css를 내장할 수 있도록 도와주는 도구

styled componets

css 모듈

- **css 파일의 확장자는 .module.css 로 지정**

```
.Box {
  background: black;
  color: white;
  padding: 2rem;
}
```

- 임포트

```
import styles from "./css/Box.module.css";
```

- 컴프넌트 사용

```
<div className={styles.Box}>class모듈</div>
```

https://styled-components.com/docs

styled components

- 스타일이 적용된 컴포넌트를 쉽게 만들 수 있고, 이미 존재하는 컴포넌트를 래핑해서 스타일이 적용된 새로운 컴포넌트로 생성

- 설치

```
>npm install styled-componets
```

- 스타일드 컴포넌트
  - 백틱안에 코드 추가

```
import styled from "styled-components";
```

```
const StyledButton = styled.button`
  color: white;
  background-color: purple;
`;
```

```
const LargeButton = styled(StyledButton)`
  font-size: 2rem;
`;
```

styled components

- 리액트 컴포넌트를 래핑하는 경우

```
const ReactButton = (props) => {
  return <button className={props.className}>{props.children}</button>;
};

const ReactLargeButton = styled(ReactButton)`
  font-size: 20px;
  color: tomato;
`;
```

```
<ReactLargeButton>리액트상속버튼</ReactLargeButton>
```

https://reactstrap.github.io/

reactstrap

- Stateless React Components for Bootstrap 5.

```
> npm install reactstrap
```

- css import

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

```
<head>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css" />
</head>
```

- 사용하기

```
import { Button } from "reactstrap";

<div> <Button color="primary" > Click Me </Button> </div>
```

호출 경로(URL)에 따라 컴포넌트를 매칭해 연결

router

SPA

- **Single Page Application**
- 한번만 **HTML** 페이지를 내려받고 **fetch**나 **ajax** 등의 함수로 서버와 주고받은 데이터를 이용해 일부 컴포넌트만 브라우저에서 **HTML**을 재구성한다**.**
- 리액트 라우터를 이용하여 **SPA** 개발
- 클라이언트 사이드 렌더링
- 단점
  - 앱의 규모가 커지면 파일이 너무 커짐 ➔ 코드 스플리팅 이용하여 라우트별로 파일을 나누어서 트래픽롸 로딩속도 개선
  - 자바스크립트를 실행하지 않는 일반 크롤러에서는 페이지의 정보를 제대로 수집이 안됨 ➔ 서버사이트 렌더링을 통해 해결

router

- 설치

```
> npm add react-router-dom
```

```
"react": "^18.3.1",
"react-dom": "^18.3.1",
"react-router-dom": "^6.23.1",
```

- **BroswerRouter 컴포넌트**
  - 리액트 라우터의 도움을 받고 싶은 컴포넌트의 최상위 컴포넌트를 감싸는 래퍼 컴포넌트
  - 프로젝트에 라우터를 사용할 수 있도록 적용

```
import { BrowserRouter } from "react-router-dom";
```

```
<BrowserRouter>
    <App />
</BrowserRouter>
```

router

- **Route 컴포넌트**
  - URL path에 맞는 컴포넌트가 렌더링 됨

```
<Route path="url주소정규식" element={컴포넌트}></Route>
```

```
<Routes>
        <Route path="/" element={<Home />}></Route>
        <Route path="/topics" element={<Topics />}></Route>
</Routes>
```

- **Link 컴포넌트**
  - 클릭하면 다른 주소로 이동시키는 컴포넌트. <a> 태그 대신 사용
  - HTML5 history API를 사용하여 브라우저의 주소만 바꿀뿐 페이지를 새로 불러오지는 않음

```
<Link to="/">Home</Link>
```

router

- **NavLink 컴포넌트**
  - 네비게이션에 사용자가 위치한 곳을 표시
  - class="active" 속성이 추가

```
import { Routes, Route, NavLink } from "react-router-dom";

function App() {
  return (
    <div className="App">
      <h1>Hello react router DOM</h1>
      <ul>
        <li><NavLink to="/">Home</NavLink></li>
        <li><NavLink to="/topics">Topics</NavLink></li>
        <li><NavLink to="/contact">Contact</NavLink></li>
      </ul>
      <Routes>
        <Route path="/" element={<Home />}></Route>
        <Route path="/topics" element={<Topics />}></Route>
        <Route path="/contact" element={<Contact />}></Route>
      </Routes>
    </div>
  );
}
```

https://reactrouter.com/en/main/hooks/use-params

URL parameter

- **useParams()**

```
var params = useParams();
var topic_id = params.topic_id;
```

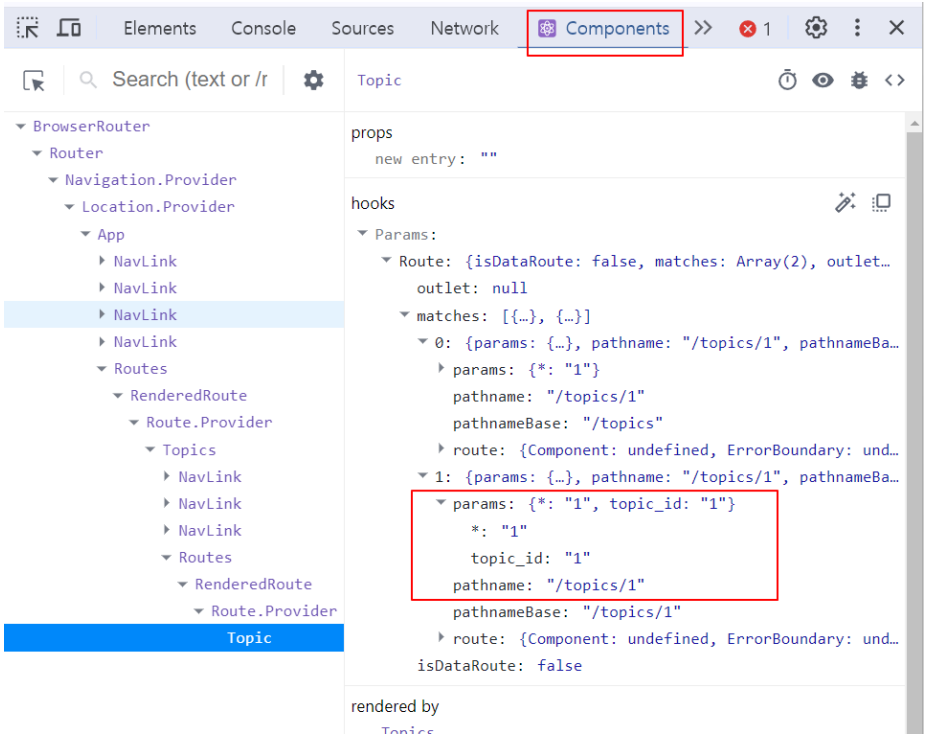
```
type UserProps = { id: string; };
const { id } = useParams <UserProps>();
```

- **Route**

```
<Route path="/topic/:topic_id"
       element={<Topic />}></Route>
```

- **URL**

```
http://localhost:3000/topic/1
```



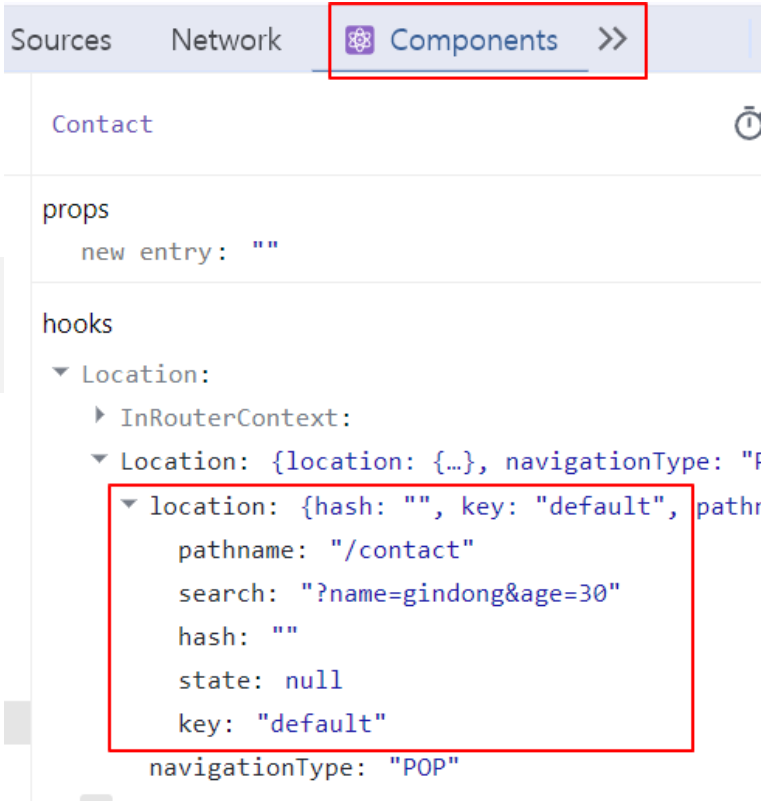
queryString

- **useLocation()**
  - location 객체에 있는 search 값에서 조회

```
const location = useLocation();
const search = new URLSearchParams(location.search);
var topic_id = search.get("name");
```

  - URL

```
http://localhost:3000/contact?name=gindong&age=30
```



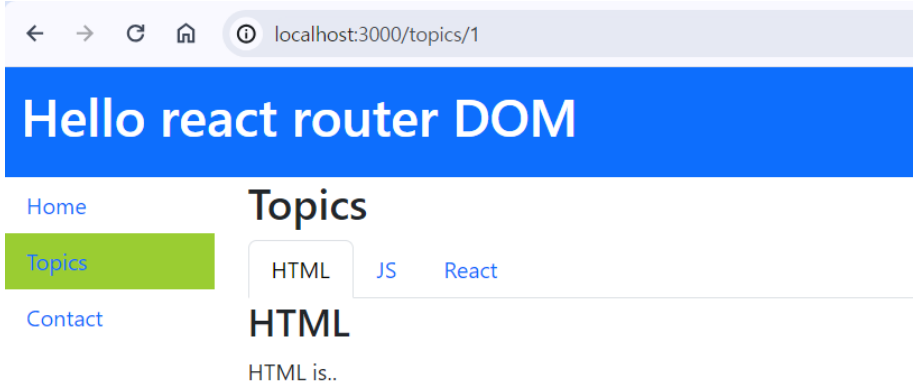
페이지 이동

- **useNavigate()**

```
const navigation = useNavigate();
```

```
  const goBack = () => {
    navigator(-1);                                  ← history.goBack(), go(-2)
  };
  const goHome = () => {
    navigator("/topics");                           ← history.push('/');
    navigation('/topics', {replace: true});         ← history.replace('Item/2');
  };
```

nested Routing



nested Routing, Outlet

```
render(
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<App />} >
        <Route path="expenses" element={<Expenses />} />
        <Route path="invoices" element={<Invoices />} />
      </Route>
    </Routes>
  </BrowserRouter>, rootElement)
```

```
import { Outlet, Link } from "react-router-dom";

export default function App() {
...
      </nav>
      <Outlet />
    </div>
  );
}
```

https://reactrouter.com/en/main/hooks/use-routes

useRoutes

```
function App() {
  const element = useRoutes([
    // Route에서 사용하는 props의 요소들과 동일
    { path: '/', element: <Home /> },
    { path: 'dashboard', element: <Dashboard /> },
    {
      path: 'invoices',
      element: <Invoices />,
      // 중첩 라우트의 경우도 Route에서와 같이 children이라는 property를 사용
      children: [
        { path: ':id', element: <Invoice /> },
        { path: 'sent', element: <SentInvoices /> },
      ],
    },
    // NotFound 페이지는 다음과 같이 구현할 수 있음
    { path: '*', element: <NotFound /> },
  ]);

  // element를 return함으로써 적절한 계층으로 구성된 element가 렌더링 될 수 있도록 함
  return element;
}
```