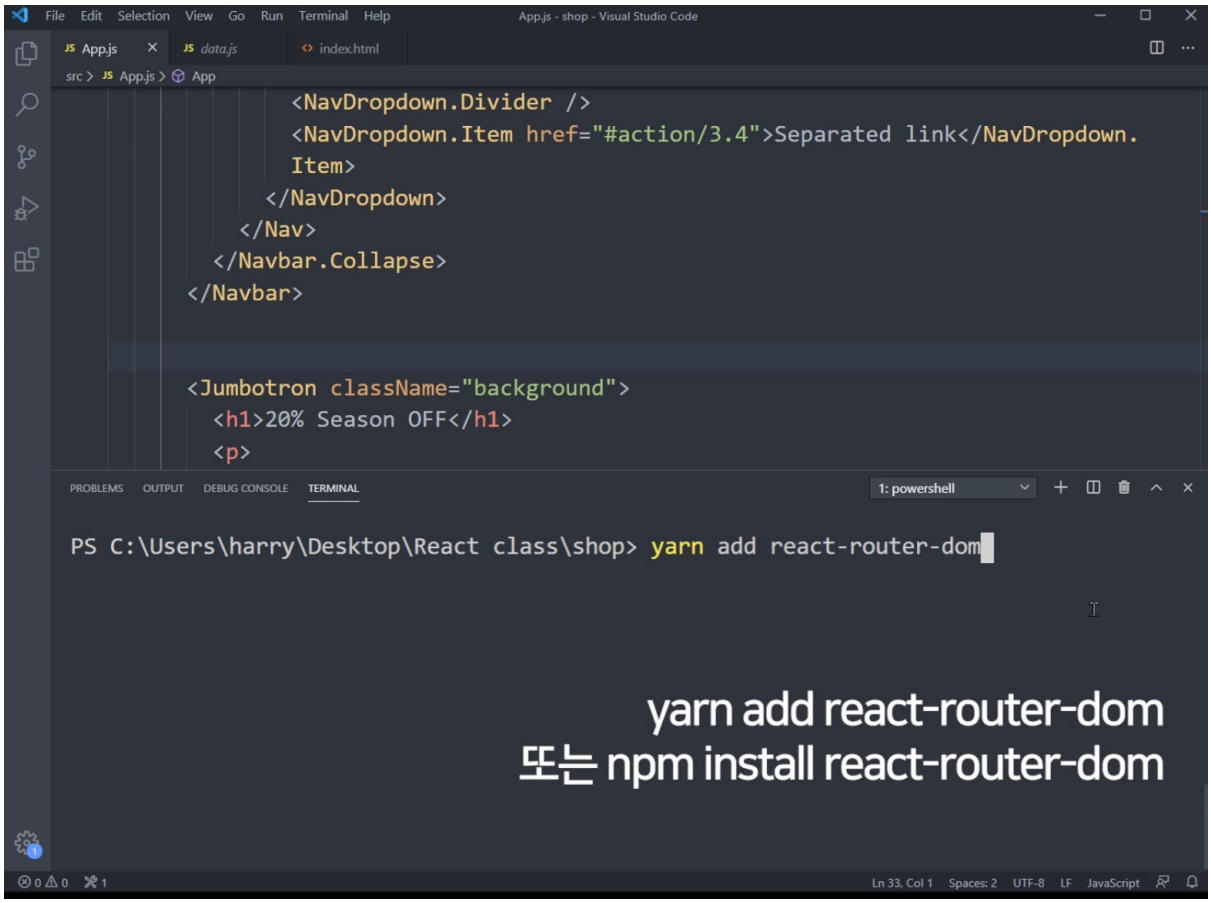
9월 6일 (월)

1. 코딩애플 - 리액트 강의 - Part 2. 쇼핑몰 프로젝트

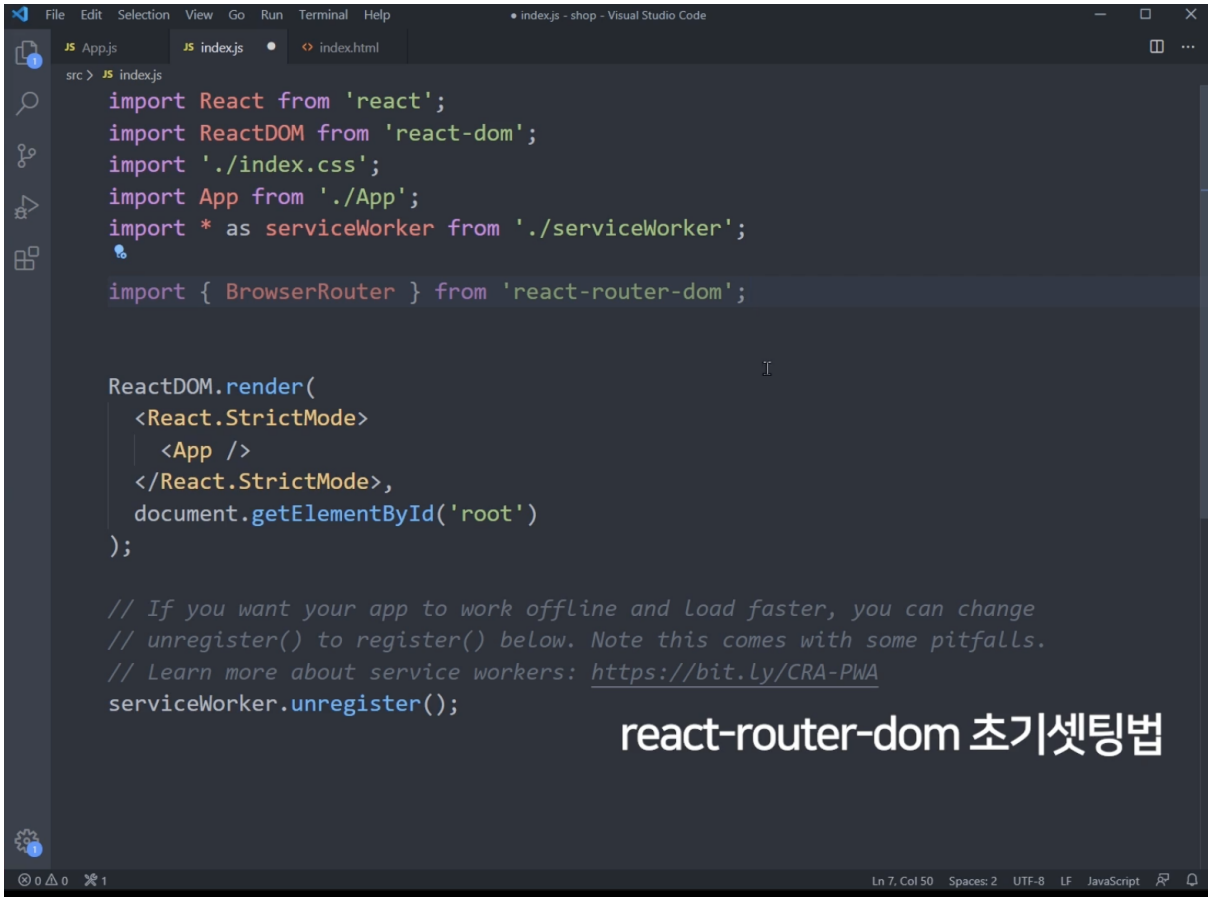
5) React Router 1 : 셋팅과 기본 라우팅

라우팅은 페이지 나누기와 같다.

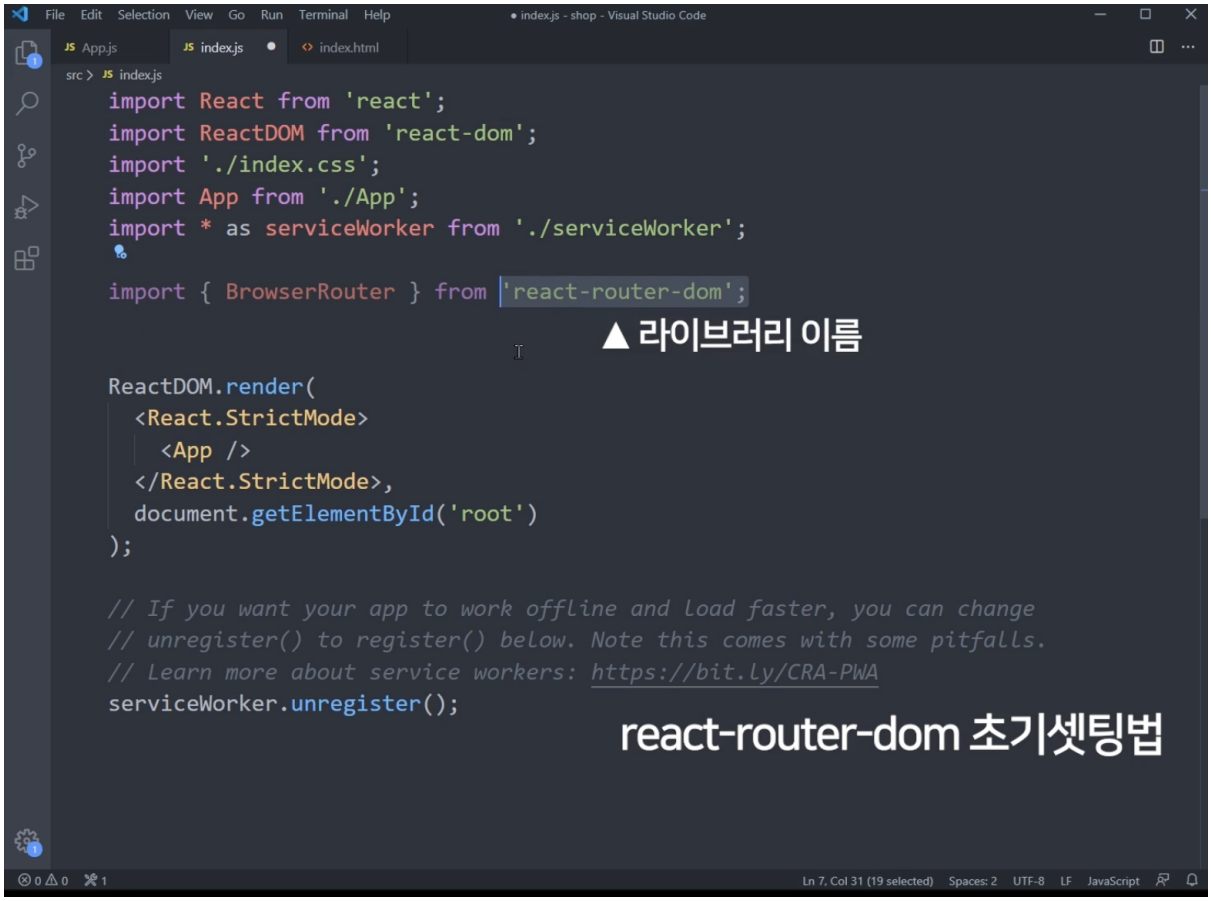
이는 react-router-dom 라이브러리를 이용해야한다.



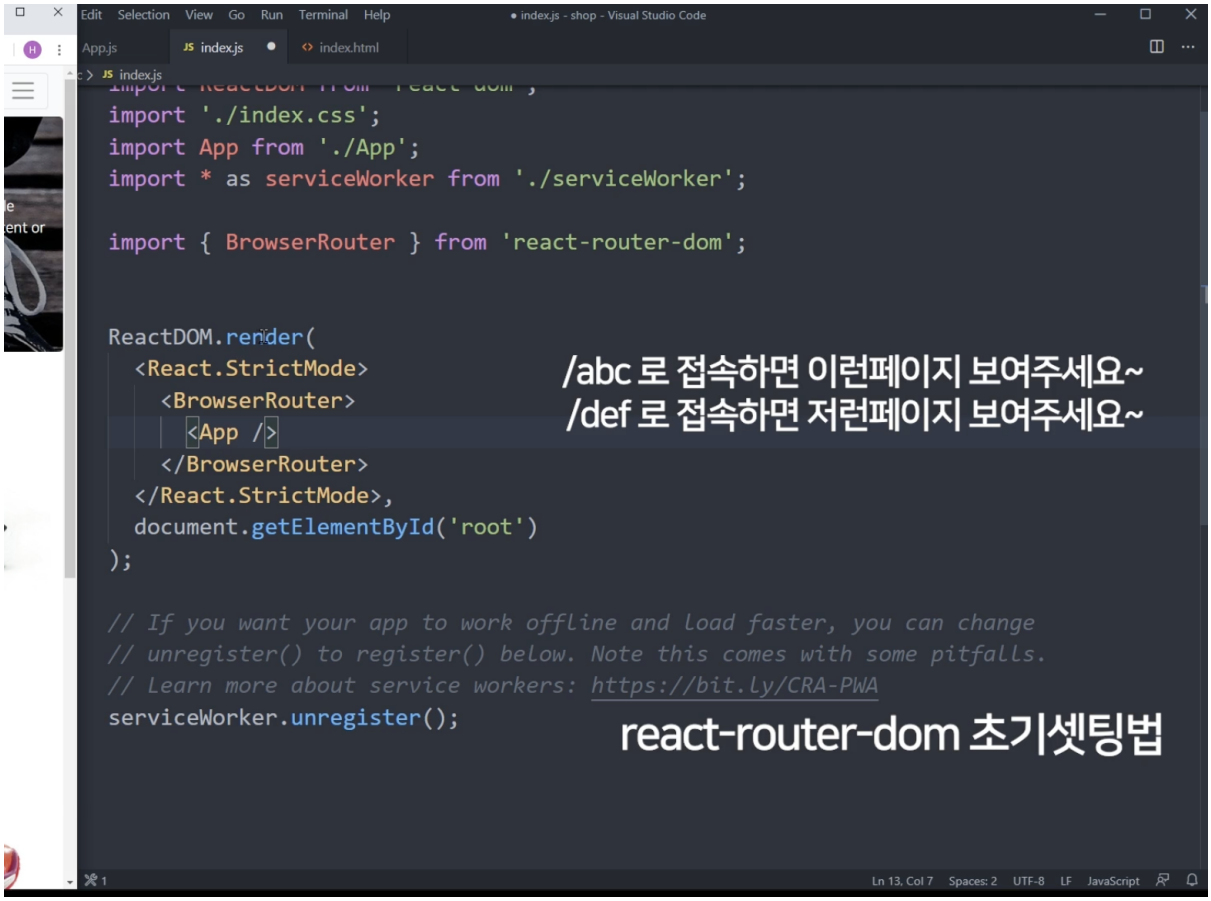
라이브러리를 설치해준다.



index.js 에 import { BrowserRouter } 을 넣어준다.

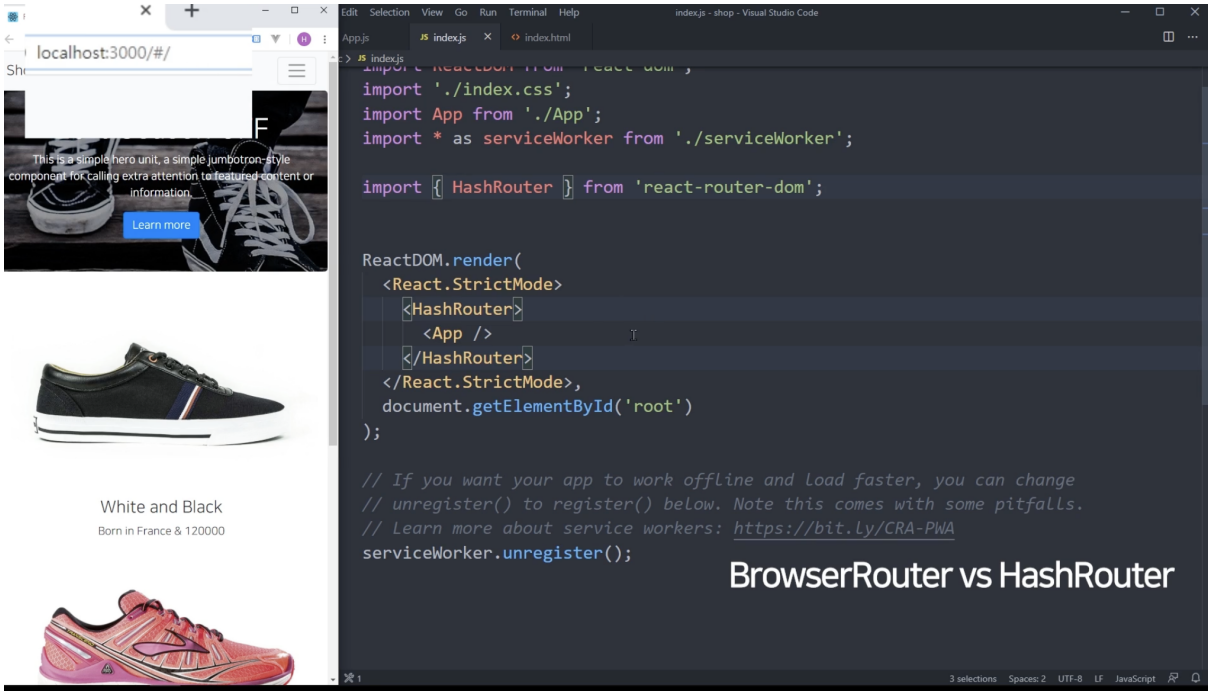


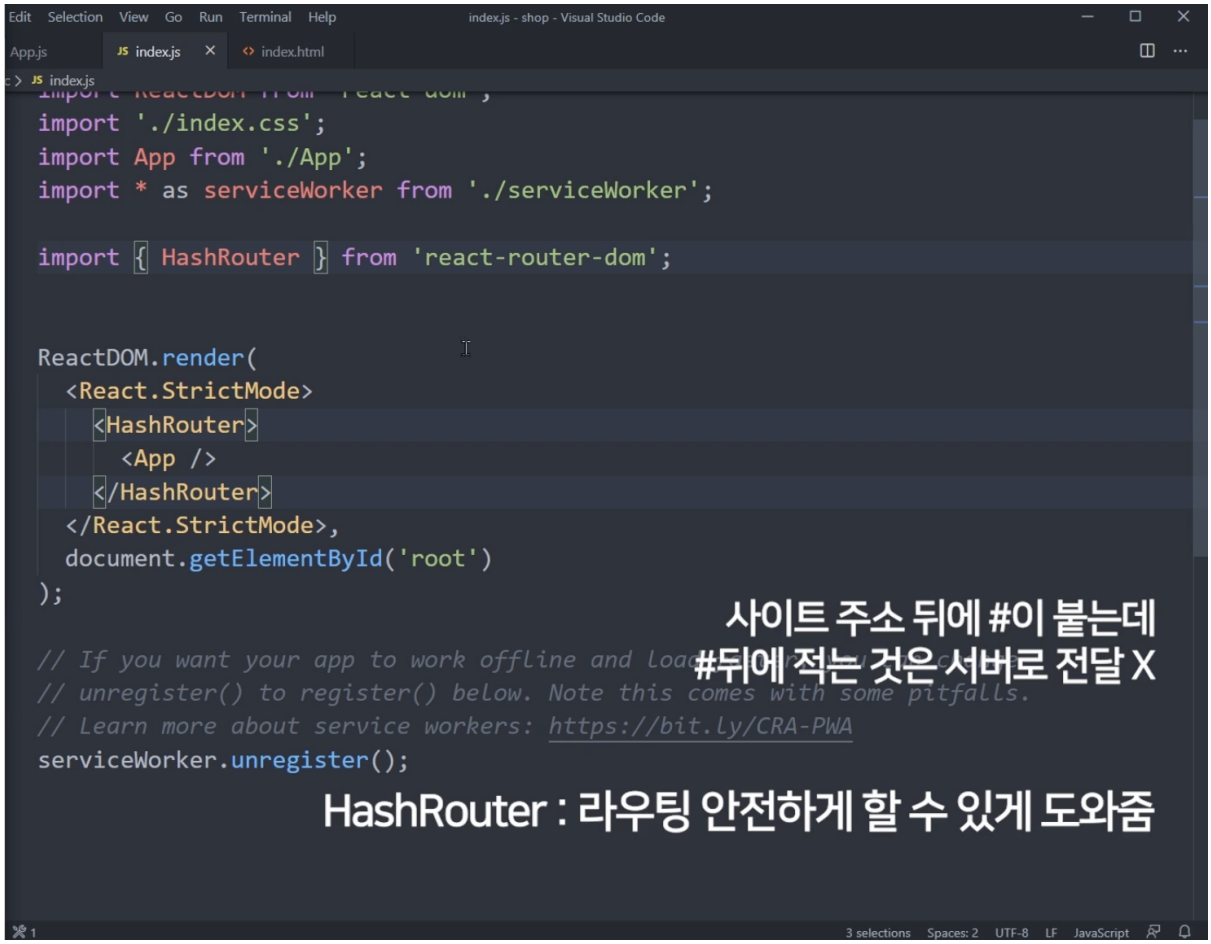
보통 import 에 " ./ " 이 없는 건 라이브러리 라고 보면 된다.



<BrowserRouter> 태그 안에 <App> 태그를 감싸면 셋팅끝이다. 이제 페이지를 여러개를 나눌 수 있다.

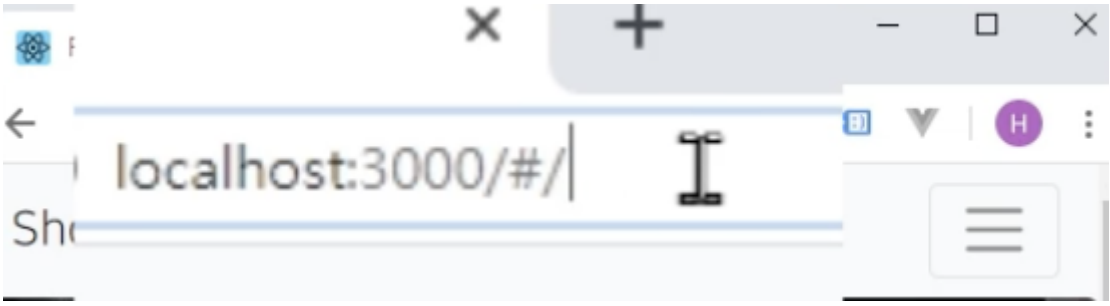
- HashRouter VS BrowserRouter


localhost:3000/#/
This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.
Learn more
White and Black
Born in France & 120000
index.js - shop - Visual Studio Code
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
import { HashRouter } from 'react-router-dom';
ReactDOM.render(
<React.StrictMode>
<HashRouter>
<App />
</HashRouter>
</React.StrictMode>,
document.getElementById('root')
);
// If you want your app to work offline and load faster, you can change
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
BrowserRouter vs HashRouter


index.js - shop - Visual Studio Code
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
import { HashRouter } from 'react-router-dom';
ReactDOM.render(
<React.StrictMode>
<HashRouter>
<App />
</HashRouter>
</React.StrictMode>,
document.getElementById('root')
);
// unregister() to register() below. Note this comes with some pitfalls.
// Learn more about service workers: https://bit.ly/CRA-PWA
serviceWorker.unregister();
사이트 주소 뒤에 #이 붙는데
#뒤에 적는 것은 서버로 전달 X
HashRouter : 라우팅 안전하게 할 수 있게 도와줌

해쉬라우터는 url 부분에 # 기호가 등장하고 브라우저라우터는 # 기호가 없다.

해쉬라우터는 라우팅을 좀 안전하게 할 수 있게 도와준다. 주소창은 서버에게 요청을 하는 공간으로 해쉬라우터는 서버에게 요청하지 않고 리액트에서 처리해주는 것이다.

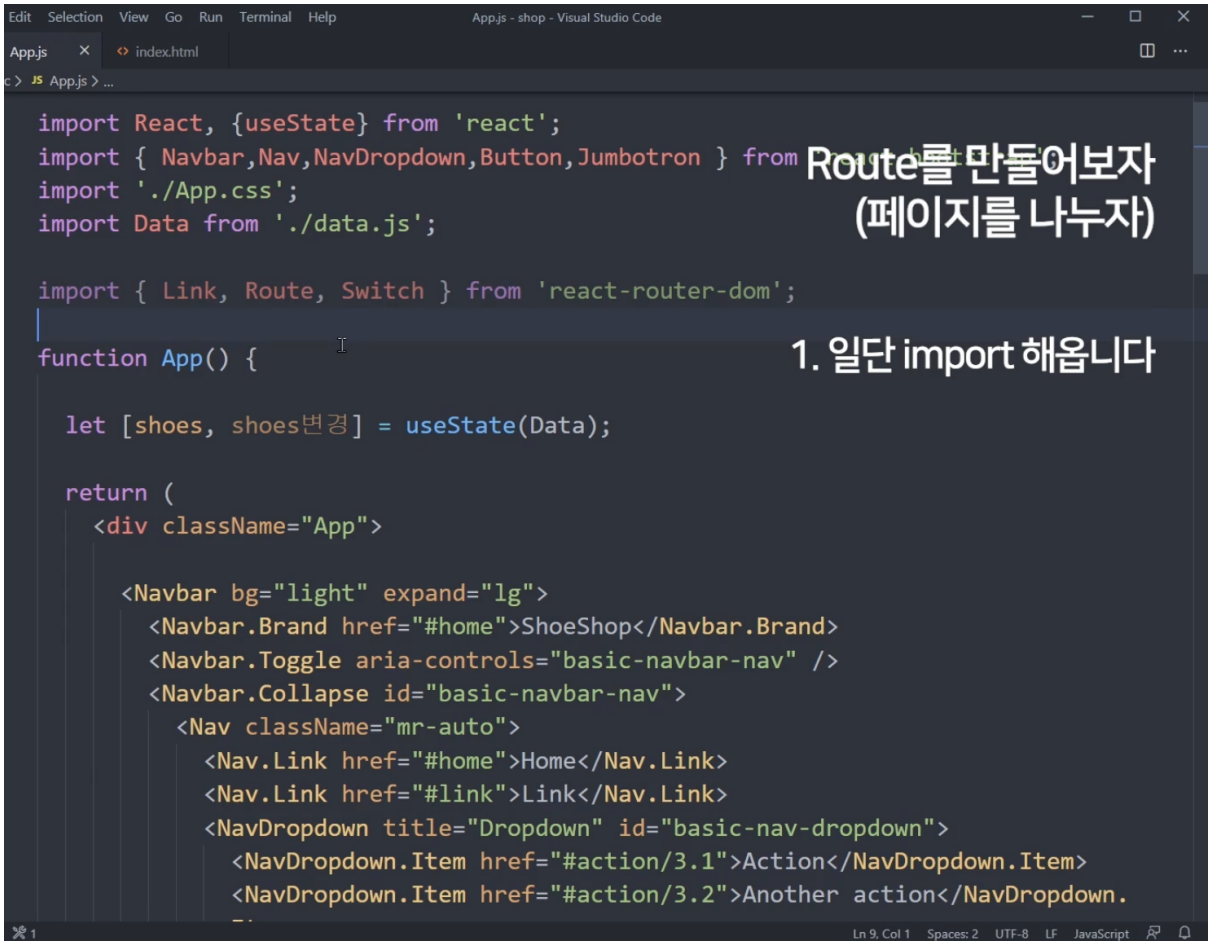


이렇게 #/ 이후에 다른 경로가 오게 된다. # 기호 뒤에 적는 것은 서버로 전달하지 않는다. 그래서 라우팅은 알아서 잘 해줄 수 있다.

브라우저라우터는 라우팅을 리액트가 아니라 서버에게 요청할 수도 있어서 위험하다. 즉, 서버에서 "그런 페이지 없는데요?" 라고 할 수 있다. 서버에서 서버 라우팅을 방지하는 API를 작성해둬야한다.

- Route 를 만들어보자. (페이지를 나누자)

⇒ 메인페이지 와 상세페이지를 나눠보자.



일단 Link Route, Switch 를 import 해온다.


Edit Selection View Go Run Terminal Help
• App.js - shop - Visual Studio Code
App.js
index.html
App.js > App

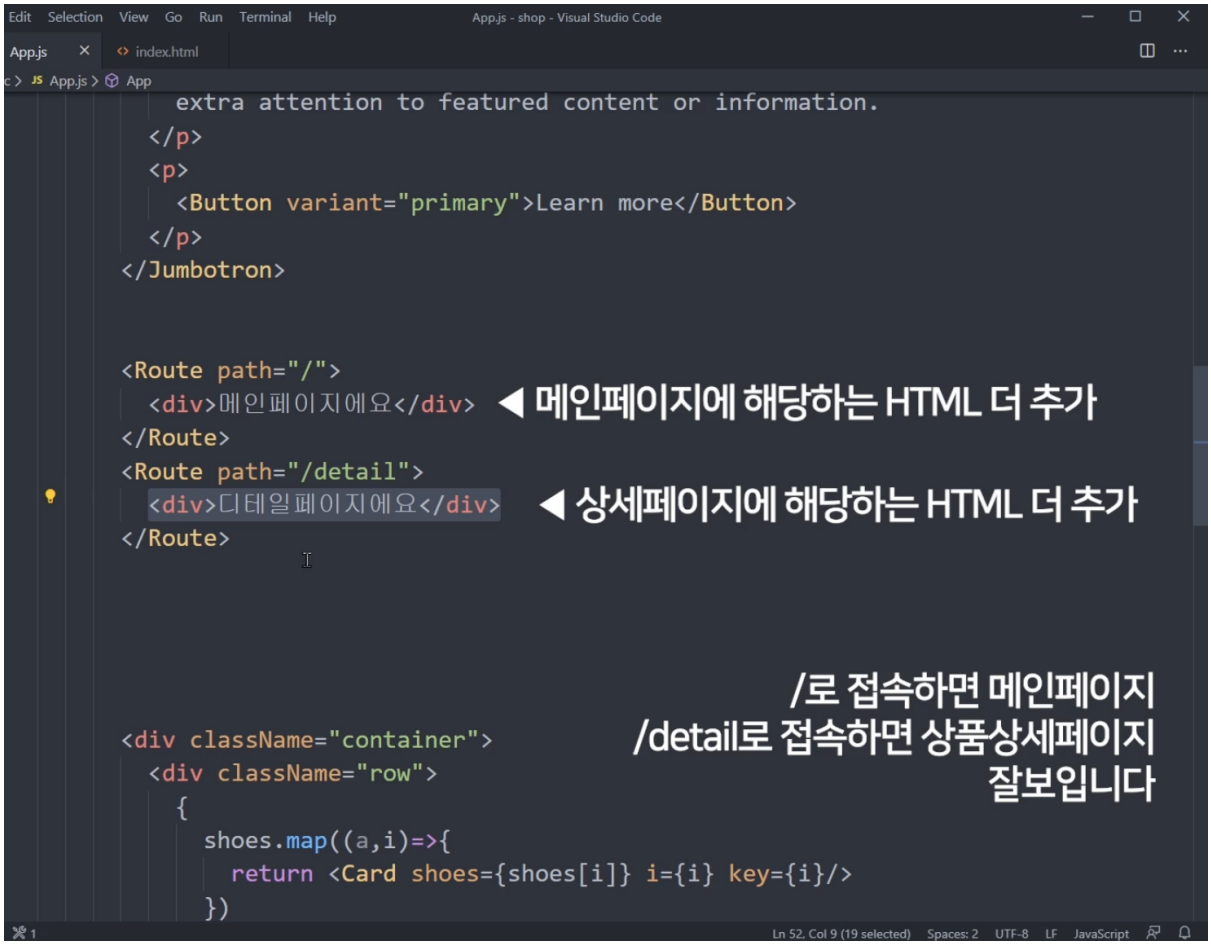
This is a simple hero unit, a simple jumbotron-style component for
calling
extra attention to featured content or information.

<Button variant="primary">Learn more</Button>

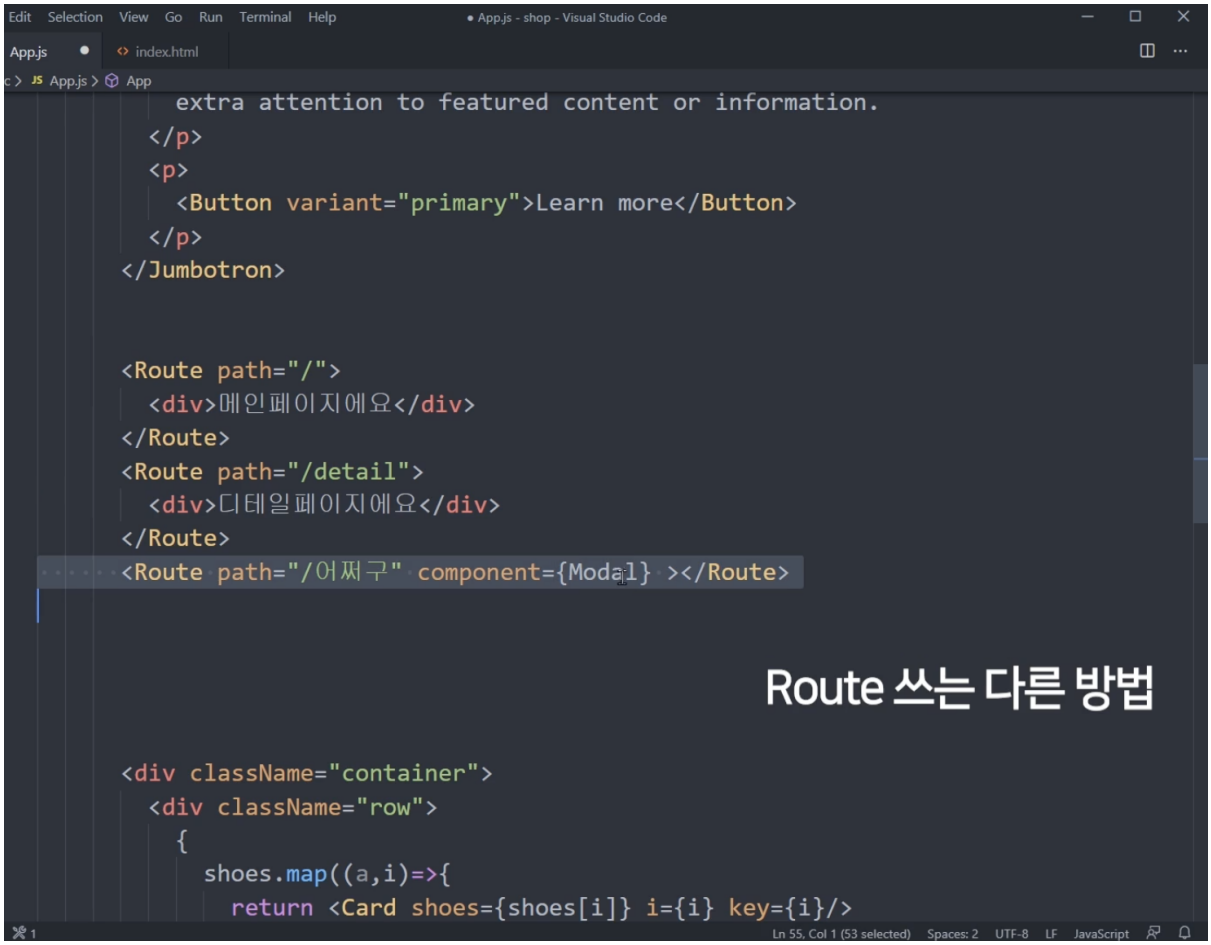
</Jumbotron>
2. <Route path="/경로"></Route>
<Route path="/"></Route>
<Route path="/detail"></Route>

{
shoes.map((a,i)=>{
return <Card shoes={shoes[i]} i={i} key={i}/>
})
}
Ln 49, Col 27 Spaces: 2 UTF-8 LF JavaScript

라우트를 나눈 것으로 어떤 사람이 '/' 로 접속을 하면 메인페이지를 보여주고, /detail 로 접속을 하면 상세페이지를 보여준다.

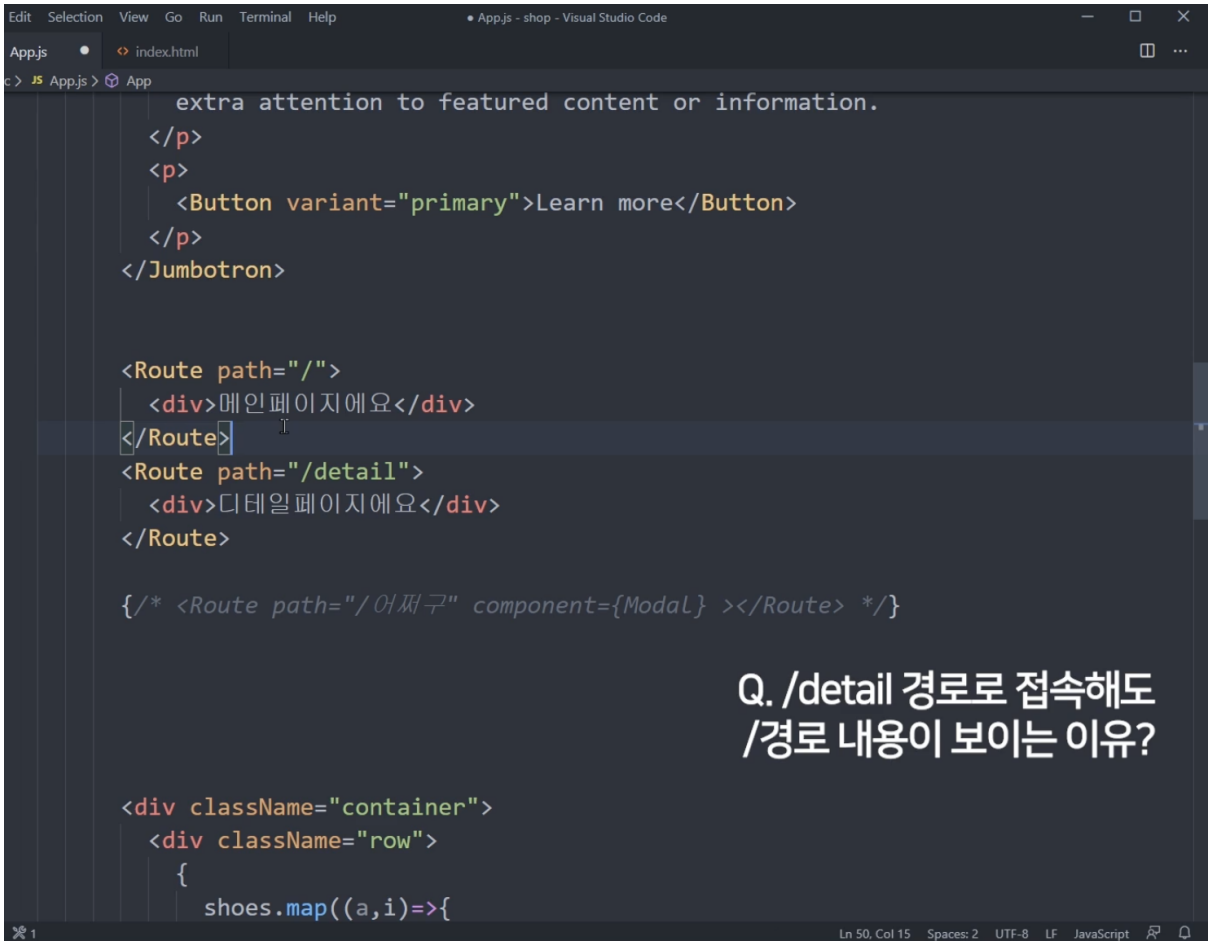


- Route 를 쓰는 다른 방법



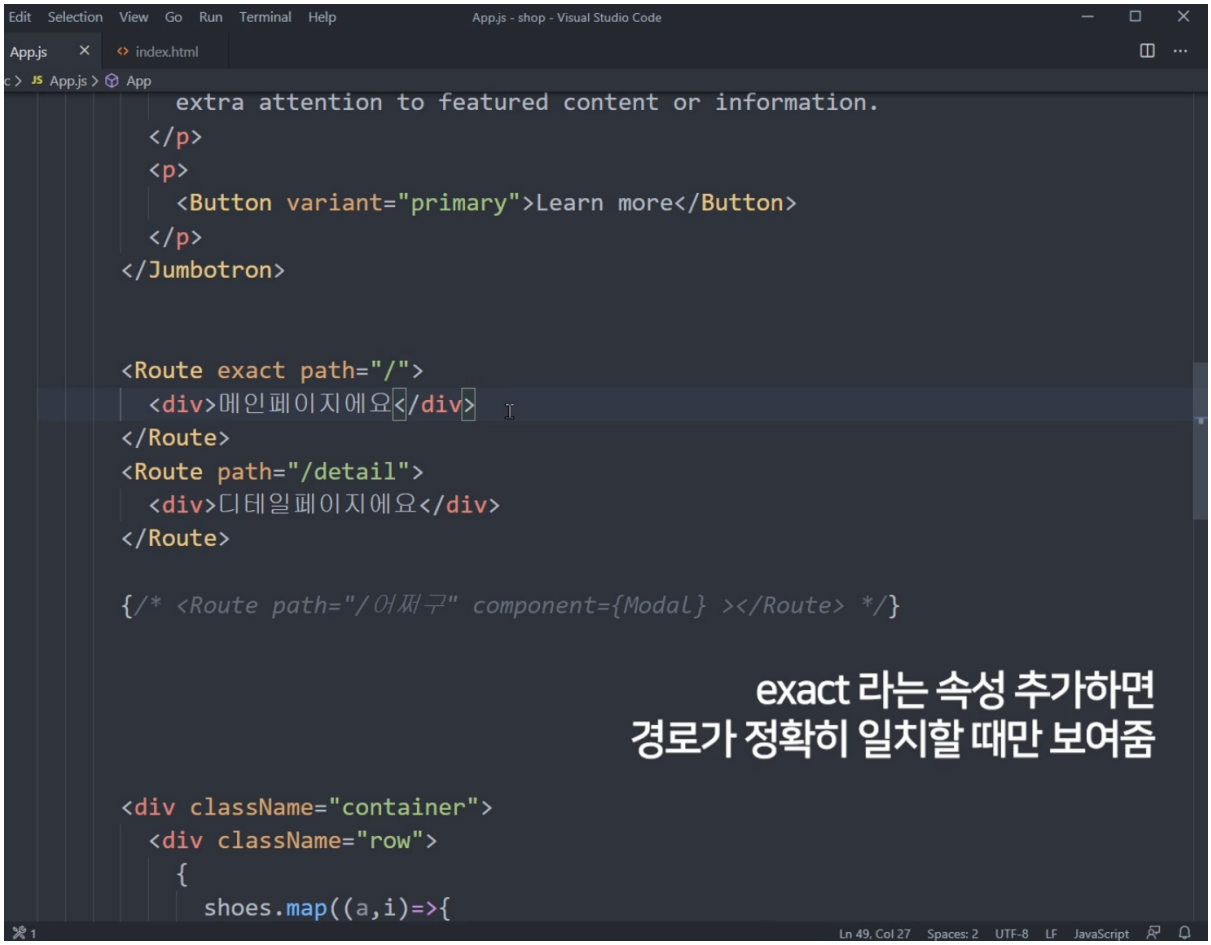
Route 태그 안에 component= { 컴포넌트 } 속성을 써서 바로 컴포넌트가 뜰 수 있도록 설정이 가능하다.

- /detail 경로로 접속해도 / 경로 내용이 보이는 이유?



매칭되는 것들은 다 보여준다. /detail 안에 / 도 있고, /detail 도 있다고 볼 수 있다.

해결법

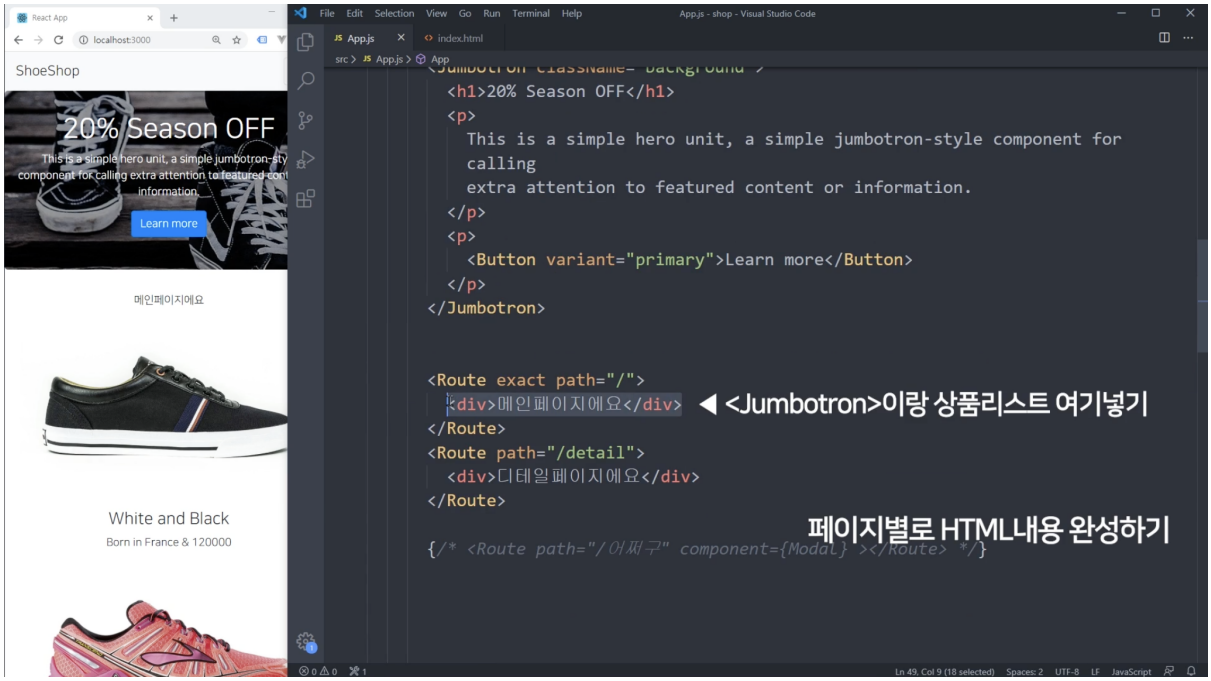

Edit Selection View Go Run Terminal Help
App.js - shop - Visual Studio Code
App.js
index.html
App.js > App
extra attention to featured content or information.

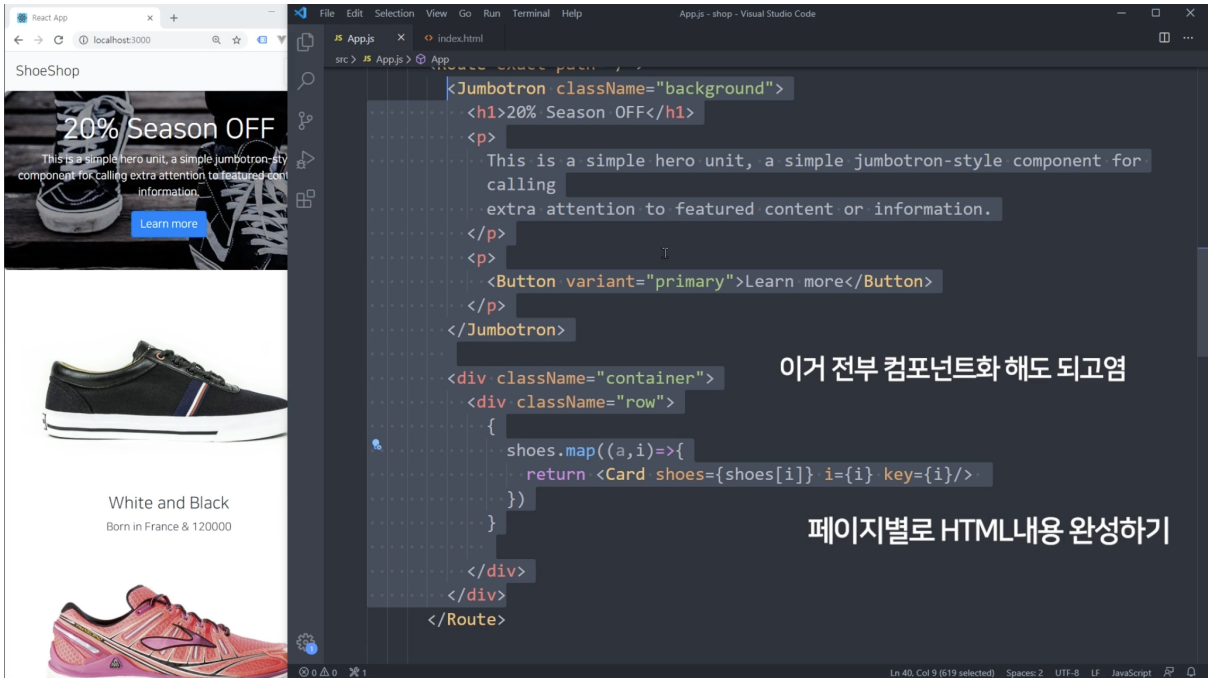
<Button variant="primary">Learn more</Button>

</Jumbotron>
<Route exact path="/">
메인페이지에요
</Route>
<Route path="/detail">
디테일페이지에요
</Route>
{/* <Route path="/어쩌구" component={Modal} ></Route> */}
exact 라는 속성 추가하면
경로가 정확히 일치할 때만 보여줌

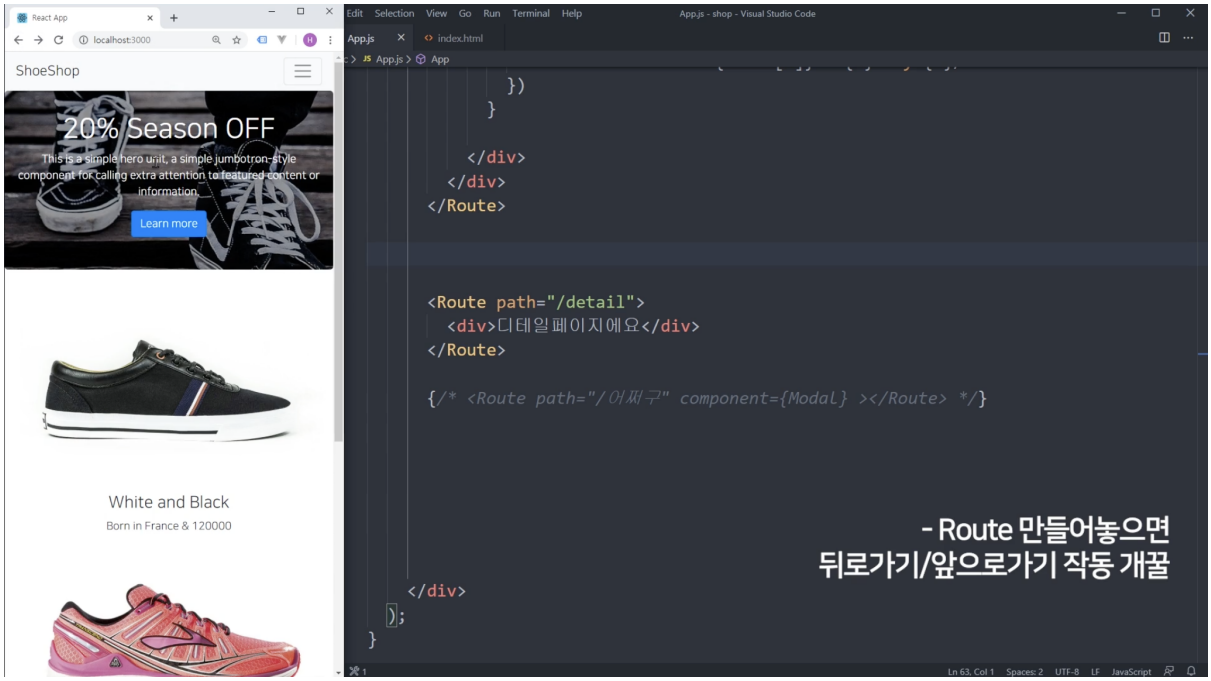
{
shoes.map((a,i)=>{
Ln 49, Col 27 Spaces: 2 UTF-8 LF JavaScript

- 페이지 완성하기



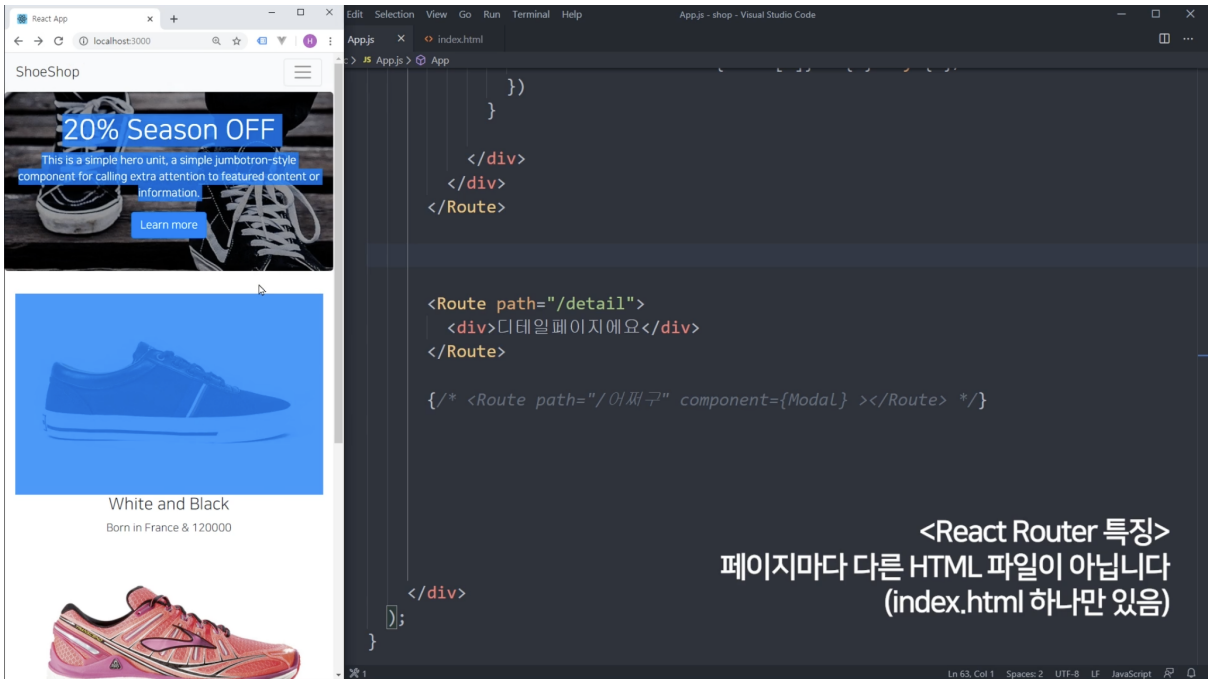


이것도 컴포넌트화를 해서 넣을 수 있다.


ShoeShop
20% Season OFF
This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.
Learn more
White and Black
Born in France & 120000

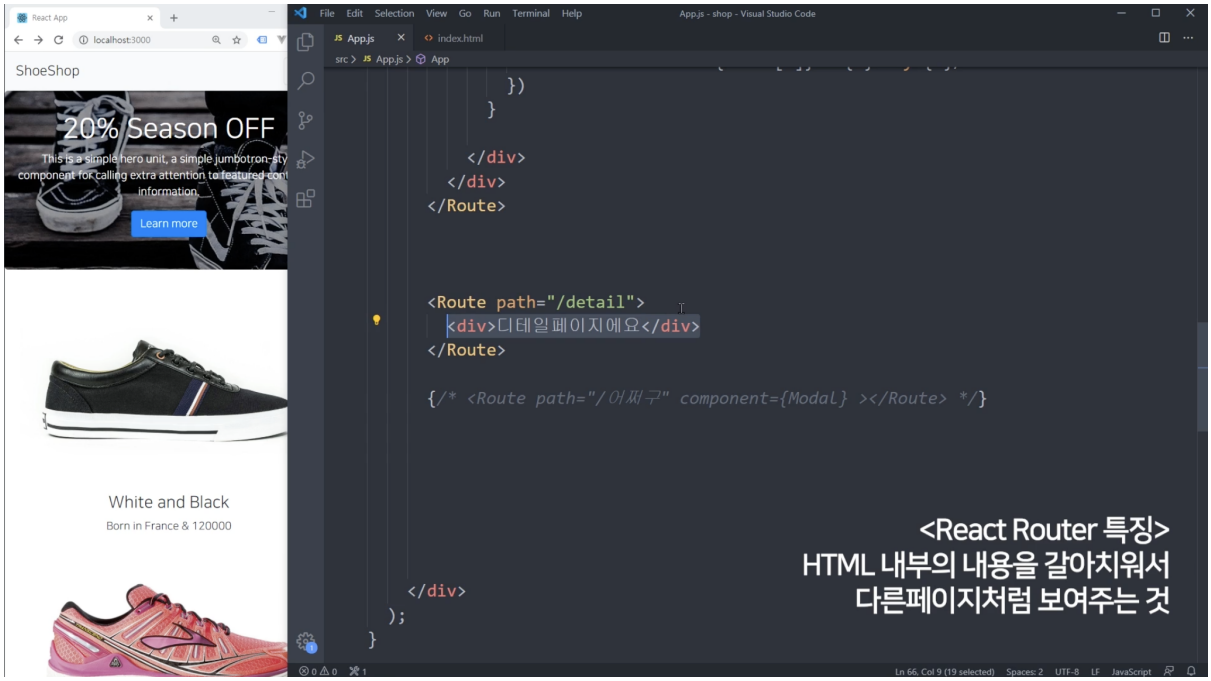
</Route>
<Route path="/detail">
디테일페이지에요
</Route>
{/* <Route path="/어쩌구" component={Modal} ></Route> */}

);
}
- Route 만들어놓으면
뒤로가기/앞으로가기 작동 개꿀


ShoeShop
20% Season OFF
This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content or information.
Learn more
White and Black
Born in France & 120000

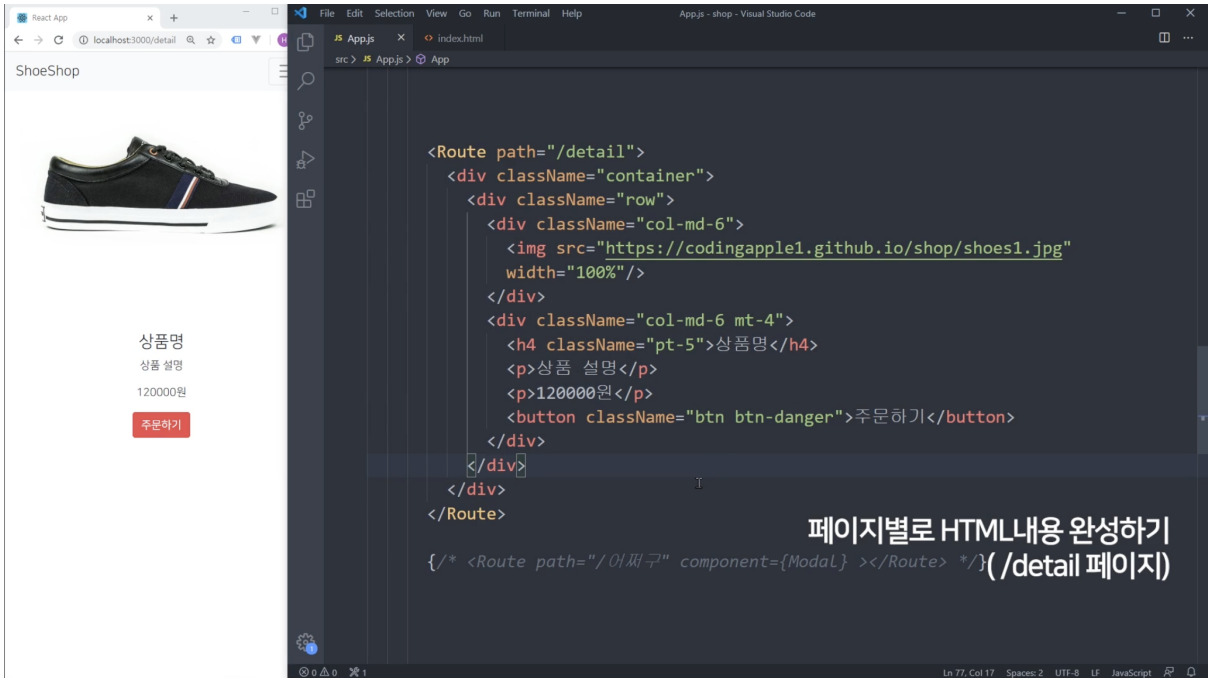
</Route>
<Route path="/detail">
디테일페이지에요
</Route>
{/* <Route path="/어쩌구" component={Modal} ></Route> */}

);
}
<React Router 특징>
페이지마다 다른 HTML 파일이 아닙니다
(index.html 하나만 있음)


ShoeShop
20% Season OFF
This is a simple hero unit, a simple jumbotron-sty
component for calling extra attention to featured cont
information.
Learn more
White and Black
Born in France & 120000

</Route>
<Route path="/detail">
디테일페이지에요
</Route>
{/* <Route path="/어쩌구" component={Modal} ></Route> */}

);
}
<React Router 특징>
HTML 내부의 내용을 갈아치워서
다른페이지처럼 보여주는 것


ShoeShop
상품명
상품 설명
120000원
주문하기
<Route path="/detail">

<img src="https://codingapple1.github.io/shop/shoes1.jpg"
width="100%" />

<h4 className="pt-5">상품명</h4>
상품 설명
120000원
<button className="btn btn-danger">주문하기</button>

</Route>
{/* <Route path="/어쩌구" component={Modal} ></Route> */}
페이지별로 HTML내용 완성하기
(/detail 페이지)

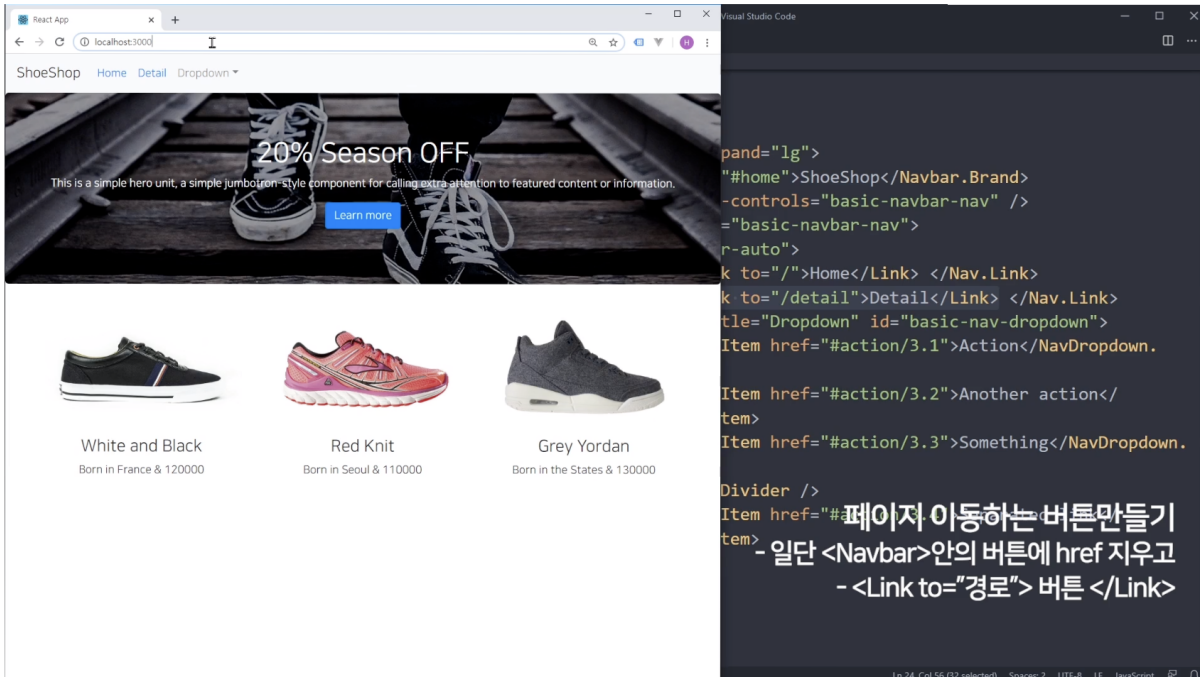
6) React Router 2 : Link, Switch, history 기능

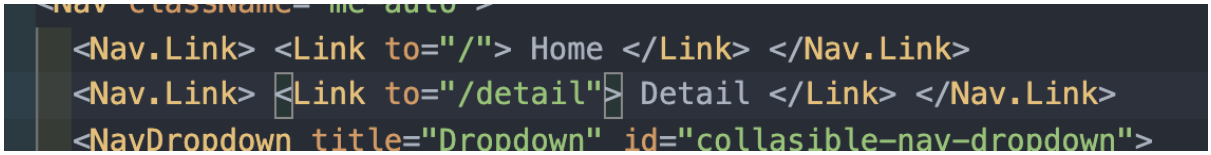
- 보통 컴포넌트를 파일로 만들 때는 대문자로 시작해야한다.

ex) Detail 컴포넌트 ==⇒ Detail.js

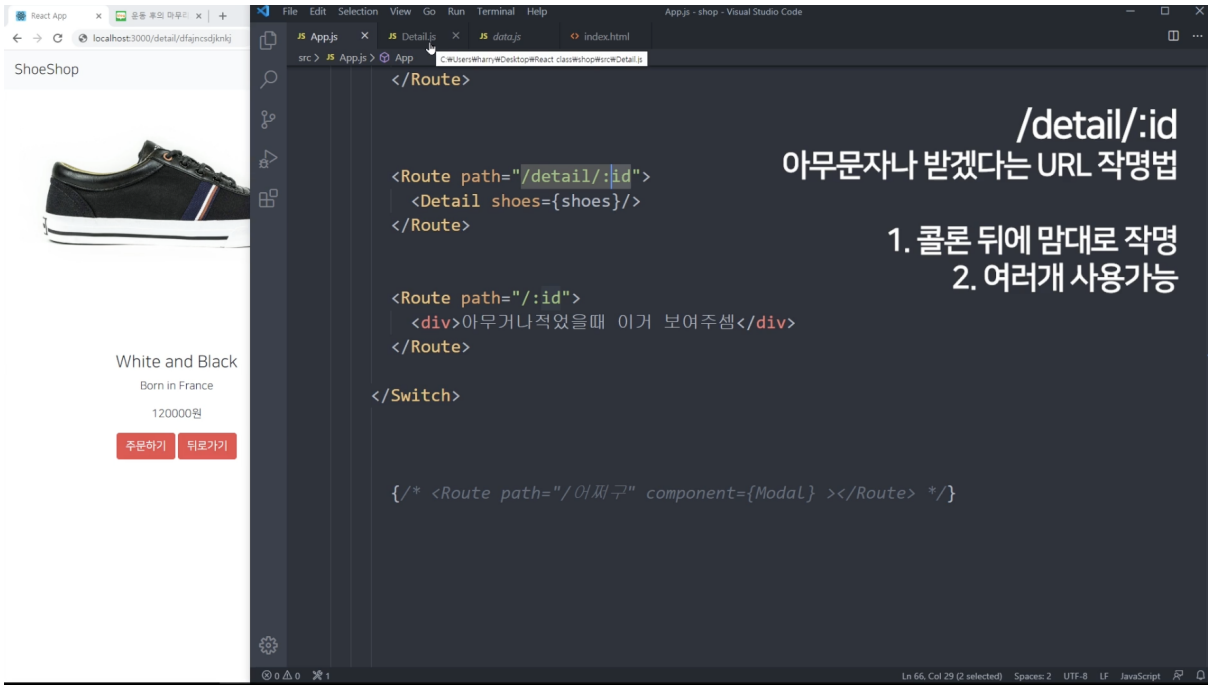
- 컴포넌트가 많아지면 src 에 하위폴더를 생성해서 컴포넌트 전용 폴더를 만들어서 관리한다.

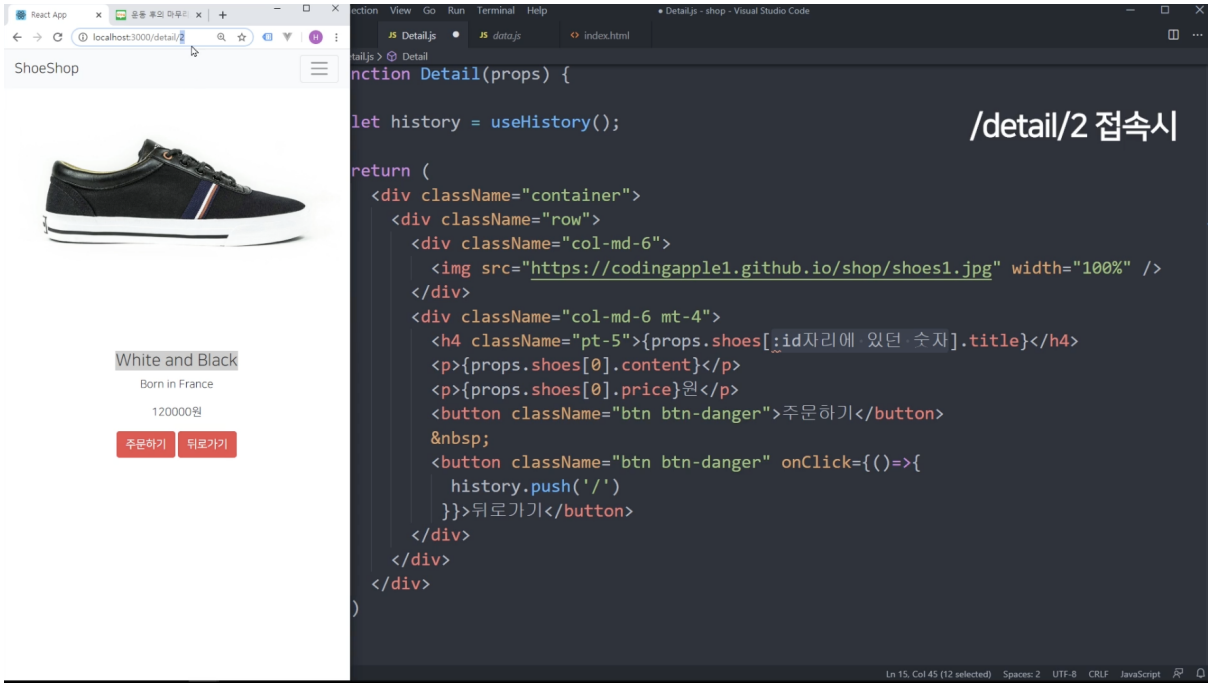
- Link 를 사용할 때 href 속성은 필요 없다.

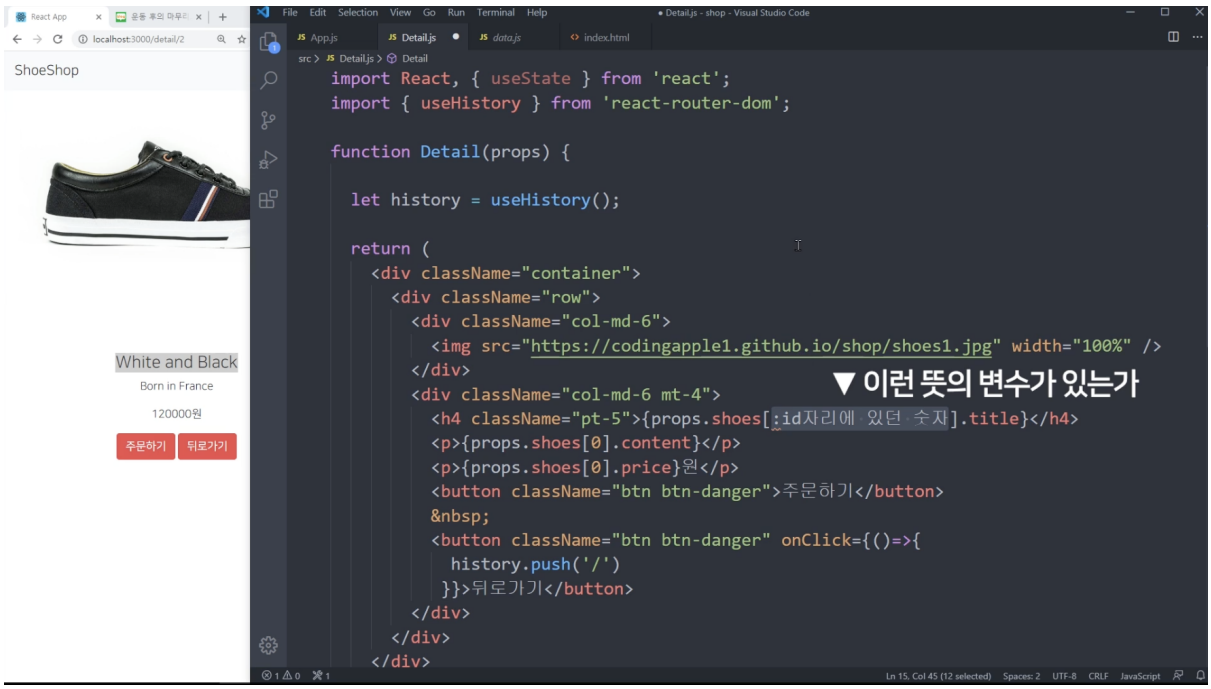




7) React Router 3 : URL 파라미터로 상세페이지 100개 만들기



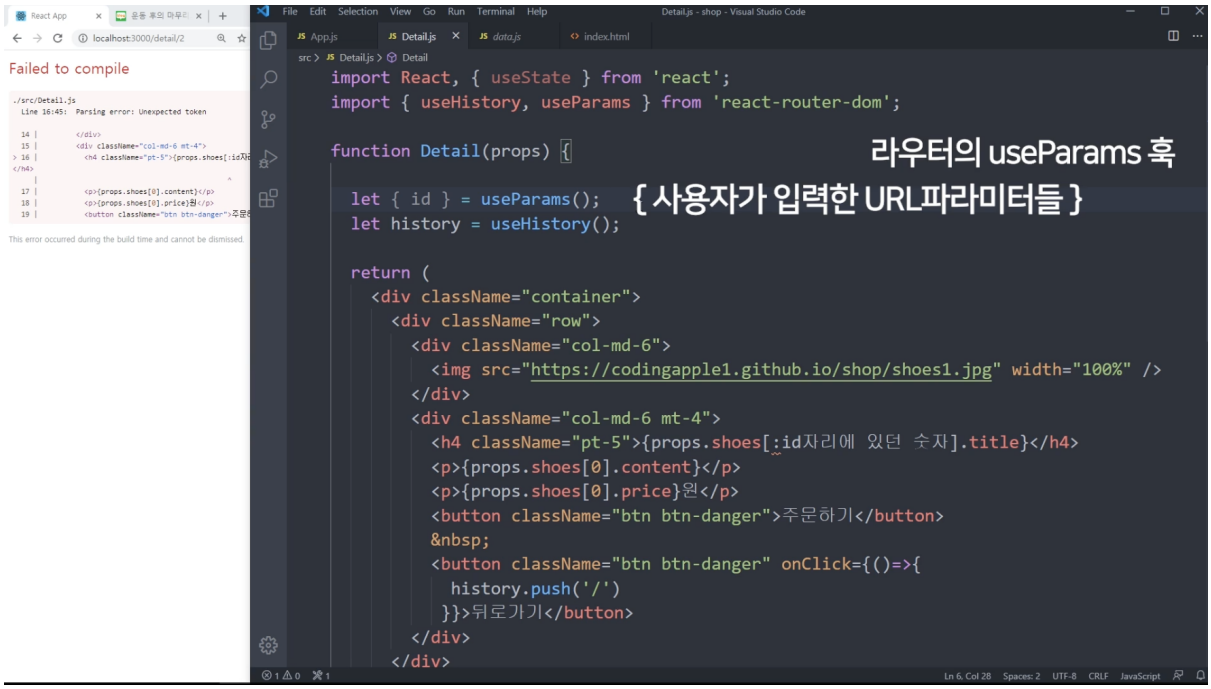



ShoeShop
White and Black
Born in France
120000원
주문하기
뒤로가기
import React, { useState } from 'react';
import { useHistory } from 'react-router-dom';
function Detail(props) {
let history = useHistory();
return (

<h4 className="pt-5">{props.shoes[:id자리에 있던 숫자].title}</h4>
{props.shoes[0].content}
{props.shoes[0].price}원
<button className="btn btn-danger">주문하기</button>

<button className="btn btn-danger" onClick={()=>{
history.push('/')
}}>뒤로가기</button>

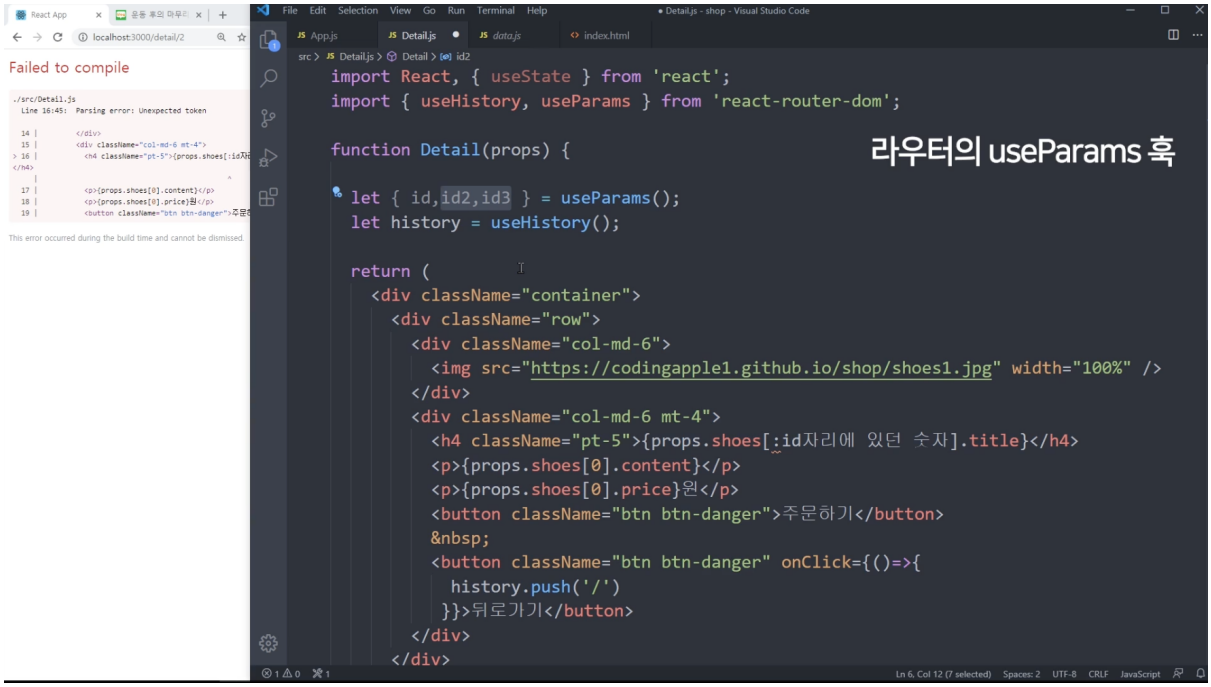
▼ 이런 뜻의 변수가 있는가


Failed to compile
import React, { useState } from 'react';
import { useHistory, useParams } from 'react-router-dom';
function Detail(props) {
let { id } = useParams();
let history = useHistory();
return (

<h4 className="pt-5">{props.shoes[:id자리에 있던 숫자].title}</h4>
{props.shoes[0].content}
{props.shoes[0].price}원
<button className="btn btn-danger">주문하기</button>

<button className="btn btn-danger" onClick={()=>{
history.push('/')
}}>뒤로가기</button>

라우터의 useParams 훅
{ 사용자가 입력한 URL파라미터들 }


Failed to compile
라우터의 useParams 훅
import React, { useState } from 'react';
import { useHistory, useParams } from 'react-router-dom';
function Detail(props) {
let { id,id2,id3 } = useParams();
let history = useHistory();
return (

<h4 className="pt-5">{props.shoes[:id자리에 있던 숫자].title}</h4>
{props.shoes[0].content}
{props.shoes[0].price}원
<button className="btn btn-danger">주문하기</button>

<button className="btn btn-danger" onClick={()=>{
history.push('/')
}}>뒤로가기</button>

여러개 사용 가능