

PORTFOLIO

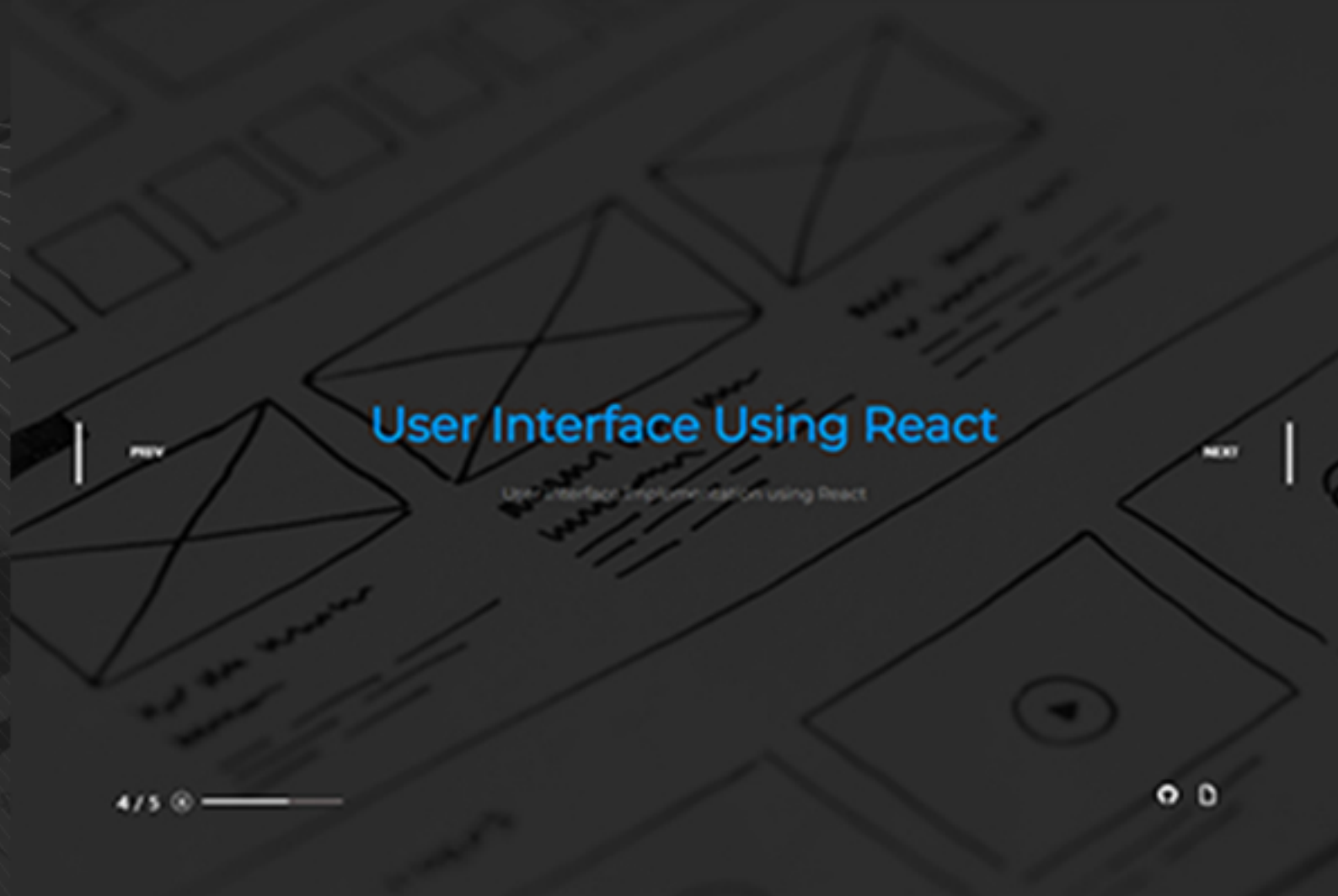
# React Deveopment Plan

- EcmaScript
- React Component
- JSX
- SwipeJS, ScollTrigger Library
- Redux Library
- Route Library

## EcmaScript 작성

ES5, ES6, ES2020 등은 ECMAScript가 배포된 버전입니다.

ECMAScript(javascript)는 엄연히 프로그래밍 언어이며,  
ES6 표준을 따른다 라는 말은 ECMAScript 2015가 사용중인 ECMA 규격을 따릅니다.  
ECMAScript 2015과 동일한 문법을 사용한다와 동의어라고 볼 수 있습니다.



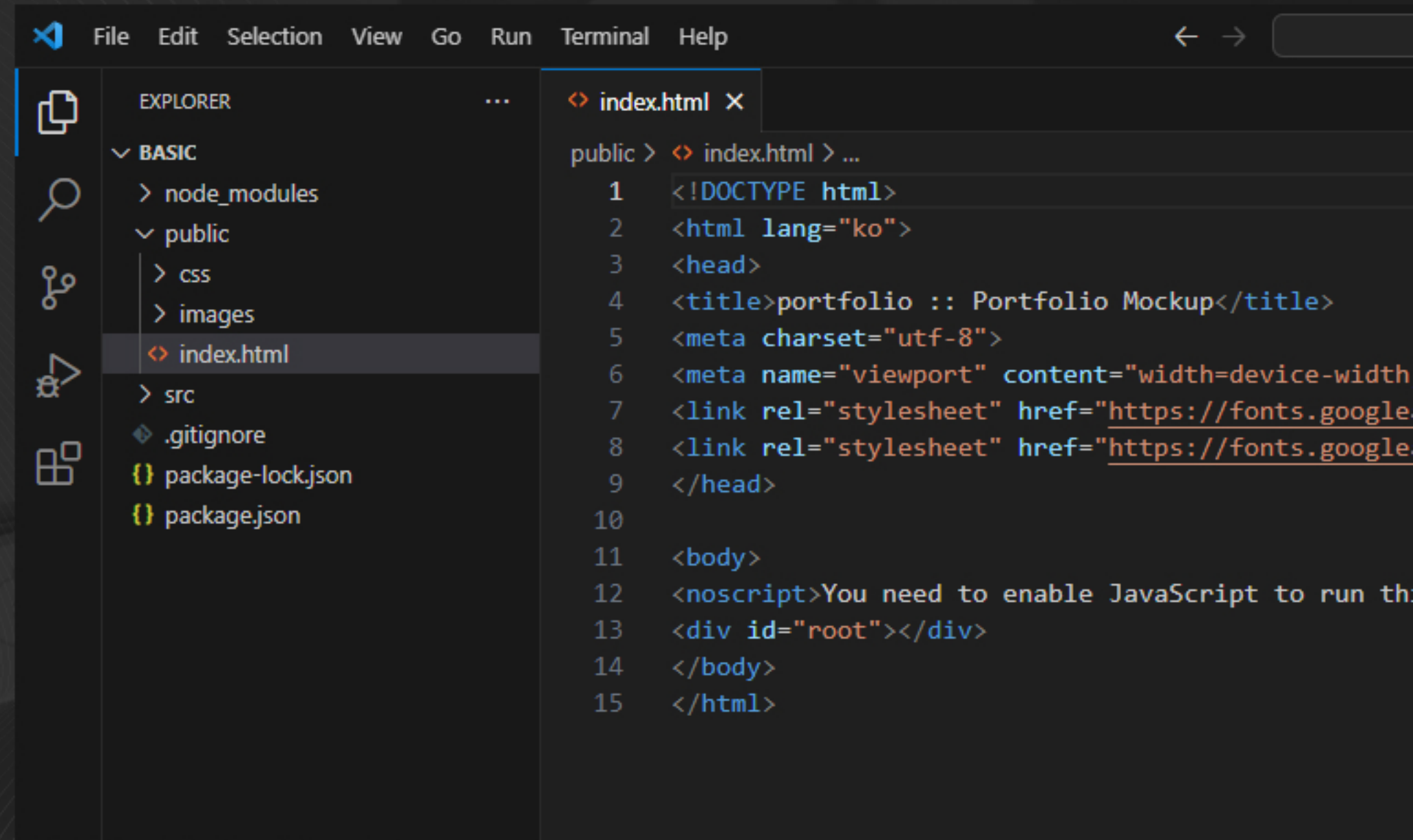
## public, App.js 구조 작성

React, Create React App 기본 구조를 작성합니다.

SwiperJS 노드를 제외하고 App.js 구조를 완성합니다.

단독 태그(img, br)을 중복 태그로 변경하고, class 속성을 className 속성으로 변경합니다.

a 태그의 href="" 속성을 href="/" 속성으로 변경합니다.



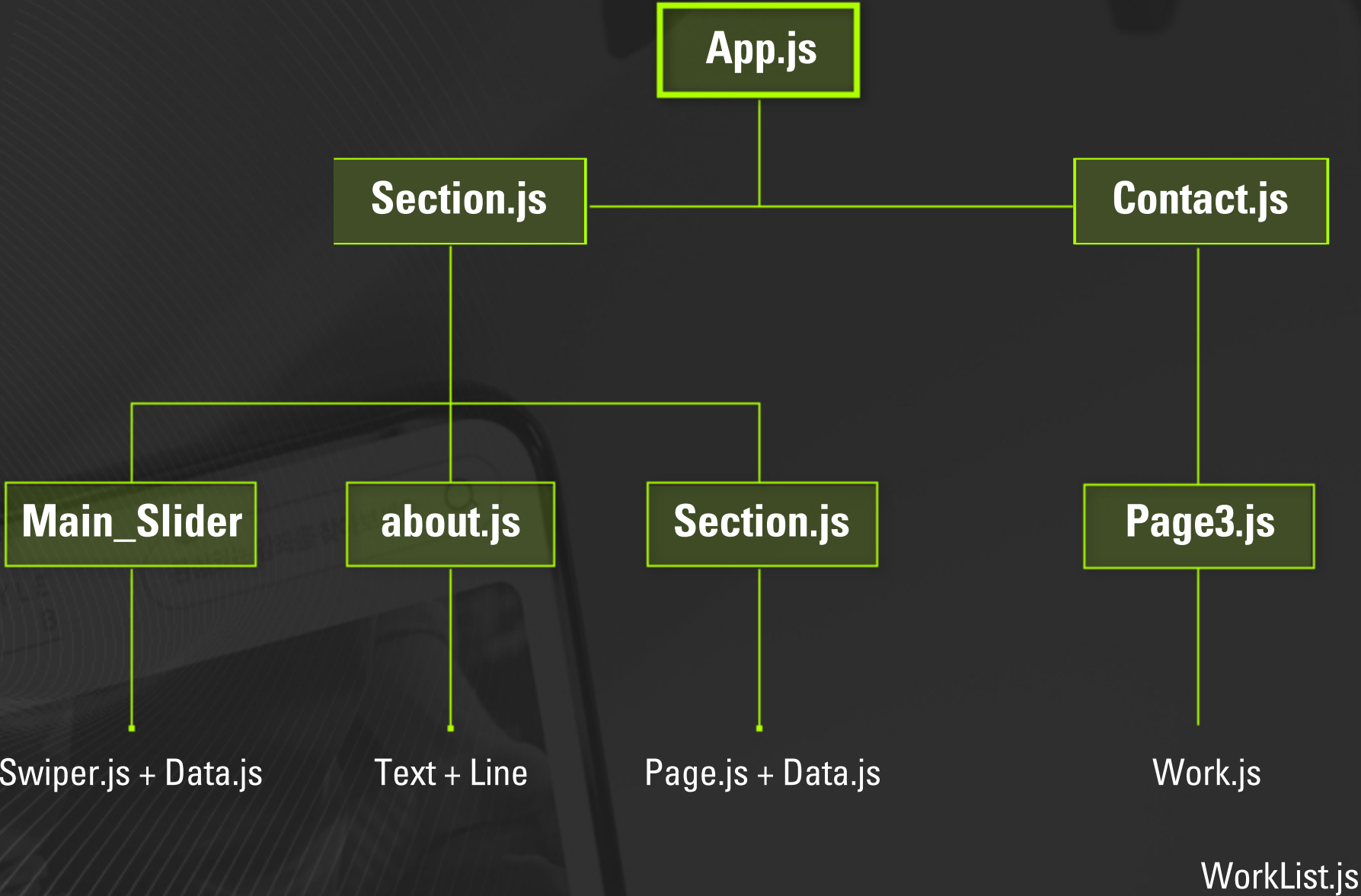


# Component 분리

정적인 페이지 구조와 동적인 페이지 구조를 구분하여 컴포넌트를 분리합니다.

React 컴포넌트는 클래스 또는 함수로 정의할 수 있습니다.

함수형 컴포넌트를 사용해 여러가지의 라이브러리를 활용할 수 있습니다.



# UI JavaScript 작성 1

SwiperJS JavaScript를 제외하고, UI JavaScript를 완성합니다.

다음은 추가 라이브러리입니다.

import useEffect from 'react';

import useSelector from 'react-redux';

다음은 업데이트 파일입니다.

src/component/swiper.js

```
section.js - portfolio.re - Visual Studio Code

src > JS section.js > ...
1  import { useEffect } from 'react';
2  import { useSelector } from "react-redux"
3  import SwiperJS from './swiperJS';
4  import About from './about';
5
6  function Section(){
7    let spec = useSelector((state) => state.spec )
8    useEffect(()=>{
9      let number=document.querySelectorAll(".number");
10     let numberLi = Array.from(number);
11     for(let i=0; i<numberLi.length; i++){
12       numberLi[i].addEventListener("mouseenter",()=>{
13         number[i].classList.add("active");
14         setTimeout(()=>{
15           number[i].classList.remove("active");
16         },450);
17       })
18     }
19   })
20   return(
21     <>
22     <div className='main_slider'>
23       <SwiperJS />
24     </div>
25     <About />
26     <section id='section'>
27       {
28         spec.map((d,i)->{
29           return(
30             <section className='page' key={i}>
31               <div className='number'>
32                 { "0"+(i+1) }
33               </div>
34               <div className='spec'>
35                 <p className='title'>{ spec[i].name }</p>
36                 <span>{ spec[i].detail }</span>
37               </div>
```



## UI JavaScript 작성2

SwiperJS JavaScript를 완성합니다.

다음은 추가 라이브러리입니다.

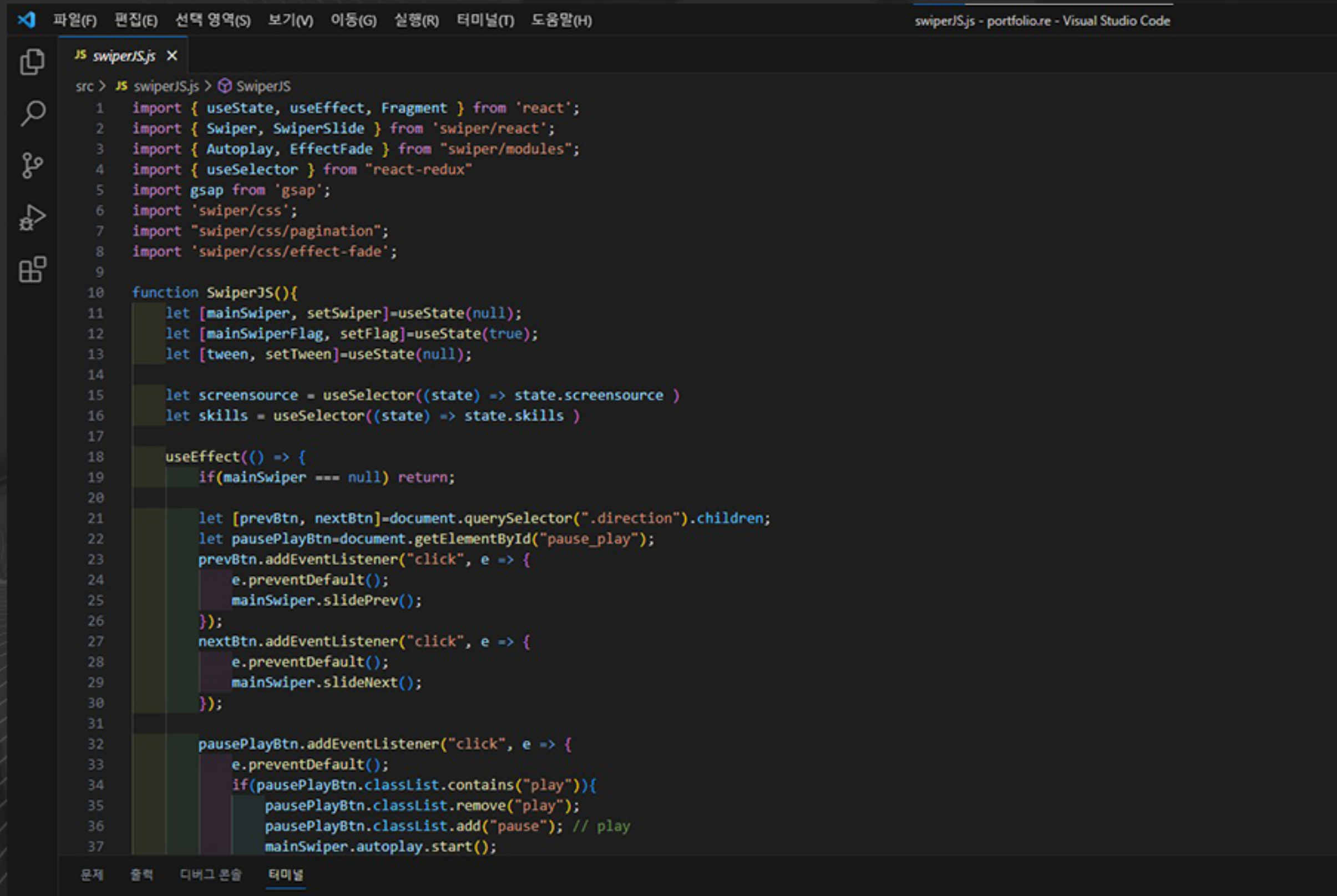
```
import { Swiper, SwiperSlide } from 'swiper/react';
```

```
import gsap from 'gsap';
```

```
import useSelector from 'react-redux';
```

다음은 업데이트 파일입니다.

```
src/component/Swiper.js
```



```
JS swiperJS.js X
src > JS swiperJS.js > SwiperJS
1  import { useState, useEffect, Fragment } from 'react';
2  import { Swiper, SwiperSlide } from 'swiper/react';
3  import { Autoplay, EffectFade } from "swiper/modules";
4  import { useSelector } from "react-redux"
5  import gsap from 'gsap';
6  import 'swiper/css';
7  import "swiper/css/pagination";
8  import 'swiper/css/effect-fade';
9
10 function SwiperJS(){
11   let [mainSwiper, setSwiper]=useState(null);
12   let [mainSwiperFlag, setFlag]=useState(true);
13   let [tween, setTween]=useState(null);
14
15   let screensource = useSelector((state) => state.screensource )
16   let skills = useSelector((state) => state.skills )
17
18   useEffect(() => {
19     if(mainSwiper === null) return;
20
21     let [prevBtn, nextBtn]=document.querySelector(".direction").children;
22     let pausePlayBtn=document.getElementById("pause_play");
23     prevBtn.addEventListener("click", e => {
24       e.preventDefault();
25       mainSwiper.slidePrev();
26     });
27     nextBtn.addEventListener("click", e => {
28       e.preventDefault();
29       mainSwiper.slideNext();
30     });
31
32     pausePlayBtn.addEventListener("click", e => {
33       e.preventDefault();
34       if(pausePlayBtn.classList.contains("play")){
35         pausePlayBtn.classList.remove("play");
36         pausePlayBtn.classList.add("pause"); // play
37         mainSwiper.autoplay.start();
```

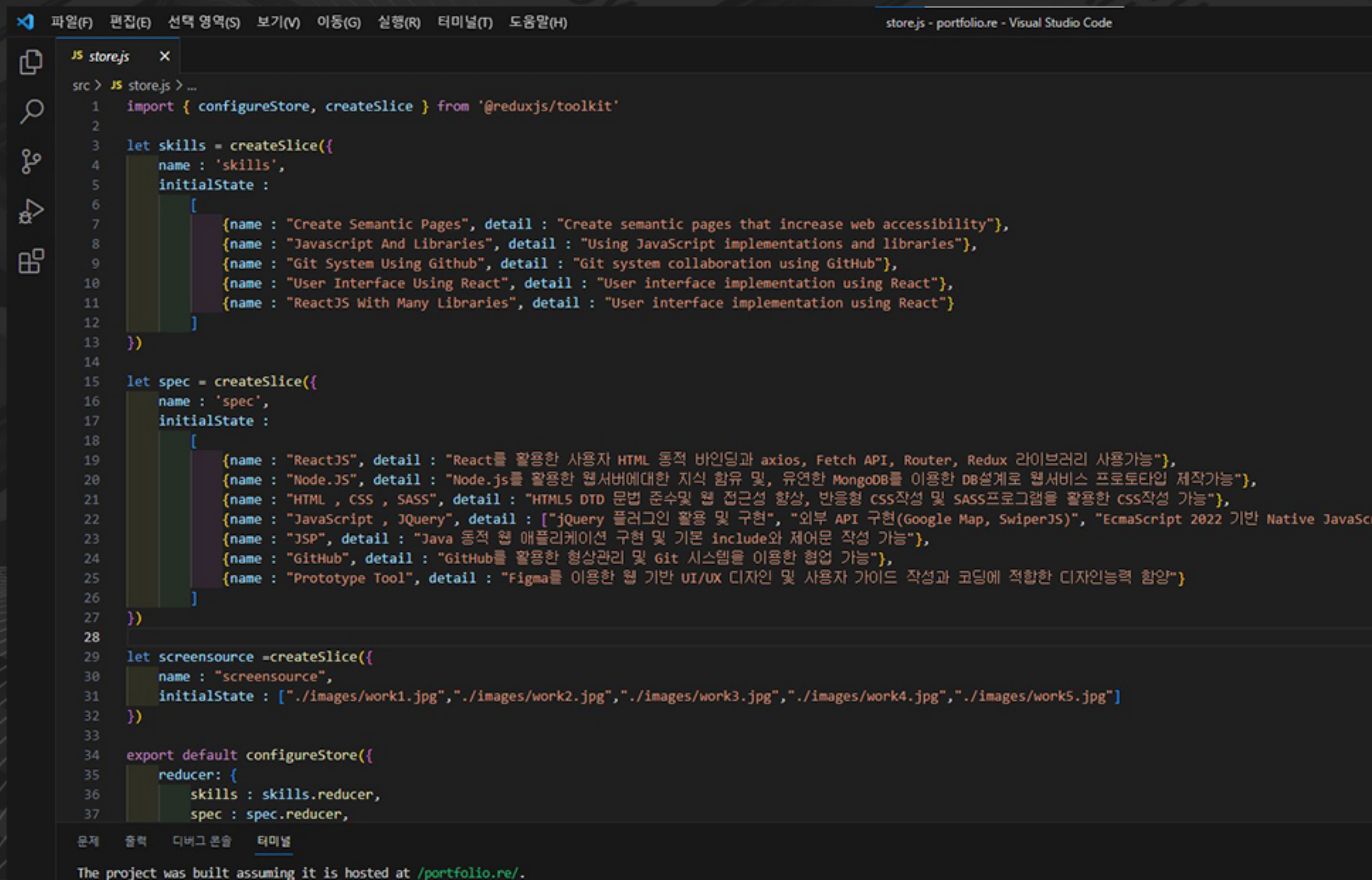


# Redux, state 컨테이너 작성

state 컨테이너를 작성합니다.

다음은 업데이트 파일입니다.

src/store.js



```
src > JS store.js > ...
1  import { configureStore, createSlice } from '@reduxjs/toolkit'
2
3  let skills = createSlice({
4    name : 'skills',
5    initialState :
6    [
7      {name : "Create Semantic Pages", detail : "Create semantic pages that increase web accessibility"},
8      {name : "Javascript And Libraries", detail : "Using JavaScript implementations and libraries"},
9      {name : "Git System Using Github", detail : "Git system collaboration using GitHub"},
10     {name : "User Interface Using React", detail : "User interface implementation using React"},
11     {name : "ReactJS With Many Libraries", detail : "User interface implementation using React"}
12   ]
13 })
14
15 let spec = createSlice({
16   name : 'spec',
17   initialState :
18   [
19     {name : "ReactJS", detail : "React를 활용한 사용자 HTML 동적 바인딩과 axios, Fetch API, Router, Redux 라이브러리 사용가능"},
20     {name : "Node.JS", detail : "Node.js를 활용한 웹서버에대한 지식 함양 및, 유연한 MongoDB를 이용한 DB설계로 웹서비스 프로토타입 제작가능"},
21     {name : "HTML , CSS , SASS", detail : "HTML5 DTD 문법 준수및 웹 접근성 향상, 반응형 CSS작성 및 SASS프로그램을 활용한 CSS작성 가능"},
22     {name : "JavaScript , JQuery", detail : ["jQuery 플러그인 활용 및 구현", "외부 API 구현(Google Map, SwiperJS)", "EcmaScript 2022 기반 Native JavaScript 활용 가능"]},
23     {name : "JSP", detail : "Java 동적 웹 애플리케이션 구현 및 기본 include와 제어문 작성 가능"},
24     {name : "Github", detail : "Github를 활용한 형상관리 및 Git 시스템을 이용한 협업 가능"},
25     {name : "Prototype Tool", detail : "Figma를 이용한 웹 기반 UI/UX 디자인 및 사용자 가이드 작성과 코딩에 적합한 디자인능력 함양"}
26   ]
27 })
28
29 let screensource =createSlice({
30   name : "screensource",
31   initialState : ["/images/work1.jpg","/images/work2.jpg","/images/work3.jpg","/images/work4.jpg","/images/work5.jpg"]
32 })
33
34 export default configureStore({
35   reducer: {
36     skills : skills.reducer,
37     spec : spec.reducer,
```

문제 출력 디버그 콘솔 터미널

The project was built assuming it is hosted at /portfolio.re/.



## Redux, state 컨테이너 작성

```
let screensource =createSlice({  
  name : "screensource",  
  initialState : ["/images/work1.jpg","/images/work2.jpg","/images/work3.jpg","/images/work4.jpg","/images/work5.jpg"]  
})
```

## User Interface Using React

User interface implementation using React

4 / 5



# Redux, state 컨테이너 작성

```
1 import { configureStore, createSlice } from '@reduxjs/toolkit'
2
3 let skills = createSlice({
4   name : 'skills',
5   initialState :
6   [
7     {name : "Create Semantic Pages", detail : "Create semantic pages that increase web accessibility"},
8     {name : "Javascript And Libraries", detail : "Using JavaScript implementations and libraries"},
9     {name : "Git System Using Github", detail : "Git system collaboration using GitHub"},
10    {name : "User Interface Using React", detail : "User interface implementation using React"},
11    {name : "ReactJS With Many Libraries", detail : "User interface implementation using React"}
12  ]
13 })
14
15 let spec = createSlice({
16   name : 'spec',
17   initialState :
18   [
19     {name : "ReactJS", detail : "React를 활용한 사용자 HTML 동적 바인딩과 axios, Fetch API, Router, Redux 라이브러리 사용가능"},
20     {name : "Node.JS", detail : "Node.js를 활용한 웹서버에대한 지식 함양 및, 유연한 MongoDB를 이용한 DB설계로 웹서비스 프로토타입 제작가능"},
21     {name : "HTML , CSS , SASS", detail : "HTML5 DTD 문법 준수및 웹 접근성 향상, 반응형 CSS작성 및 SASS프로그램을 활용한 CSS작성 가능"},
22     {name : "JavaScript , JQuery", detail : ["jQuery 플러그인 활용 및 구현", "외부 API 구현(Google Map, SwiperJS)", "EcmaScript 2022 기반 JSP 구현"]},
23     {name : "JSP", detail : "Java 동적 웹 애플리케이션 구현 및 기본 include와 제어문 작성 가능"},
24     {name : "GitHub", detail : "GitHub를 활용한 형상관리 및 Git 시스템을 이용한 협업 가능"},
25     {name : "Prototype Tool", detail : "Figma를 이용한 웹 기반 UI/UX 디자인 및 사용자 가이드 작성과 코딩에 적합한 디자인능력 함양"}
26   ]
27 })
28
```

## User Interface Using React

User Interface Implementation using React

4 / 5

## Start of Change!

빠르게 변화하는 환경에서 더 큰 목표를 위해  
항상 새로운 것에 도전하고 끊임 없는 변화를 시도합니다.

### 01 ReactJS

React를 활용한 사용자 HTML 동적 바인딩과 axios, Fetch API, Router, Redux 라이브러리 사용가능

### 02 Node.JS

Node.js를 활용한 웹서버에대한 지식 함양 및, 유연한 MongoDB를 이용한 DB설계로 웹서비스 프로토타입 제작가능

### 03 HTML, CSS, SASS

HTML5 DTD 문법 준수 및 웹 접근성 향상, 반응형 CSS작성 및 SASS프로그램을 활용한 CSS작성 가능

### 04 JavaScript, JQuery

JQuery 플러그인 활용 및 구현(Google Map, SwiperJS) EcmaScript 2022 기반 Native JavaScript 구현 가능

### 05 JSP

Java 동적 웹 애플리케이션 구현 및 기본 include와 제어문 작성 가능

### 06 GitHub

GitHub를 활용한 형상관리 및 Git 시스템을 이용한 협업 가능

### 07 Prototype Tool

Figma를 이용한 웹 기반 UI/UX 디자인 및 사용자 가이드, 책상과 코딩에 적합한 디자인능력 함양