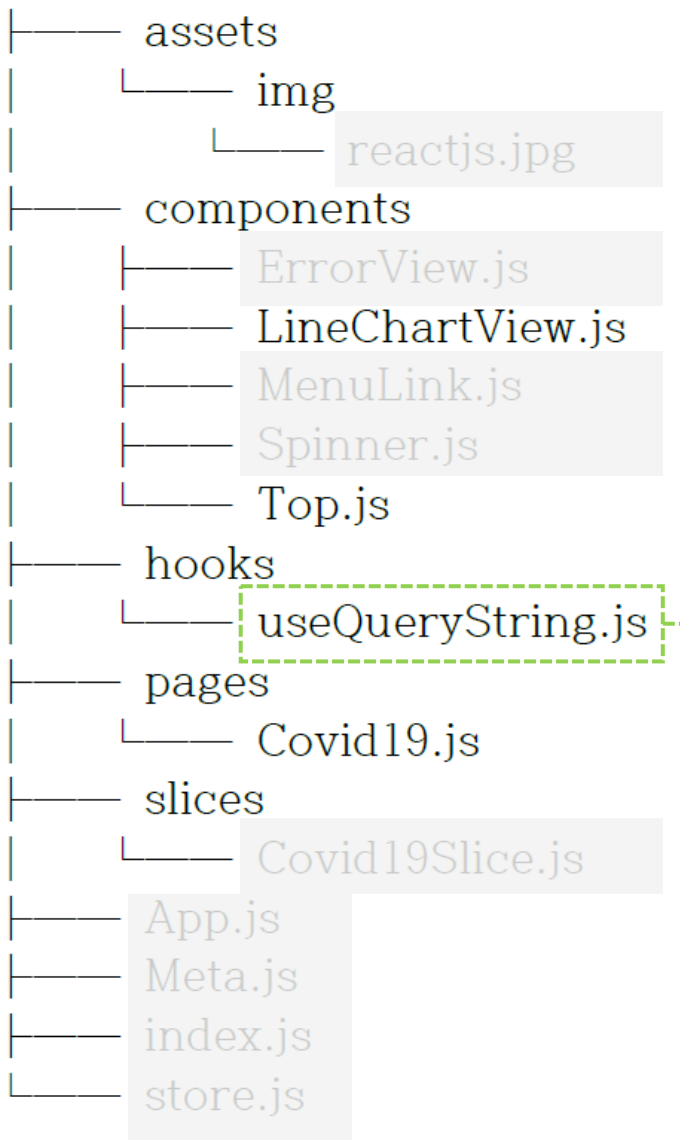
Covid19Slice.js

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

const URL = "http://localhost:3001/covid19";

export const getCovid19List = createAsyncThunk(
  "Covid19Slice/getCovid19List",
  async (payload = null, { rejectWithValue }) => {
    let result = null;

    try {
      if (payload === null) {
        result = await axios.get(URL);
      } else {
        result = await axios.get(
          `${URL}?date_gte=${payload.params.start}&date_lte=${payload.params.end}`
        );
      }
    } catch (err) {
      result = rejectWithValue(err.response);
    }

    return result;
  }
);

const Covid19Slice = createSlice({
  name: "covid19List",
  initialState: {
    data: null,
    loading: false,
    error: null,
  },
  reducer: {},
  extraReducers: {
    [getCovid19List.pending]: (state, { payload }) => {
      return { state, loading: true };
    },
    [getCovid19List.fulfilled]: (state, { payload }) => {
      return {
        data: payload?.data,
        loading: false,
        error: null,
      };
    },
    [getCovid19List.rejected]: (state, { payload }) => {
      return {
        data: payload?.data,
        loading: false,
        error: {
          code: payload?.status ? payload.status : 500,
          message: payload?.statusText ? payload.statusText : "Server Error",
        },
      };
    },
  },
});

export default Covid19Slice.reducer;
```

```
try {
  if (payload === null) {
    result = await axios.get(URL);
  } else {
    result = await axios.get(
      `${URL}?date_gte=${payload.params.start}&date_lte=${payload.params.end}`
    );
  }
} catch (err) {
  result = rejectWithValue(err.response);
}
```

url 추가 처리

2. 설계



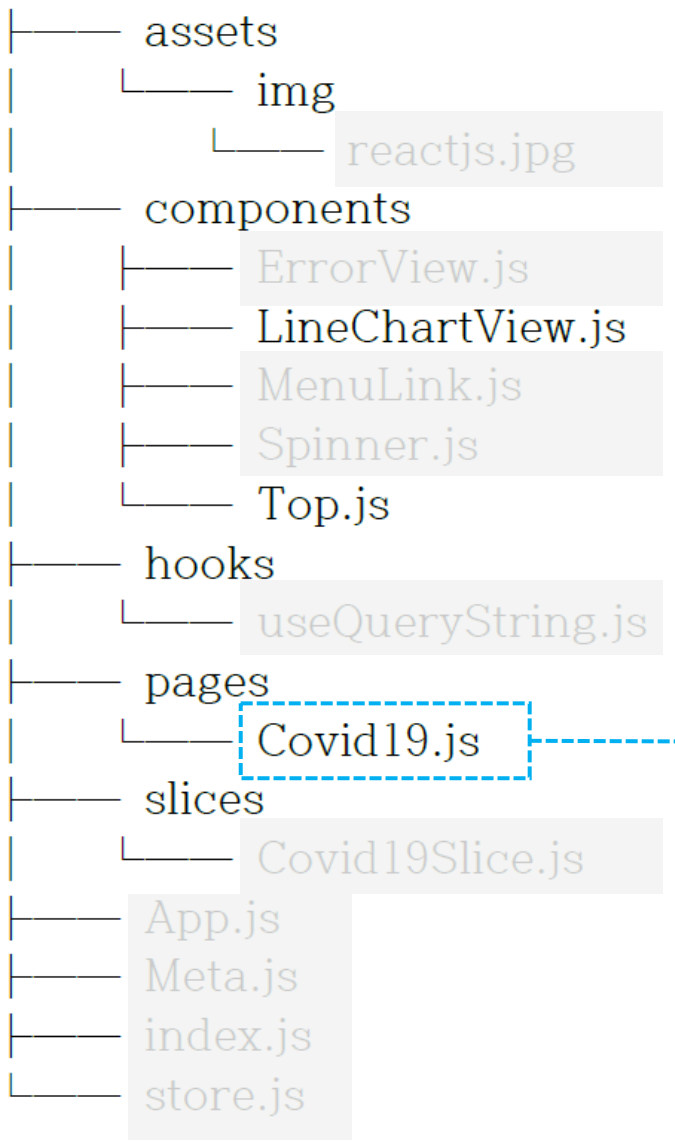
현재 url을 받아 쿼리 부분을 분리시켜
시작 / 끝 날짜를 배열로 반환

2. 설계

useQueryString.js

```
import React from "react";

const useQueryString = () => {
  const query = window.location.search;
  const searchParams = new URLSearchParams(query);
  const dates = Array.from(searchParams.values());

  return dates;
};

export default useQueryString;
```

2. 설계



날짜가 변경되는 것을 useEffect로 감지 후
데이터 디스패치하여 차트로 출력함

2. 설계

Covid19.js

```jsx
import React, { memo, useEffect } from "react";
import { useLocation } from "react-router-dom";

import MenuLink from "../components/MenuLink";
import Spinner from "../components/Spinner";
import ErrorView from "../components/ErrorView";
import LineChartView from "../components/LineChartView";

import useQueryString from "../hooks/useQueryString";

import { useSelector, useDispatch } from "react-redux";
import { getCovid19List } from "../slices/Covid19Slice";

import dayjs from "dayjs";
```

```jsx
const Covid19 = memo(() => {
  // 쿼리스트링 가져오기
  const [startDate, lastDate] = useQueryString();

  // 현재 페이지 필드명 가져오기
  const location = useLocation();
  let pageName = location.pathname.split("/").pop();

  // Redux 처리 : 마운트 시, 날짜 변경 시 디스패치
  const { data, loading, error } = useSelector((state) => state.covid19List);
  const dispatch = useDispatch();

  useEffect(() => {
    dispatch(
      getCovid19List({
        params: {
          start: dayjs(startDate).format("YYYY-MM-DDTHH:mm:ss[Z]"),
          end: dayjs(lastDate).format("YYYY-MM-DDTHH:mm:ss[Z]"),
        },
      })
    );
  }, [dispatch, startDate, lastDate]);

  return (
    <div>
      <Spinner visible={loading} />
      <nav>
        <MenuLink to="/covid19/confirmed">일일확진자</MenuLink>
        <MenuLink to="/covid19/confirmed_acc">누적확진자</MenuLink>
        <MenuLink to="/covid19/active">격리환자</MenuLink>
        <MenuLink to="/covid19/released">격리해제</MenuLink>
        <MenuLink to="/covid19/released_acc">누적격리해제</MenuLink>
        <MenuLink to="/covid19/death">사망자</MenuLink>
        <MenuLink to="/covid19/death_acc">누적사망자</MenuLink>
      </nav>

      {error ? (
        <ErrorView error={error} />
      ) : (
        <>
          <LineChartView chartData={data} fieldName={pageName} />
        </>
      )}
    </div>
  );
});

export default Covid19;
```

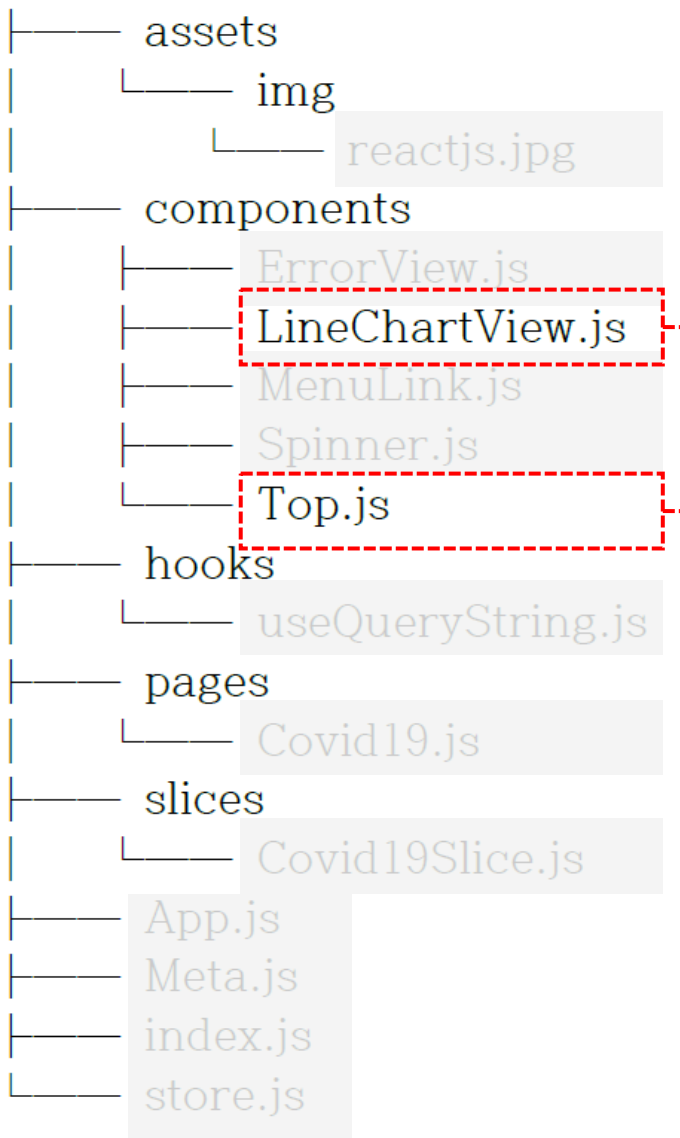
2. 설계

Covid19.js

```
useEffect(() => {
  dispatch(
    getCovid19List({
      params: {
        start: dayjs(startDate).format("YYYY-MM-DDTHH:mm:ss[Z]"),
        end: dayjs(lastDate).format("YYYY-MM-DDTHH:mm:ss[Z]"),
      },
    })
  );
}, [dispatch, startDate, lastDate]);
```

JSON 파일에 명시된 날짜 양식으로 포맷 맞춰 줌

2. 설계



pages.js에서 데이터를 받아
그래프 출력

날짜 변경 처리

2. 설계

Top.js

```
import React, { useState, useEffect, useCallback, useRef, memo } from "react";
import styled from "styled-components";

import useQueryString from "../hooks/useQueryString";

import dayjs from "dayjs";

const TopContainer = styled.header`
  input {
    line-height: 20px;
  }

  button {
    margin-left: 10px;
    line-height: 20px;
  }

  .form-container {
    position: sticky;
    top: 0;
    z-index: 1000;
  }
`;
```

```
const Top = memo(() => {
  // 쿼리스트링 가져오기
  const [startDate, lastDate] = useQueryString();

  // 타겟 날짜 기록
  const [targetDt, setTargetDt] = useState({
    start: "",
    end: "",
  });
  const start = useRef();
  const end = useRef();

  // 이벤트 : 검색 시
  const onSearchDate = useCallback((e) => {
    setTargetDt({
      ...targetDt,
      start: dayjs(start.current.value),
      end: dayjs(end.current.value),
    });
    console.log(targetDt.start, targetDt.end);
  });

  return (
    <TopContainer>
      <h1>Covid19 현황</h1>

      <hr />
      <div className="form-container">
        <form method="get" onSubmit={onSearchDate}>
          <input
            type="date"
            name="start"
            value={startDate || targetDt.start}
            onChange={(e) => {
              setTargetDt({ ...targetDt, start: e.target.value });
            }}
          />
          ~
          <input
            type="date"
            name="end"
            value={lastDate || targetDt.last}
            onChange={(e) => {
              setTargetDt({ ...targetDt, last: e.target.value });
            }}
          />
          <button type="submit">검색</button>
        </form>
      </div>
      <hr />
    </TopContainer>
  );
});

export default Top;
```

2. 설계

Top.js

```
const Top = memo(() => {
  // 쿼리스트링 가져오기
  const [startDate, lastDate] = useQueryString();

  // 타겟 날짜 기록
  const [targetDt, setTargetDt] = useState({
    start: "",
    end: "",
  });
  const start = useRef();
  const end = useRef();
```

url에서 submit된 날짜를 가져와 input태그 기본값으로 넣어 줌

```
<form method="get" onSubmit={onSearchDate}>
  <input
    type="date"
    name="start"
    value={startDate || targetDt.start}
    onChange={(e) => {
      setTargetDt({ ...targetDt, start: e.target.value });
    }}
  />
  ~
  <input
    type="date"
    name="end"
    value={lastDate || targetDt.last}
    onChange={(e) => {
      setTargetDt({ ...targetDt, last: e.target.value });
    }}
  />
  <button type="submit">검색</button>
</form>
```

이 때, onChange 이벤트 없이 기본값만을 적용하면 input 태그가 수정 불가능한 읽기 전용 폼이 되므로 onChange 이벤트를 별도 적용

2. 설계

LineChartView.js

```jsx
import React, { memo } from "react";

import {
  Chart as ChartJS,
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend,
} from "chart.js";

import { Line } from "react-chartjs-2";
import dayjs from "dayjs";

ChartJS.register(CategoryScale, LinearScale, PointElement, LineElement, Title, Tooltip, Legend);

const LineChartView = memo(({ chartData, fieldName }) => {
  if (chartData && fieldName) {
    const options = {
      responsive: true,
      plugins: {
        legend: {
          position: "top",
        },
        title: {
          display: true,
        },
      },
    };

    const labels = chartData.map((v) => dayjs(v.date).format("MM/DD"));
    // console.log();

    const data = {
      labels,
      datasets: [
        {
          label: "Covid19",
          data: labels.map((v, i) => {
            return chartData[i][fieldName];
          }),
          borderColor: "rgb(255, 99, 132)",
          backgroundColor: "rgba(255, 99, 132, 0.5)",
        },
      ],
    };

    return <Line options={options} data={data} />;
  }

  return <></>;
});

export default LineChartView;
```

```jsx
const labels = chartData.map((v) => dayjs(v.date).format("MM/DD"));

const data = {
  labels,
  datasets: [
    {
      label: "Covid19",
      data: labels.map((v, i) => {
        return chartData[i][fieldName];
      }),
      borderColor: "rgb(255, 99, 132)",
      backgroundColor: "rgba(255, 99, 132, 0.5)",
    },
  ],
};
```

전송된 데이터를 가공하여 활용

소요시간

- 코딩 : 15시간 전후

후기

- React와 Redux에 대한 이해가 아주 낮은 상태에서 시험을 치르게 되어 새 기능을 넣을 때마다 오류에 부딪히고 헤맸습니다.
- 날짜를 쿼리로 전달하는 부분을 설계하는 데 가장 난항을 겪었습니다. JSON 데이터를 어떻게 탐색해야 하는지 아직 많은 공부가 필요하다고 느꼈습니다.
- 특히 한 번 날짜를 입력한 다음에 해당 정보를 저장시키고 싶었는데 이 과정에서 많은 시행착오를 겪기도 했지만 결과적으로는 useRef의 의미, form의 submit 이벤트, onChange 이벤트 등에 대해 많이 공부하는 기회가 되어 보람이 있었습니다.