

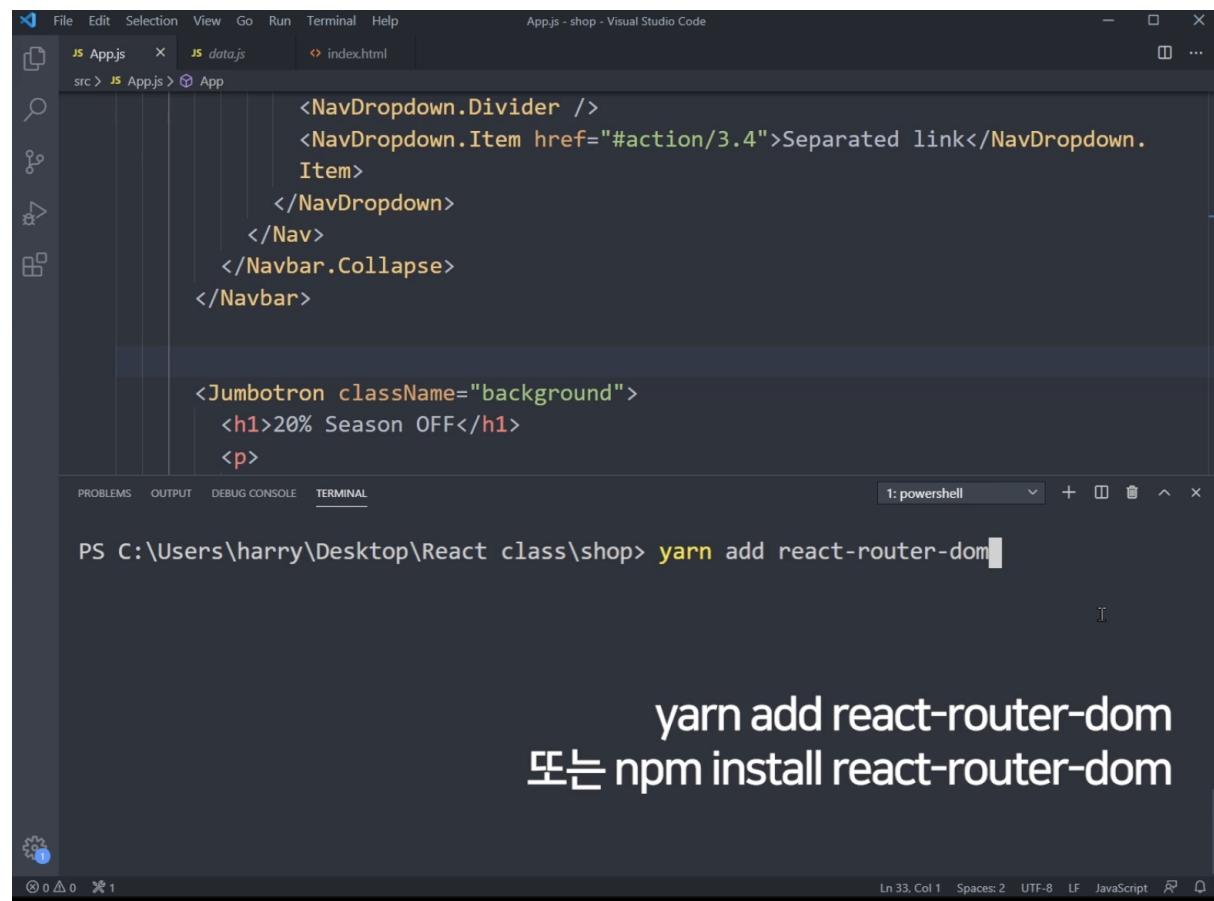
9월 6일 (월)

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.



```
<NavDropdown.Divider />
<NavDropdown.Item href="#action/3.4">Separated link</NavDropdown.
Item>
</NavDropdown>
</Nav>
</Navbar.Collapse>
</Navbar>

<Jumbotron className="background">
<h1>20% Season OFF</h1>
<p>
```

```
PS C:\Users\harry\Desktop\React class\shop> yarn add react-router-dom
```

yarn add react-router-dom
또는 npm install react-router-dom

라이브러리를 설치해준다.

The screenshot shows the Visual Studio Code interface with the title bar "index.js - shop - Visual Studio Code". The left sidebar has icons for file, folder, search, and refresh. The main editor area displays the following code:

```
File Edit Selection View Go Run Terminal Help
index.js - shop - Visual Studio Code
src > index.js
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
ReactDOM.render(
  <React.StrictMode>
    <App />
  </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();
```

react-router-dom 초기셋팅법

Ln 7, Col 50 Spaces:2 UTF-8 LF JavaScript ⚙️

index.js 에 import { BrowserRouter } 을 넣어준다.

```
File Edit Selection View Go Run Terminal Help
index.js - shop - Visual Studio Code
File Edit Selection View Go Run Terminal Help
index.js
src > index.js
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';
...
import { BrowserRouter } from 'react-router-dom';

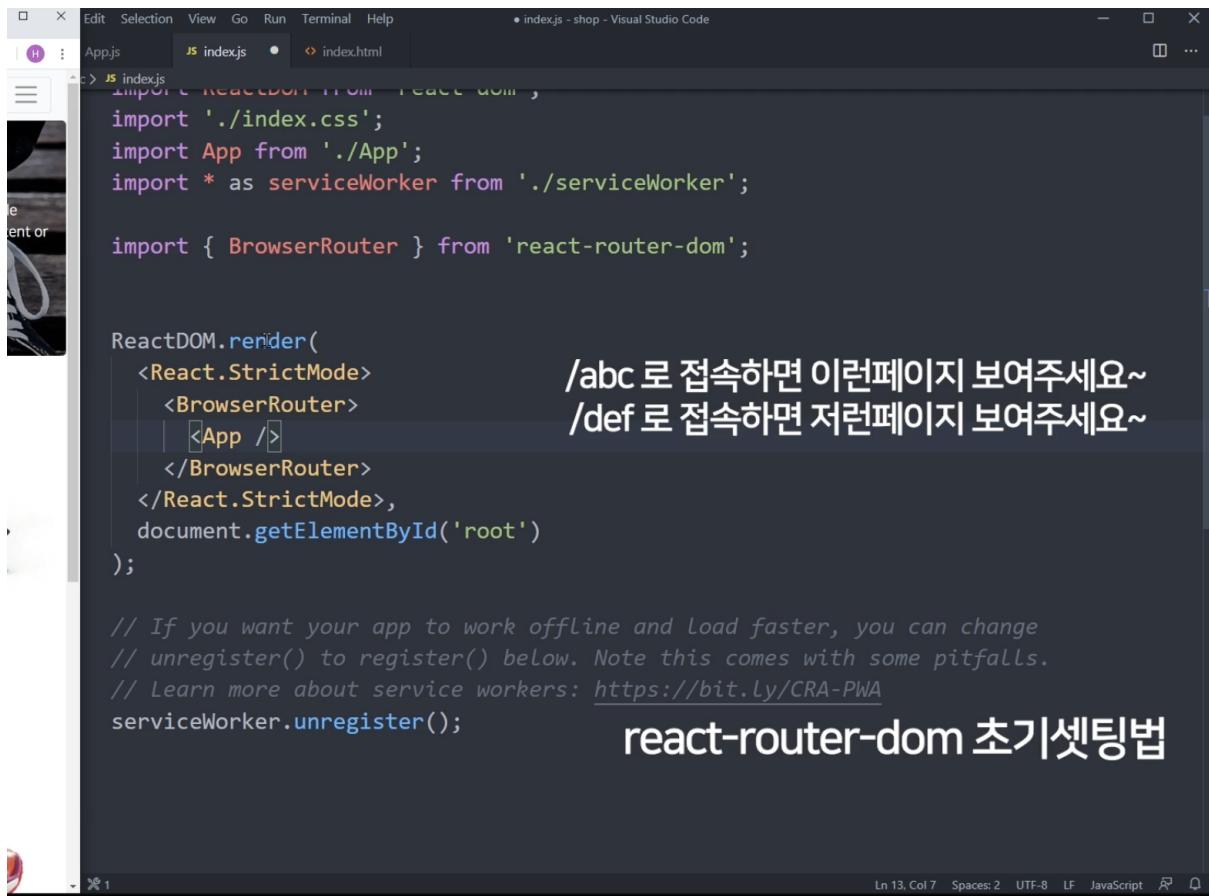
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);

// 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();
```

▲ 라이브러리 이름

react-router-dom 초기셋팅법

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



```
index.js - shop - Visual Studio Code
Edit Selection View Go Run Terminal Help
App.js JS index.js index.html
c> JS index.js
import { StrictMode } from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

import { BrowserRouter } from 'react-router-dom';

ReactDOM.render(
  <StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </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();

```

/abc로 접속하면 이런페이지 보여주세요~
/def로 접속하면 저런페이지 보여주세요~

react-router-dom 초기셋팅법

Ln 13, Col 7 Spaces:2 UTF-8 LF JavaScript ⌂ ⌂

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

- HashRouter VS BrowserRouter

localhost:3000/#/

```

index.js - shop - Visual Studio Code
Edit Selection View Go Run Terminal Help
App.js index.js index.html
import { HashRouter } from 'react-router-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

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

```

Edit Selection View Go Run Terminal Help
App.js index.js index.html
import { HashRouter } from 'react-router-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

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();

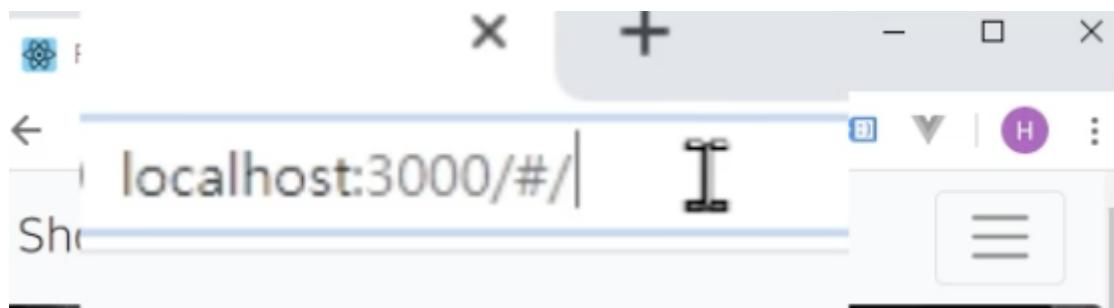
```

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 작성해둬야 한다.

- Route 를 만들어보자. (페이지를 나누자)

⇒ 메인페이지 와 상세페이지를 나눠보자.

```
import React, {useState} from 'react';
import { Navbar, Nav, NavDropdown, Button, Jumbotron } from 'react-bootstrap';
import './App.css';
import Data from './data.js';

import { Link, Route, Switch } from 'react-router-dom';

function App() {
  let [shoes, shoes변경] = useState(Data);

  return (
    <div className="App">

      <Navbar bg="light" expand="lg">
        <Navbar.Brand href="#home">ShoeShop</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="mr-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#link">Link</Nav.Link>
            <NavDropdown title="Dropdown" id="basic-nav-dropdown">
              <NavDropdown.Item href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">Another action</NavDropdown.Item>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Navbar>
    </div>
  );
}

export default App;
```

일단 Link Route, Switch 를 import 해온다.

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows 'App.js' as the active file.
- Code Editor:** Displays the following code snippet:

```
<p>
  This is a simple hero unit, a simple jumbotron-style component for
  calling
  extra attention to featured content or information.
</p>
<p>
  <Button variant="primary">Learn more</Button>
</p>
</Jumbotron>
```
- Text Block:** To the right of the code editor, the text "2. <Route path="/경로"/></Route>" is displayed.
- Code Snippet:** Below the current code, a code snippet for a 'Container' component is shown:

```
<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
        return <Card shoes={shoes[i]} i={i} key={i}/>
      })
    }
  </div>
</div>
```
- Status Bar:** Shows the current line and column (Ln 49, Col 27), character encoding (UTF-8), and file type (JavaScript).

```
extra attention to featured content or information.

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

<Route path="/">
  <div>메인페이지에요</div>
</Route>
<Route path="/detail">
  <div>상세페이지에요</div>
</Route>

<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
        return <Card shoes={shoes[i]} i={i} key={i}/>
      })
    }
  </div>
</div>
```

2. <Route path="/경로"></Route>
3. <Route>안에 HTML 적기

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

The screenshot shows a Visual Studio Code interface with the file `App.js` open. The code is a React component named `App` that contains a `Jumbotron` and two `Route` components. The first `Route` handles the root path and displays a main page message. The second `Route` handles the `/detail` path and displays a detail page message. Annotations in Korean are overlaid on the code:

- `<div>메인페이지에요</div>` ◀ 메인페이지에 해당하는 HTML 더 추가
- `<div>디테일페이지에요</div>` ◀ 상세페이지에 해당하는 HTML 더 추가
- `/로 접속하면 메인페이지
/detail로 접속하면 상품상세페이지
잘보입니다`

```
extra attention to featured content or information.

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

<Route path="/">
  <div>메인페이지에요</div> ◀ 메인페이지에 해당하는 HTML 더 추가
</Route>
<Route path="/detail">
  <div>디테일페이지에요</div> ◀ 상세페이지에 해당하는 HTML 더 추가
</Route>

<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
        return <Card shoes={shoes[i]} i={i} key={i}/>
      })
    }
  </div>
</div>

/로 접속하면 메인페이지
/detal로 접속하면 상품상세페이지
잘보입니다
```

- `Route` 를 쓰는 다른 방법

```
extra attention to featured content or information.

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

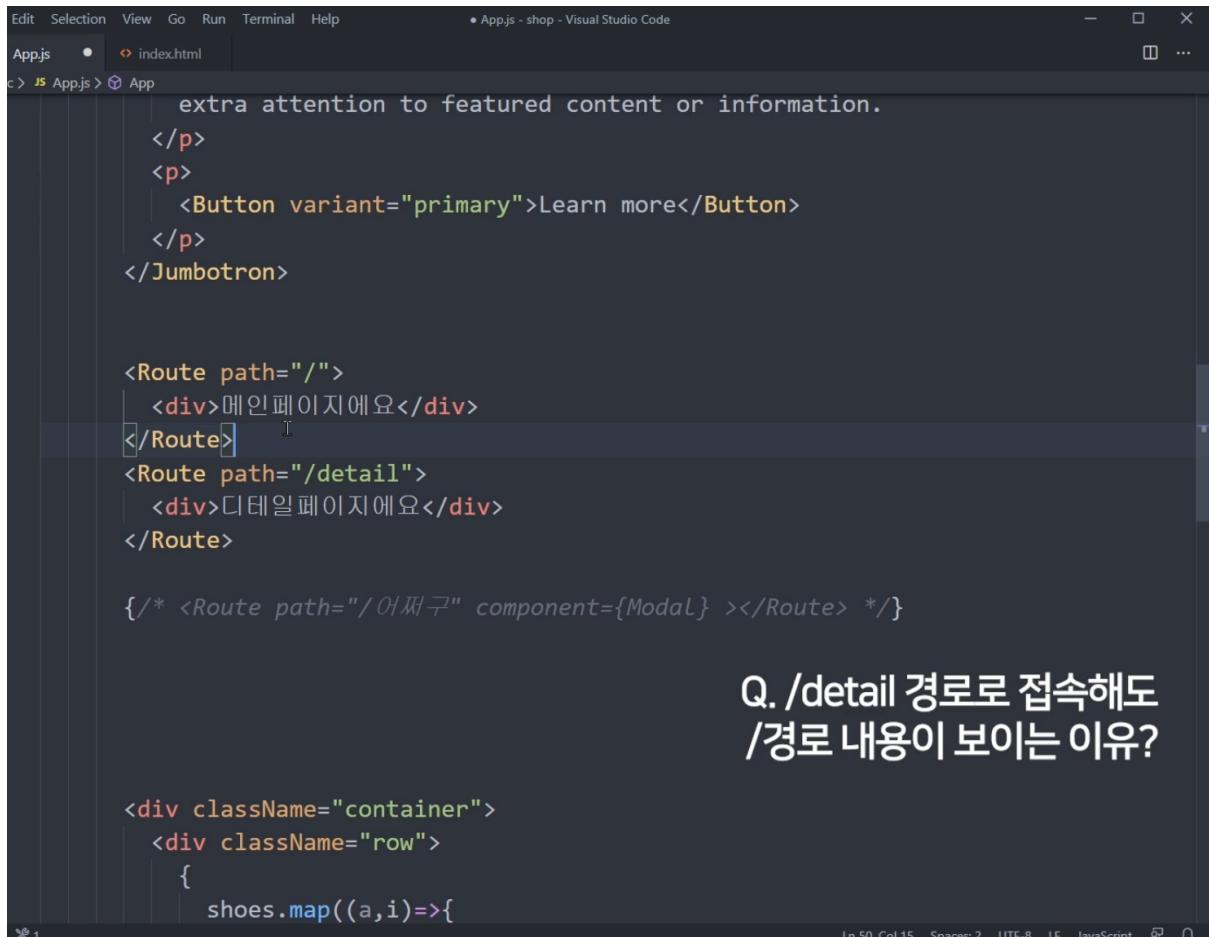
<Route path="/">
    <div>메인페이지에요</div>
</Route>
<Route path="/detail">
    <div>디테일페이지에요</div>
</Route>
<Route path="/어찌구" component={Modal1}></Route>
```

Route 쓰는 다른 방법

```
<div className="container">
    <div className="row">
        {
            shoes.map((a,i)=>{
                return <Card shoes={shoes[i]} i={i} key={i}/>
            })
        }
    </div>
</div>
```

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

- /detail 경로로 접속해도 / 경로 내용이 보이는 이유?



```
extra attention to featured content or information.

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

<Route path="/">
  <div>메인페이지에요</div>
</Route>
<Route path="/detail">
  <div>디테일페이지에요</div>
</Route>

/* <Route path="/Modal" component={Modal} ></Route> */
```

Q. /detail 경로로 접속해도
/경로 내용이 보이는 이유?

```
<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
```

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

해결법

```
extra attention to featured content or information.

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

<Route exact path="/">
  <div>메인페이지에요</div>
</Route>
<Route path="/detail">
  <div>디테일페이지에요</div>
</Route>

/* <Route path="/Modal" component={Modal} ></Route> */
```

**exact라는 속성 추가하면
경로가 정확히 일치할 때만 보여줌**

```
<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
```

Ln 49, Col 27 Spaces: 2 UTF-8 LF JavaScript ⚙ ⌂

- 페이지 완성하기

ShoeShop

```

<Jumbotron className="background">
  <h1>20% Season OFF</h1>
  <p>
    This is a simple hero unit, a simple jumbotron-style component for
    calling extra attention to featured content or information.
  </p>
  <p><Button variant="primary">Learn more</Button></p>
</Jumbotron>

<Route exact path="/">
  <div>메인페이지에요</div> ◀ <Jumbotron>이랑 상품리스트 여기 넣기
</Route>
<Route path="/detail">
  <div>디테일페이지에요</div>
</Route>
/* <Route path="/modal" component={Modal} ></Route> */

```

페이지별로 HTML내용 완성하기

White and Black
Born in France & 120000

White and Black
Born in France & 120000

ShoeShop

```

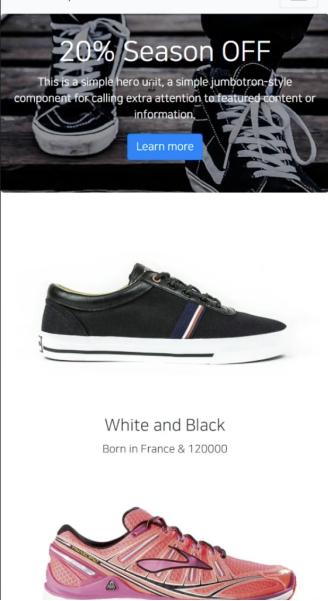
<Jumbotron className="background">
  <h1>20% Season OFF</h1>
  <p>
    This is a simple hero unit, a simple jumbotron-style component for
    calling extra attention to featured content or information.
  </p>
  <p><Button variant="primary">Learn more</Button></p>
</Jumbotron>

<div className="container">
  <div className="row">
    {
      shoes.map((a,i)=>{
        return <Card shoes={shoes[i]} i={i} key={i}/>
      })
    }
  </div>
</div>
</Route>

```

이거 전부 컴포넌트화 해도 되고 엄
페이지별로 HTML내용 완성하기

이것도 컴포넌트화를 해서 넣을 수 있다.



The screenshot shows a React application window titled "ShoeShop". On the left, there's a hero unit with a 20% discount offer. Below it are two shoe products: "White and Black" (black and white sneakers) and "Pink" (pink running shoes). The right side shows the corresponding code in App.js.

```

        }
      }
    
```

);
}

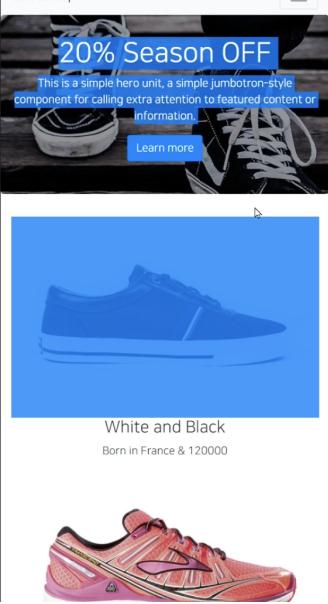
```

<Route path="/detail">
  <div>디테일페이지에요</div>
</Route>

/* <Route path="/modal" component={Modal} ></Route> */
```

);
}

- Route 만들어놓으면
뒤로가기/앞으로가기 작동 개꿀



The screenshot shows the same ShoeShop application. The "White and Black" shoe product is now highlighted in blue, while the "Pink" shoe is shown in its original color. The right side shows the corresponding code in App.js.

```

        }
      }
    
```

);
}

```

<Route path="/detail">
  <div>디테일페이지에요</div>
</Route>

/* <Route path="/modal" component={Modal} ></Route> */
```

);
}

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

ShoeShop

```

        <div>
          <div>
            <h1>20% Season OFF</h1>
            <p>This is a simple hero unit, a simple jumbotron-style component for calling extra attention to featured content information.</p>
            <a href="#">Learn more </a>
          </div>
        </div>

        <div>
          <h2>White and Black</h2>
          <p>Born in France & 120000</p>
          <img alt="Black and white sneaker" data-bbox="145 235 295 305" />
        </div>

        <div>
          <h2>Pink and Red</h2>
          <p>Born in France & 120000</p>
          <img alt="Pink and red sneaker" data-bbox="145 345 295 375" />
        </div>
      
```

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

ShoeShop

```

        <div>
          <div>
            <img alt="Black and white sneaker" data-bbox="145 535 295 585" />
          </div>
        </div>

        <div>
          <h3>상품명</h3>
          <p>상품 설명</p>
          <p>120000원</p>
          <button>주문하기</button>
        </div>
      
```

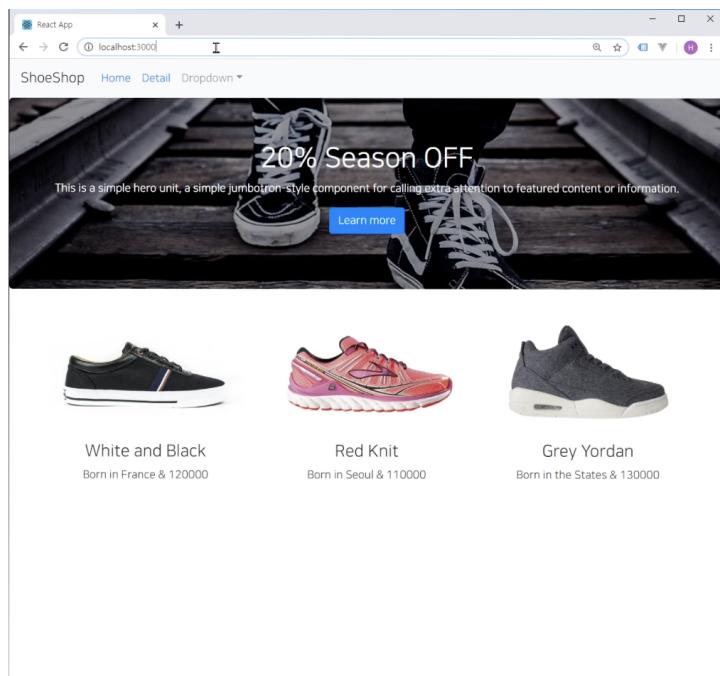
페이지별로 HTML내용 완성하기
(/detail 페이지)

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

- 보통 컴포넌트를 파일로 만들 때는 대문자로 시작해야 한다.

ex) Detail 컴포넌트 => Detail.js

- 컴포넌트가 많아지면 src에 하위폴더를 생성해서 컴포넌트 전용 폴더를 만들어서 관리 한다.
- Link를 사용할 때 href 속성은 필요 없다.



The screenshot shows a web browser window displaying a React application titled "ShoeShop". The navigation bar includes links for "Home", "Detail", and "Dropdown". Below the navigation is a hero unit featuring a pair of shoes with a "20% Season OFF" discount offer. The main content area displays three shoe models: "White and Black", "Red Knit", and "Grey Yordan", each with a price of 120000, 110000, and 130000 respectively. To the right of the browser window is a "Visual Studio Code" editor showing the corresponding React component code.

```
pand="lg">
"#home">ShoeShop</Navbar.Brand>
-controls="basic-navbar-nav" />
="basic-navbar-nav">
r-auto">
k to="/">Home</Link> </Nav.Link>
k to="/detail">Detail</Link> </Nav.Link>
tle="Dropdown" id="basic-nav-dropdown">
Item href="#action/3.1">Action</NavDropdown>
Item href="#action/3.2">Another action</NavDropdown>
Item href="#action/3.3">Something</NavDropdown>
Divider />
Item href="#">페이지 이동하는 버튼만들기</NavDropdown>
- 일단 <Navbar>안의 버튼에 href 지우고
- <Link to="경로"> 버튼 </Link>
```

```
<Nav className="mc auto">
  <Nav.Link> <Link to="/"> Home </Link> </Nav.Link>
  <Nav.Link> <Link to="/detail"> Detail </Link> </Nav.Link>
  <NavDropdown title="Dropdown" id="collapsible-nav-dropdown">
```

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



A screenshot of a web browser displaying a shoe detail page. The page shows a black and white photograph of a shoe with a small blue and red stripe on the side. Below the image, the text "White and Black" is displayed, followed by "Born in France" and "120000원". At the bottom are two red buttons labeled "주문하기" and "뒤로가기".

The browser's address bar shows the URL `localhost:3000/detail/dfajncsdknkj`. The Visual Studio Code editor window is open, showing the `App.js` file. The code includes a `<Route path="/detail/:id">` and a `<Route path="/:id">`.

`/detail/:id`
아무문자나 받겠다는 URL 작명법

1. 콜론 뒤에 맘대로 작명
2. 여러개 사용가능

```
JS App.js    JS Details.js    JS data.js    index.html
src > JS App.js > App
C:\Users\Harry\Desktop\React class\shop\src\Detail.js

</Route>

<Route path="/detail/:id">
| <Detail shoes={shoes}/>
</Route>

<Route path="/:id">
| <div>아무거나적었을때 이거 보여주셈</div>
</Route>

</Switch>

/* <Route path="/Modal" component={Modal} ></Route> */
```



A screenshot of a web browser displaying a shoe detail page. The page shows a black and white photograph of a shoe with a small blue and red stripe on the side. Below the image, the text "White and Black" is displayed, followed by "Born in France" and "120000원". At the bottom are two red buttons labeled "주문하기" and "뒤로가기".

The browser's address bar shows the URL `localhost:3000/detail/2`. The Visual Studio Code editor window is open, showing the `Detail.js` file. The code defines a `Detail` component that uses `useHistory()` to handle navigation.

`/detail/2 접속시`

```
JS Detail.js    JS data.js    index.html
Detail.js > Detail

function Detail(props) {
  let history = useHistory();

  return (
    <div className="container">
      <div className="row">
        <div className="col-md-6">
          
        </div>
        <div className="col-md-6 mt-4">
          <h4>{props.shoes[0].title}</h4>
          <p>{props.shoes[0].content}</p>
          <p>{props.shoes[0].price}</p>
          <button className="btn btn-danger">주문하기</button>
          &nbsp;
          <button className="btn btn-danger" onClick={()=>{
            history.push('/')
          }}>뒤로가기</button>
        </div>
      </div>
    </div>
  )
}
```

```

import React, { useState } from 'react';
import { useHistory } from 'react-router-dom';

function Detail(props) {
  let history = useHistory();

  return (
    <div className="container">
      <div className="row">
        <div className="col-md-6">
          
        </div>
        <div className="col-md-6 mt-4">
          <h4>{props.shoes[0].title}</h4>
          <p>{props.shoes[0].content}</p>
          <p>{props.shoes[0].price}원</p>
          <button className="btn btn-danger">주문하기</button>
          &ampnbsp
          <button className="btn btn-danger" onClick={()=>{
            history.push('/')
          }}>뒤로가기</button>
        </div>
      </div>
    </div>
  );
}

export default Detail;

```

▼ 이런 뜻의 변수가 있는가

Failed to compile

```

./src/Detail.js
Line 16:45:  Parsing error: Unexpected token
          </div>
          ^

          <div className="col-md-6 mt-4">
            <h4>{props.shoes[0].title}</h4>
            <p>{props.shoes[0].content}</p>
            <p>{props.shoes[0].price}원</p>
            <button className="btn btn-danger">주문하기</button>
          </div>
        </div>
      </div>
    </div>
  </div>

```

This error occurred during the build time and cannot be dismissed.

라우터의 useParams 혹은 사용자가 입력한 URL파라미터들

```

import React, { useState } from 'react';
import { useHistory, useParams } from 'react-router-dom';

function Detail(props) {
  let { id } = useParams();
  let history = useHistory();

  return (
    <div className="container">
      <div className="row">
        <div className="col-md-6">
          
        </div>
        <div className="col-md-6 mt-4">
          <h4>{props.shoes[0].title}</h4>
          <p>{props.shoes[0].content}</p>
          <p>{props.shoes[0].price}원</p>
          <button className="btn btn-danger">주문하기</button>
          &ampnbsp
          <button className="btn btn-danger" onClick={()=>{
            history.push('/')
          }}>뒤로가기</button>
        </div>
      </div>
    </div>
  );
}

export default Detail;

```

```
./src/Detail.js
Line 16:45:  Parsing error: Unexpected token

14 |         </div>
15 |         <div className="col-md-6 mt-4">
16 |             <h4 className="pt-5">{props.shoes[id2]}
17 |
18 |             <p>{props.shoes[0].content}</p>
19 |             <p>{props.shoes[0].price}</p>
20 |             <button className="btn btn-danger">주문하기</button>
```

라우터의 useParams 헥

This error occurred during the build time and cannot be dismissed.

Ln 6, Col 12 (7 selected) Spaces: 2 UTF-8 CRLF JavaScript ⚡

여러개 사용 가능