# React - React-Router-Dom

## 목표

React-Router-DOM을 이용해서 Single Page Application을 제작할 수 있어야 합니다.

Create & Read & Update & Delete 기능을 구현할 수 있어야 합니다.

## 강의시간

4시간

## 전체 소스코드

<https://github.com/egoingsb/react-component-tutorial.git>

## Single Page Application이란?

전통적인 멀티페이지 애플리케이션과 다르게 하나의 페이지 안에서 리로드 없이 UI가 변경되는 애플리케이션을 의미합니다. SPA라고 부릅니다.

### SPA의 핵심기술

1. DOM을 이용해서 웹페이지를 동적으로 변경하는 기술
2. UI가 바뀌면 URL을 변경하고, URL이 변경되면 UI가 변경되도록 하는 기술
3. Ajax

## History & Location

history.pushState을 이용해서 URL을 동적으로 변경하고, location.pathname을 이용해서 URL의 path를 알아낼 수 있습니다.

### 예제

| <  html>  <body>  <header>  <h1><a href="/" **onclick="**  **event.preventDefault();**  **history.pushState({},'','/');**  **document.querySelector('article').innerHTML = `<h2>Welcome</h2>Hello, WEB!`;**  **"**>WEB</a></h1>  </header>  <nav>  <ol>  <li><a href="/read/1" **onclick="**  **event.preventDefault();**  **history.pushState({}, '', '/read/1');**  **document.querySelector('article').innerHTML = `<h2>HTML</h2>HTML is HyperText Markup Language.`;**  **"**>html</a></li>  </ol>  </nav>  <article>  </article>  <script>  **if(location.pathname === '/'){**  **document.querySelector('article').innerHTML = `<h2>Welcome</h2>Hello, WEB!`;**  **} else if(location.pathname === '/read/1'){**  **document.querySelector('article').innerHTML = `<h2>HTML</h2>HTML is HyperText Markup Language.`;**  **}**  </script>  </body>  </html> |
| --- |

### 실행

어떤 경로로 접속해도 동일한 파일을 호스팅할 수 있어야 합니다. serve를 이용하면 쉽게 구현할 수 있습니다.

| npx serve -s DIRECTORY |
| --- |

## React-Router-DOM

React에서 URL에 따라서 UI가 달라지는 애플리케이션을 손쉽게 구현할 수 있도록 도와주는 라이브러리 입니다. RRD라고 부르겠습니다.

### 홈페이지

<https://reactrouter.com/>

### 설치

npm install react-router-dom@6

### 주요 API

#### BrowserRouter

RRD의 적용범위를 특정하는 컴포넌트입니다.

<https://github.com/egoingsb/react-component-tutorial/commit/b72cb4faa17134bc00f5251d3faced7009c66e28>

| **import {BrowserRouter} from 'react-router-dom';** ReactDOM.render(  <React.StrictMode>  **<BrowserRouter>**  <App />  **</BrowserRouter>**  </React.StrictMode>,  document.getElementById('root')  ); |
| --- |

#### Link

URL을 동적으로 변경하는 컴포넌트입니다.

<https://github.com/egoingsb/react-component-tutorial/commit/783119f20315515612ef1342ecff1acac4382273>

#### Routers, Route

path에 따라서 출력될 컴포넌트를 지정하는 컴포넌트입니다.

<https://github.com/egoingsb/react-component-tutorial/commit/02e9776b0028e0eeed467fe92103e5e670e0438d>

| **function Read(props){**  **return <Article title="Read" body="Hello, Read"></Article>**  **}** |
| --- |

| **<Routes>**  **<Route path="/" element={<Article title="Welcome" body="Hello, WEB"></Article>}></Route>**  **<Route path="/read/:id" element={<Read topics={topics}></Read>}></Route>**  **</Routes>** |
| --- |

#### useParams

URL 패턴과 일치하는 입력값을 추출해서 컴포넌트로 공급하는 hook

<https://github.com/egoingsb/react-component-tutorial/commit/8f7ec6c42089350c4d9ed8bece2b918656d96131>

| <Route path="/read/**:id**" element={<Read topics={topics}></Read>}></Route> |
| --- |

| function Read(props){  **let params = useParams();**  let title, body = '';  for(let i=0; i<props.topics.length; i++){  let t = props.topics[i];  if(t.id === Number(**params.id**)){  title = t.title;  body = t.body;  break;  }  }  return <Article title={title} body={body}></Article>  } |
| --- |

## 상태변경

state를 변경할 때 값이 원시 데이터 타입이 아닌 경우 값을 복제한 후에 변경해야 합니다.

원시 데이터 타입은 아래와 같습니다.

* Boolean 타입
* Null 타입
* Undefined 타입
* Number 타입
* BigInt 타입
* String 타입
* Symbol 타입

그 외에 객체(배열 포함)의 경우 복제 후 변경해야 합니다.

<https://github.com/egoingsb/react-component-tutorial/commit/22af0d91babfc9aaa010a3e3583e4beb2f41c7fc>

| let [topics, setTopics] = useState([  {id:1, title:'HTML', body:'HTML is ...'},  {id:2, title:'CSS', body:'CSS is ...'},  {id:3, title:'JavaScript', body:'JavaScript is ...'} ]); let [nextId, setNextId] = useState(4); |
| --- |

| let newTopics = [...topics]; newTopics.push({id:nextId, title:data.title, body:data.body}); setTopics(newTopics); setNextId(nextId+1); |
| --- |

위의 예제를 보면 topics는 복제한 후에 수정했고, nextId는 복제하지 않고 수정했습니다.

## 폼 다루기

<https://github.com/egoingsb/react-component-tutorial/commit/6caf5ea5163c9c3d79c2192325f78385af469016>

| function Update(props){  let params = useParams();  let \_title, \_body = '';  for(let i=0; i<props.topics.length; i++){  let t = props.topics[i];  if(t.id === Number(params.id)){  \_title = t.title;  \_body = t.body;  break;  }  }  let [title, setTitle] = useState(\_title);  let [body, setBody] = useState(\_body);  return <article>  <form onSubmit={event=>{  event.preventDefault();  let t = event.target.title.value;  let b = event.target.body.value;  let i = props.id;  props.onUpdate({id:params.id, title:t, body:b});  }}>  <p><input type="text" name="title" placeholder="Title" value={title} onChange={e=>{setTitle(e.target.value)}} /></p>  <p><textarea name="body" placeholder="Body" value={body} onChange={e=>{setBody(e.target.value)}}></textarea></p>  <p><input type="submit" value="Create" /></p>  </form>  </article> } |
| --- |

1. props를 통해서 전달한 값은 변경되지 않습니다. 아래 코드처럼 props를 state로 전환한 후에 값이 변경될 때마다 state의 값을 변경해주도록 처리해야 합니다.
2. submit 버튼을 눌렀을 때 onSubmit 이벤트가 호출됩니다.

## CRUD 구현

### create

<https://github.com/egoingsb/react-component-tutorial/commit/22af0d91babfc9aaa010a3e3583e4beb2f41c7fc>

### read

<https://github.com/egoingsb/react-component-tutorial/commit/8f7ec6c42089350c4d9ed8bece2b918656d96131>

### update

<https://github.com/egoingsb/react-component-tutorial/commit/6caf5ea5163c9c3d79c2192325f78385af469016>

### delete

<https://github.com/egoingsb/react-component-tutorial/commit/768d538a2dd14f44255d6b11c108f508d2f758f1>