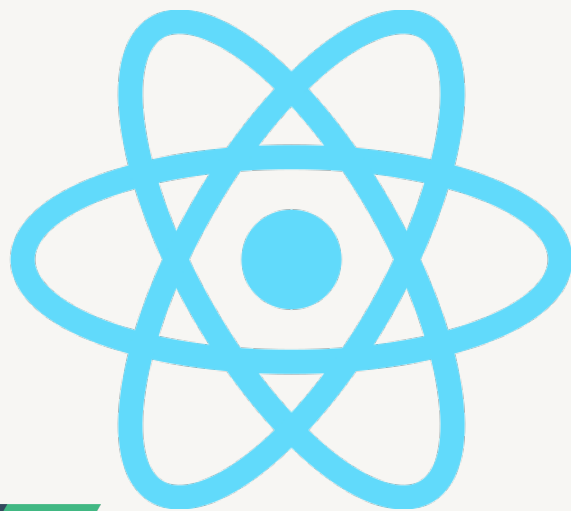
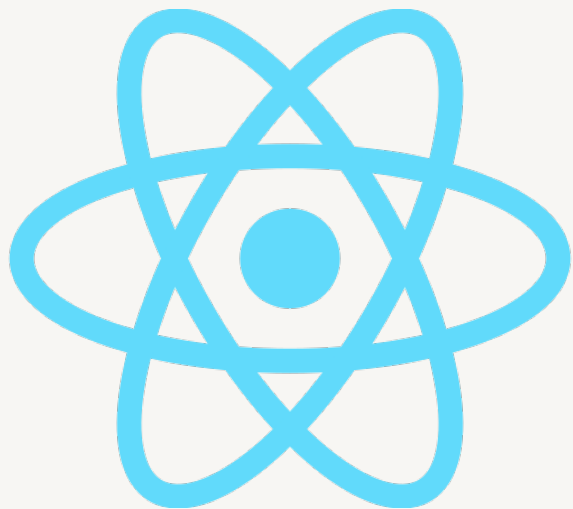


Abstract geometric lines in the top left corner, consisting of several overlapping, irregular polygons and lines in a light brown color.

WEB프로젝트

자바스크립트의 한계와 각종 프론트엔트 라이브러리, 프레임워크의 등장





장점

1. 거대한 React 커뮤니티
2. JS 라이브러리의 호환
3. 구성 요소 재사용 용이

단점

1. 높은 학습 난이도
2. JSX
3. 환경 구성



장점

1. 쉬운 학습 난이도
2. 가벼운 크기
3. 유연성

단점

1. 제한된 기능
2. 커뮤니티 규모가 작음

REACT란

React.JS는 프레임워크가 아닌 라이브러리

•**프레임워크** : 애플리케이션 구축 시 모든 애플리케이션의 공통적인 부분을 제공해줌.
필요한 기능이 이미 만들어져 있어 만들어진 '틀' 안에 '내용물'을 채워넣음으로써 완성시킴.
뼈대에 해당. 미리 만들어진 틀 밖으로 벗어나기가 어려움.
(Angular JS, Django, Vue JS 등)

•**라이브러리** : 필요한 부분만을 단독으로 가지고 와서 사용하는 것이 가능.
기능을 하게 하는 부품에 해당. 가벼움.
React.JS는 유저 인터페이스를 만들기 위한 라이브러리

REACT특징

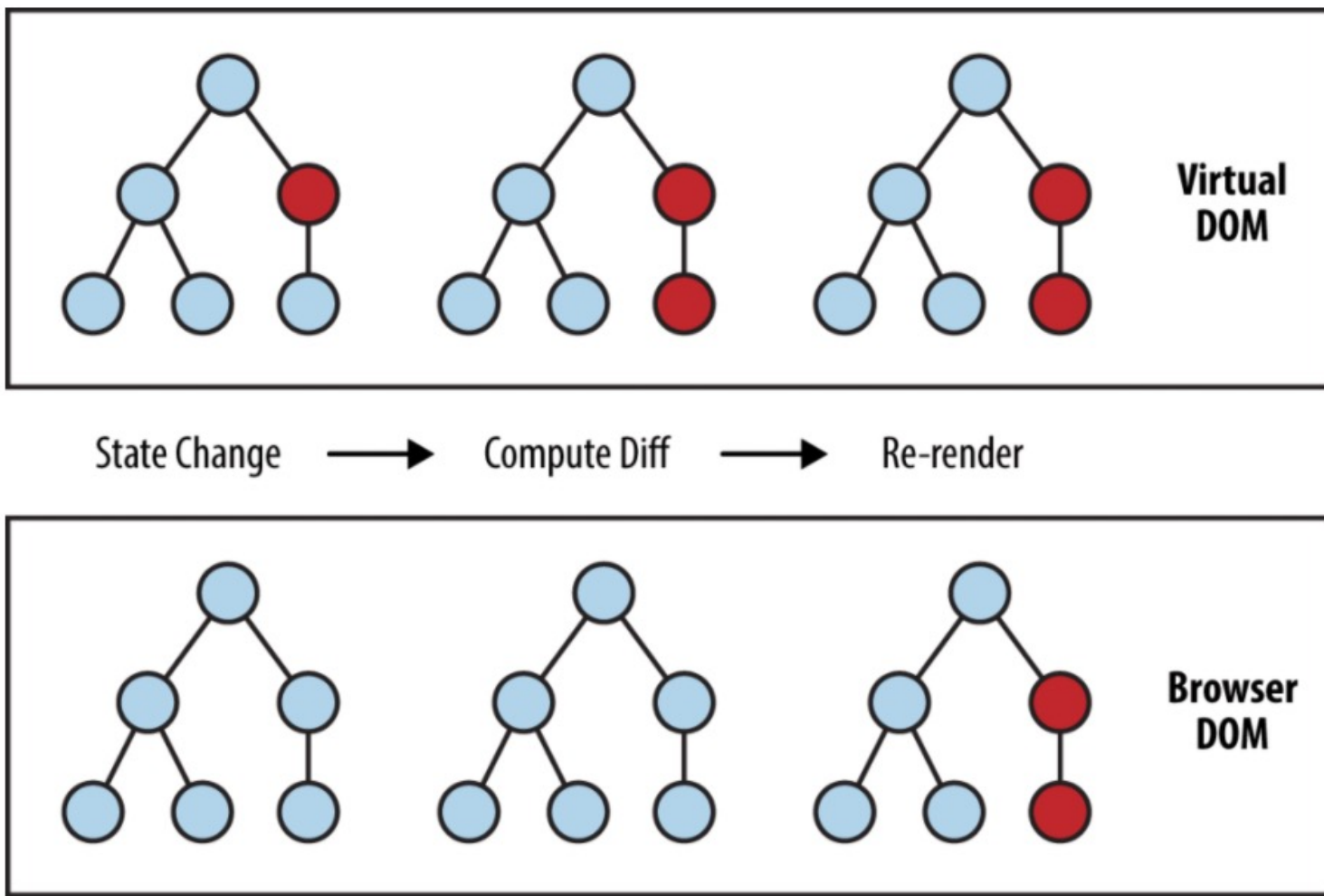
1. 가상 DOM
2. 컴포넌트 기반
3. JSX 문법
4. Props, State

1. 가상 DOM

기존의 웹페이지를 렌더링 하는 방식은 HTML파일을 해석하고 이에 따른 CSS파일도 해석. DOM트리를 생성하고 이를 최종적으로 브라우저 Viewport에 렌더링 하는 과정을 거친다.

여기서 HTML, CSS 파일에 조금이라도 수정이 발생하면 모든 내용을 다시 렌더링 해야 된다는 문제가 발생한다.

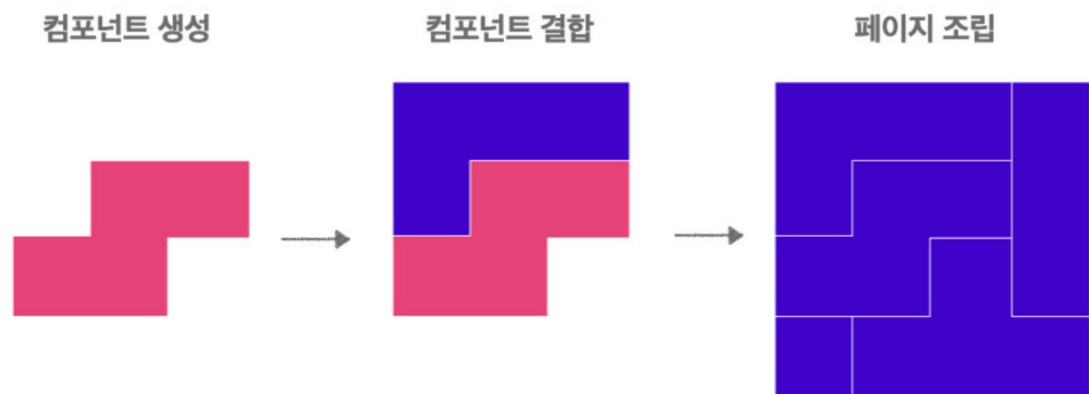
이를 방지하기 위해 리엑트는 가상 DOM을 도입, 내용이 수정된 가상DOM과 원본DOM을 비교하여, 내용의 차이가 있는 부분만을 재렌더링하여 불필요한 렌더링을 줄이는 방식이다.



2. 컴포넌트 기반

리엑트는 컴포넌트 기반으로 페이지를 개발할 수 있다.
페이지를 하나의 html 문서로 작성하지 않고,
여러 개의 컴포넌트로 나눠서 작성, 관리를 하게 되면
큰 프로젝트를 진행할 때, 문서의 가독성, 생산성 면에서
큰 이점을 가지게 된다.

Component-Driven Development(CDD)



3. JSX 문법

JSX는 자바스크립트의 확장 문법이다.

JSX로 작성된 코드는 브라우저에서 실행되기 전에 일반 자바스크립트로 변환되며,

일반 자바스크립트 보다 훨씬 쉽고 간단하게 코드를 작성할 수 있고 HTML 태그를 쓰듯이 코드 작성을 할 수 있어 편리하다는 장점이 있다.



4. props, state

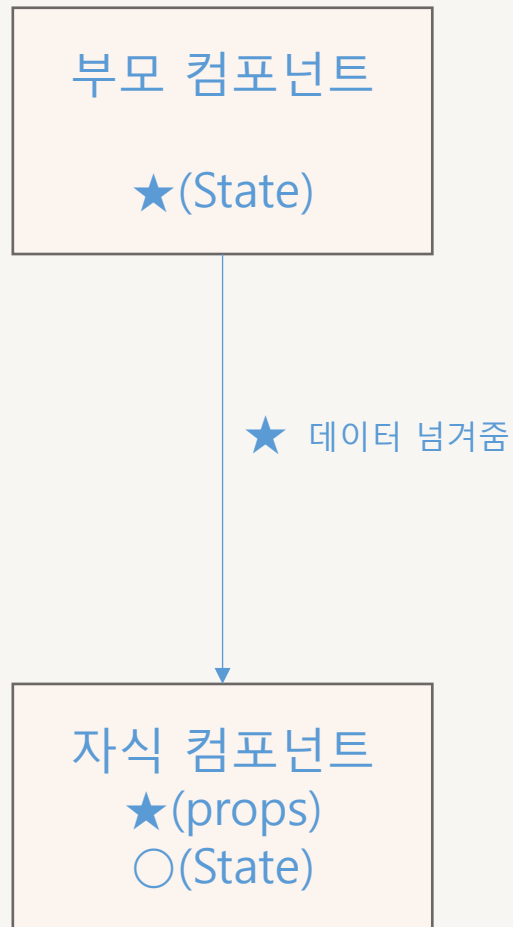
기존의 웹페이지를 렌더링 하는 방식은
props와 state는 리액트에서 데이터를 다룰 때 반드시 필요한 필수적인 요소들이다.

props

리액트에서 props는 변하지 않는 데이터 값으로,
상위(부모) 컴포넌트에서 하위(자식) 컴포넌트로 데이터를 넘겨줄 때 사용한다.

state

state는 props와 반대로 유동적으로 변하는 데이터를 다룰 때 사용한다. 시간에 따라, 혹은 사용자의 입력에 따라 값이 변하는 데이터를 다룰 때 state를 사용하게 된다.



리액트를 사용하지 않고 웹페이지를 구현한 경우.

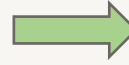
하나의 html파일에 모든 내용을 작성해야된다.

```
<header>
  <h1>Logo</h1>
</header>
<nav>
  <ul>
    <li>
      <a href="#">메뉴1</a>
    </li>
    <li>
      <a href="#">메뉴2</a>
    </li>
  </ul>
</nav>
<section>
  <p>Hello World!</p>
</section>
```

```
const Hlogo = () => {
  return (
    <header>
      <h1>Logo</h1>
    </header>
  );
};
```

```
const Anav = () => {
  return (
    <nav>
      <ul>
        <li>
          <a href="#">메뉴1</a>
        </li>
        <li>
          <a href="#">메뉴2</a>
        </li>
      </ul>
    </nav>
  );
};
```

```
const Bsection = ({ amumal }) => {
  return (
    <section>
      <p>{amumal}</p>
    </section>
  );
};
```



```
const App = () => {
  return (
    <div className="App">
      <Hlogo />
      <Anav />
      <Bsection amumal="Hello World!" />
    </div>
  );
};

export default App;
```

리액트를 사용한 경우.

페이지의 내용을 각각의 컴포넌트(부품)으로 쪼개서
관리, 사용이 가능하다.

리액트 사용 시 장점

웹 페이지 제작에 도움을 주는 프레임워크.

웹 페이지에 주로 사용되는 레이아웃이나 모달, 네이게이션, 버튼, 목록 등의 요소를 디자인이 완성된 형태로 바로 가져와서 사용할 수 있다.

PC뿐 아니라 태블릿, 스마트폰과 같은 디자인도 반응형으로 지원하기 때문에 직접 CSS 코드를 작성하여 수정하는 것에 비해 시간 소요가 굉장히 줄어든다.

<https://getbootstrap.kr/>



개발 환경 구축

Node.js

Node.js는 자바스크립트를 브라우저 외의 환경에서도 코드를 실행할 수 있게 하는 런타임 환경이다.

이를 통해 자바스크립트로 서버를 구축하는데 주로 사용 되지만,

서버에 관한 내용은 다루지 않고, 노드에서 제공하는 패키지 매니저를 통해

리액트와 리액트 개발에 필요한 모듈을 다운 받기 위해서 설치를 한다.

<https://nodejs.org/ko>



Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

New security releases to be made available August 8th, 2023

다운로드 - Windows (x64)

18.17.0 LTS

안정적, 신뢰도 높음

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

20.5.0 현재 버전

최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

LTS 일정은 [여기서](#) 확인하세요.

LTS 버전 다운로드 후, 설치

REACT

리액트 프로젝트를 만들고 작업할 폴더를 생성한다.

vscode에서 해당 폴더를 열고,

터미널에 `npx create-react-app {프로젝트 이름}` 입력

성공적으로 설치 시 우측 하단과 같은 파일들이 추가됨.

```
문제  출력  디버그 콘솔  터미널

PS C:\Users\yubeen\webproject\react> npx create-react-app react-app

Creating a new React app in C:\Users\yubeen\webproject\react\react-app.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...

added 1427 packages in 1m

231 packages are looking for funding
  run `npm fund` for details

Installing template dependencies using npm...
```

▼ react-app

> node_modules

> public

> src

📄 .gitignore

📄 package-lock.json

📄 package.json

📄 README.md

생성 된 리액트 폴더 디렉토리로 이동 후,
npm start를 입력한다.

localhost:3000 페이지가 열리면서

리액트의 샘플 어플리케이션 화면이 나온다면

기본 설치 및 실행까지 완료.

```
● PS C:\Users\yubeen\webproject\react> cd react-app  
PS C:\Users\yubeen\webproject\react\react-app> npm start
```



Edit src/App.js and save to reload.

[Learn React](#)

REACT의 구조

리액트 프로젝트에서 npm start를 입력하면

index.js 파일을 찾아서 실행하게 된다.

index.js는 <App> 이라는 컴포너트를 렌더링 하는

역할을 한다.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
        <a
          className="App-link"
          href="https://reactjs.org"
          target="_blank"
          rel="noopener noreferrer"
        >
          Learn React
        </a>
      </header>
    </div>
  );
}

export default App;
```



```
import logo from './logo.svg';
import './App.css';

function App() {
  return (
    <div className="App">
      | 안녕하세요!
    </div>
  );
}

export default App;
```

안녕하세요!

따라서 index.js에서 불러온 <App> 컴포넌트의 원본 파일인 App.js의 컴포넌트 부분을 수정하면

리액트 앱에 바로 반영이 되는 것을 볼 수 있다.

```

.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

```



```

body {
  background-color: green;
}

```

안녕하세요!

또한 각 컴포넌트와 연결된 css 파일 또한 수정을 통해 스타일 변경이 가능하다.

컴포넌트가 렌더링 되는 구역을 지정할 수 있을까?

```
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);
```

const root = id가 'root'인 요소를 찾는다.

root.render = 해당 요소에 안에 작성된 컴포넌트를 렌더링한다.

(root의 위치는?)

리액트 앱의 publi폴더 > index.html 파일에서 확인이 가능하다.

기본 생성 된 파일에 root id를 가진 div태그가 있는 모습.

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
</body>
</html>
```

REACT앱 배포

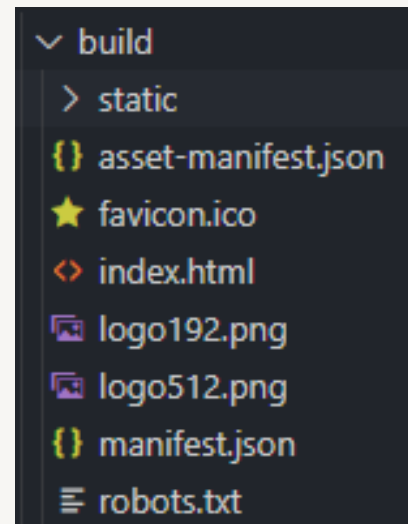
npm start로 실행한 앱은 웹페이지를 개발할 때는 유용하지만

실행 시, 불필요한 메시지가 나타나는 등 실제 배포하기에는 적합하지 않다.

배포를 위해서는 터미널에 `npm run build`를 입력한다.

입력 시 작성 한 내용을 토대로 빌드가 진행되며,

public 폴더와 유사한 구조의 build 폴더가 생성된다.



추가적으로 `npx serve -s build` 명령어를 입력하여
빌드한 앱을 배포한다.

그렇게 생성된 링크로 접속하면 빌드한 버전의 앱을 볼 수 있다.

```
PS C:\Users\yubeen\webproject\react\react-app> npx serve -s build
Need to install the following packages:
  serve@14.2.0
Ok to proceed? (y) y
```

Serving!

- Local: http://localhost:3000
- Network: http://121.167.104.162:3000

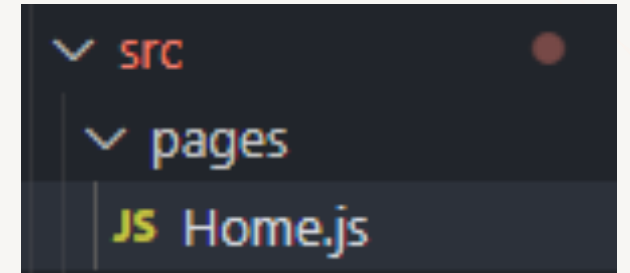
Copied local address to clipboard!

안녕하세요!

컴포넌트 추가

새로운 컴포넌트를 생성하여 App 컴포넌트에 추가해보자.

pages폴더 생성 > Home.js 생성 (리액트 컴포넌트는 반드시 첫 글자가 대문자)



```
import React from "react";

function Home(){
  return <h1>홈 화면 입니다.</h1>;
}

export default Home;
```

Home.js에 다음과 같은 함수형 컴포넌트를 작성하고,
export 해서 다른 컴포넌트에서 Home 컴포넌트를 가져올 수 있도록 한다.

```
import './App.css';  
import Home from './pages/Home'  
  
function App() {  
  return (  
    <div className="App">  
      <Home />  
    </div>  
  );  
}  
  
export default App;
```

App.js에서 Home 컴포넌트를 import 하여 가져온다.

그 후 App 컴포넌트 안에 Home 컴포넌트를 추가하고

npm start로 리액트 앱을 실행시키면 화면이 갱신된다.

홈 화면 입니다.

라우팅과 네비게이션

사용자의 반응, 이벤트 발생 등 상황에 따라 보여줄 컴포넌트를 지정할 수 있도록 라우팅 기능을 추가하고 이를 통해 네비게이션 메뉴 기능을 구현한다.

```
PS C:\Users\yubeen\webproject\react\react-app> npm install react-router-dom@6
added 3 packages, and audited 1504 packages in 4s

240 packages are looking for funding
  run `npm fund` for details

6 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force
```

npm install react-router-dom@6로 리액트-라우터-돔을 설치

package.json 파일에 react-router-dom 항목이 추가 되었다면 설치 완료.

```
"dependencies": {
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "react": "^18.2.0",
  "react-dom": "^18.2.0",
  "react-router-dom": "^6.14.2",
  "react-scripts": "5.0.1",
  "web-vitals": "^2.1.4"
},
```

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import {BrowserRouter} from 'react-router-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

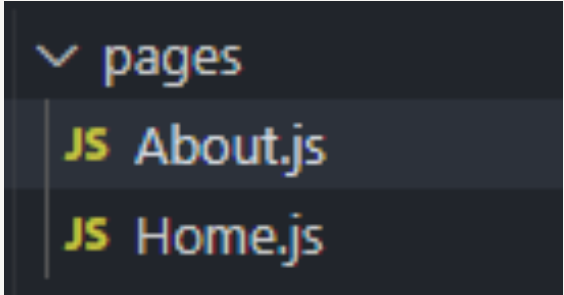
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

index.js에 react-router-dom의 BrowserRouter를 impor한다.

render안의 <App>컴포넌트를 <BrowserRouter>로 감싸준다.

메뉴를 클릭했을 때, 렌더링 되는 컴포넌트가 바뀌는 것을 확인하기 위해
About 이라는 추가 컴포넌트를 작성한다.
(추가 컴포넌트의 내용은 스스로 작성해본다.)



```
▼ pages
  JS About.js
  JS Home.js
```

```
import React from "react";
import {Routes, Route, Link} from "react-router-dom";
import './App.css';
import Home from './pages/Home';
import About from './pages/About';

function App() {
  return (
    <div className="App">
      <nav>
        <Link to="/">Home</Link>|
        <Link to="/about">About</Link>
      </nav>
      <Routes>
        <Route path="/" element={<Home />}></Route>
        <Route path="/about" element={<About />}></Route>
      </Routes>
    </div>
  );
}

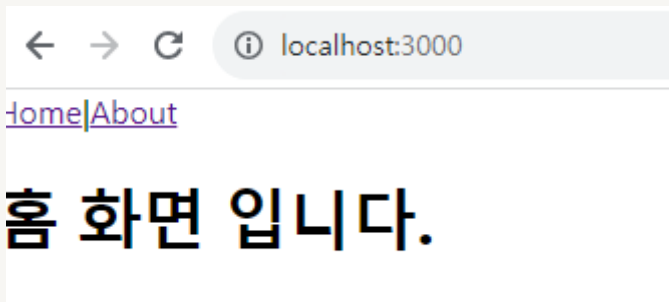
export default App;
```

App 컴포넌트를 수정하여 라우팅 기능을 추가한다.

react-router-dom에서 import한 Link태그를 사용하여
메뉴 클릭 시, 해당 링크로 이동하는 기능을 추가한다
(About클릭 시, localhost:3000/about으로 이동)

Routes > Route는

현재 페이지의 주소(path)가 /(기본) 혹은 /about일 때,
element 안에 해당하는 컴포넌트 내용을 출력하는 기능을 한다.



Home 컴포넌트와 About 컴포넌트가 라우팅을 통해 렌더링 된다.

STATE와 카운터

리액트에서 state를 사용하여 숫자가 1씩 증가, 감소하는 카운터를 간단하게 구현할 수 있다.

pages폴더 > Counter.js 컴포넌트를 생성하여 다음과 같이 코드를 작성.

(추가된 Counter 컴포넌트는 네비게이션 메뉴 항목 추가, 라우팅을 통해 렌더링 될 수 있도록 App.js 코드를 수정해본다.)



적용이 된 경우, +1 버튼을 누르면 숫자가 증가한다

```
import React from "react";
import {useState} from "react";

const Counter = () => {
  const [number, setNumber] = useState(0);

  const increase = () => {
    setNumber(number + 1);
  }

  return (
    <div>
      <button onClick={increase}>+1</button>
      <button>-1</button>
      <p>{number}</p>
    </div>
  )
}

export default Counter;
```

USESTATE

state 값을 관리하기 위해서는 useState 함수를 사용한다.

```
const [number, setNumber] = useState(0);
```

useState 함수를 사용하기 위해서는 위 처럼 기본 값(0)을 파라미터로 넣어서 호출해준다.

이렇게 useState를 호출하면 배열이 반환되는데
첫번째 원소는 현재 상태,
두번째 원소는 상태를 변경하는 함수다.(Setter함수)

+1 버튼의 onClick이벤트가 발생했을 때 실행되는 increase 함수에 setNumber를 통해 현재 상태(number)를 변경할 수 있게 된다.

number 상태는 <p>태그 안에 렌더링 된다.

(※ 감소하는 decrease 함수도 구현하기)

```
const Counter = () => {  
  const [number, setNumber] = useState(0);  
  
  const increase = () => {  
    setNumber(number + 1);  
  }  
}
```

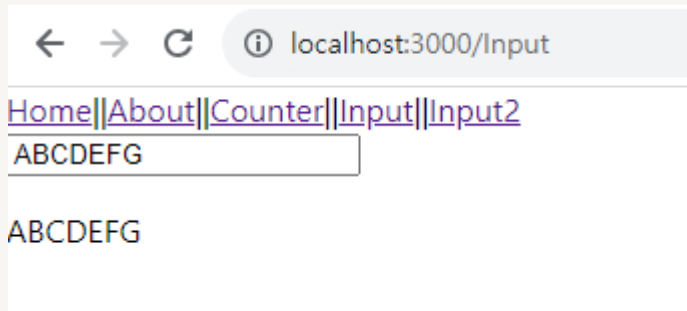
```
<button>-1</button>  
<p>{number}</p>  
</div>
```

다양한 입력 형식

useState 함수를 이용해서, 숫자뿐만 아니라 문자열을 입력 받는 경우 또한 상태 관리가 가능하다.

pages/Input.js 파일을 생성하여 다음과 같이 작성.

onChange의 (e) = element = input 태그 요소



```
import React, {useState} from "react";

const Input = () => {
  const [textValue, setTextValue] = useState("");

  const onChange = (e) => {
    setTextValue(e.target.value);
  };

  return (
    <div>
      <input type="text" value={textValue} onChange={onChange}></input>
      <br/>
      <p>{textValue}</p>
    </div>
  )
}

export default Input;
```


pages/Input2.js 생성

이번엔 하나의 컴포넌트에서 여러 개의 입력을 받을 때, 여러 개의 state를 관리하는 방법을 알아본다.

```
import React, {useState} from "react";

const Input2 = () => {
  const [inputs, setInputs] = useState({
    name: "",
    email: "",
    tel: ""
  });

  const {name, email, tel} = inputs;

  const onChange = (e) => {
    const value = e.target.value;
    const id = e.target.id;

    // inputs[id] = value;
    setInputs({
      ...inputs, //spread
      [id]:value
    })
  };
};
```

```
return (
  <div>
    <div>
      <label>이름</label>
      <input type="text" id="name" value={name} onChange={onChange}/>
    </div>
    <div>
      <label>이메일</label>
      <input type="text" id="email" value={email} onChange={onChange}/>
    </div>
    <div>
      <label>전화번호</label>
      <input type="text" id="tel" value={tel} onChange={onChange}/>
    </div>
    <p>{name}</p>
    <p>{email}</p>
    <p>{tel}</p>
  </div>
)
}

export default Input2;
```