13 장바구니(Cart) 페이지 제작 – 함수와 변수를 아무데서나 불러서 사용하는 Redux (@reduxjs/toolkit 최신버전) props 대신 reducer로 사용

<table>
<tr><th>순서</th><th>작업환경</th><th colspan="3">명령어</th></tr>
<tr><td>1</td><td>터미널에서
Redux 설치</td><td colspan="2">npm install @reduxjs/toolkit react-redux</td><td>@reduxjs/toolkit 두 개의 라이브러리를 한 번에 설치
toolkit은 Redux를 더 쉽게 쓰게 해줌 (둘 다 세트로 쓰는 게 기본)</td></tr>
<tr><td>2</td><td>index.js 파일에
사용한다고 설정하기</td><td colspan="3">
<table>
<tr><td>index.js
상단</td><td colspan="2">import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './store'</td></tr>
<tr><td>render(
안에
)</td><td>// <React.StrictMode>
<Provider store= {store}>
<BrowserRouter>
<App />
</BrowserRouter>
</Provider>
// </React.StrictMode></td><td>const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
// <React.StrictMode>
<Provider store={store}>
<BrowserRouter>
<App />
</BrowserRouter>
</Provider>
// </React.StrictMode>
);</td></tr>
<tr><td>전체코드</td><td colspan="2">import React from "react";
import ReactDOM from "react-dom/client";
import "./index.css";
import App from "./App";
import reportWebVitals from "./reportWebVitals";
import { BrowserRouter } from "react-router-dom";
import { Provider } from "react-redux";
import store from "./store";

const root = ReactDOM.createRoot(document.getElementById("root"));
root.render(
// <React.StrictMode>
<Provider store={store}>
<BrowserRouter>
<App />
</BrowserRouter>
</Provider>
// </React.StrictMode>
);
reportWebVitals();</td></tr>
</table>
</td></tr>
</table>

store.js - 리덕스 파일 만들기 (순수 자바스크립트 코드) - 앱 상태를 중앙에서 관리

순서	작업환경	명령어	
3	src폴더에서 우클릭 새파일 만들기 **store.js** 리듀서사용해서 변수랑 함수 만들기	상단	import { configureStore, createSlice } from '@reduxjs/toolkit'

		상단
		바로아래
			// 'user'라는 이름의 상태 만들기
			let **user** = createSlice({
			name : 'user',
			initialState : { name : '홍길동', age : 20 }, // 처음 상태 값
			reducers :{
			changeName(state){ // 이름을 '손오공'으로 바꾸는 함수
			state.name = '손오공'
			},
			
			// 나이를 전달받은 숫자만큼 올려주는 함수
			increase(state, action){
			state.age += action.payload
			}
			}
			})
			
			// 위에서 만든 함수들 밖에서도 쓸 수 있게 내보내기
			export let { **changeName, increase** } = user.actions

		위의 코드
		바로 아래
			// 'cart'라는 이름의 상태 만들기 (장바구니)
			let **cart** = createSlice({
			name : 'cart',
			initialState : [
			{id : 1, imgurl:'fruit1.jpg', name : '수박', count : 2},
			{id : 2, imgurl:'fruit2.jpg', name : '참외', count : 1},
			{id : 3, imgurl:'fruit3.jpg', name : '사과', count : 1}
			],
			reducers : {
			// 상품 수량 1개 늘리기
			addCount(state, action) {
			let num = state.findIndex((a) => {
			return a.id === action.payload;

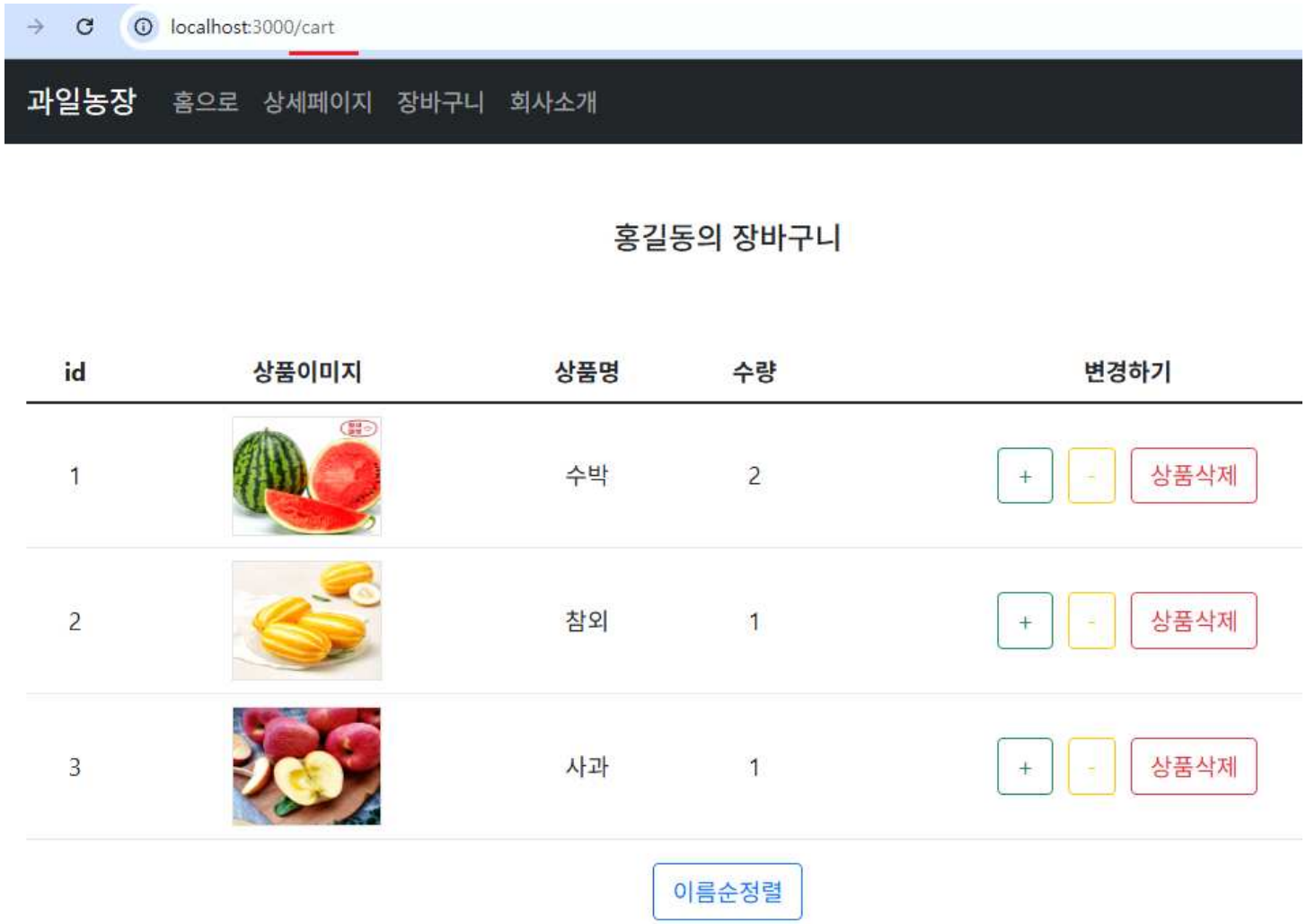
```
    });
    console.log(num);
    console.log("내가 선택한 상품"+ action.payload);
    console.log("내가 추가한 상품아이디는"+ state[num].id);
    console.log("내가 추가한 상품갯수는"+ state[num].count);

    state[num].count++;
  },
    // 상품 수량 1개 줄이기 (0개 이하는 알림 띄움)
  decreaseCount(state, action) {
    let num = state.findIndex((a) => {
      return a.id === action.payload;
    });
    console.log(num);
    if (state[num].count > 0) {
      state[num].count--;
    } else if (state[num].count === 0) {
      alert("상품이 더 이상 없습니다.");
    }
  },
   // 장바구니에 상품 추가하기, 이미 있으면 수량만 +1, 없으면 새로 추가
  addItem(state, action) {
    let num = state.findIndex((a) => a.id === action.payload.id);
    if (num !== -1) {
      state[num].count++;
    } else {
      state.push(action.payload);
    }
  },
// 장바구니에서 상품 삭제하기
  deleteItem(state, action) {
    let num = state.findIndex((a) => {
      return a.id === action.payload;
    });
    state.splice(num, 1);
  },
    // 이름순으로 상품 정렬하기
  sortName(state, action) {
    state.sort((a, b) => (a.name > b.name ? 1 : -1));
```

			}
			}
			})
			
			// cart 함수들도 밖에서 쓸 수 있게 내보내기
			export let { **addCount, decreaseCount, addItem, deleteItem, sortName** } = cart.actions;

		위의 코드
		바로 아래
			// 실제로 Redux에 등록해주는 부분
			export default configureStore({
			reducer: {
			user: user.reducer,
			cart: cart.reducer,
			},
			});
			
			/*
			createSlice: 상태랑 관련 함수들 한 번에 만들기
			reducers: 상태를 바꿔주는 함수들
			action.payload: 함수에 보낼 값
			configureStore: 만든 상태들을 Redux에 등록하는 역할
			*/

Cart.js 장바구니 페이지 만들기

구현 화면



순서	작업환경	명령어	
4	src폴더 > components > **Cart.js** 만들기	상단	import { Table } from "react-bootstrap"; import { useDispatch, useSelector } from "react-redux"; // import store, {changeName, increase} from './store.js' import { addCount, decreaseCount, deleteItem, sortName } from "../store.js"; import { Button } from "react-bootstrap"; import { Link } from "react-router-dom";

		상단
		바로 아래
			function Cart() {
			
			// Redux에서 전체 state를 구조분해할당으로 가져옴
			const { user: { name, age }, cart} = useSelector((state) => state);
			
			// dispatch는 store.js 로 요청보내주는 함수
			let dispatch = useDispatch();
			
			const smallProdcuctStyle = {
			border: "1px solid #ddd",
			width: "100px",
			height: "80px",
			cursor: "pointer",
			};
			
			let textverticalAlign = {
			verticalAlign: "middle",
			textAlign: "center",
			};
			
			return (
			<>
			
			<div class="col-sm-12" style={{ textAlign: "center" }}>
			{/* 사용자 이름과 나이 보여주기 */}
			<h5 style={{ padding: "50px" }}>
			{/* {name} {age}의 장바구니/ */}
			{name}의 장바구니

```
</h5>

<Table>
  <thead>
    <tr>
      <th>id</th>
      <th>상품이미지</th>
      <th>상품명</th>
      <th>수량</th>
      <th>변경하기</th>
    </tr>
  </thead>
  <tbody>
    {/* 장바구니에 있는 상품 목록 출력 */}
    {cart.map(({ id, imgurl, name, count }, i) => (
      <tr key={i}>
        <td style={textverticalAlign}>{id}</td>
        {/* 이미지 클릭 시 해당 상품 상세 페이지로 이동 */}
        <td>
          <Link to={`/detail/${id}`}>
            <img
              src={`img/${imgurl}`}
              style={smallProdcuctStyle}
            />
          </Link>
        </td>
        {/* 상품명, 수량 보여주기 */}
        <td style={textverticalAlign}>{name}</td>
        <td style={textverticalAlign}>{count}</td>

        <td style={textverticalAlign}>
          {/* 상품수량  + 버튼 */}
          <Button
            onClick={() => {
              dispatch(addCount(id));
            }}
            variant="outline-success"
            style={{ marginRight: "10px" }}
          >
```

```
            +
          </Button>

          {/* 상품수량  - 버튼 */}
          <Button
            onClick={() => {
              dispatch(decreaseCount(id));
            }}
            variant="outline-warning"
            style={{ marginRight: "10px" }}
          >
            -
          </Button>

          {/* 상품 삭제 버튼 */}
          <Button
            onClick={() => {
              dispatch(deleteItem(id));
            }}
            variant="outline-danger"
          >
            상품삭제
          </Button>

        </td>
      </tr>
    ))}
  </tbody>
</Table>

{/* 이름순으로 정렬하는 버튼 */}
<Button
  variant="outline-primary"
  onClick={() => {
    dispatch(sortName());
  }}
>
  이름순정렬
</Button>{" "}
```

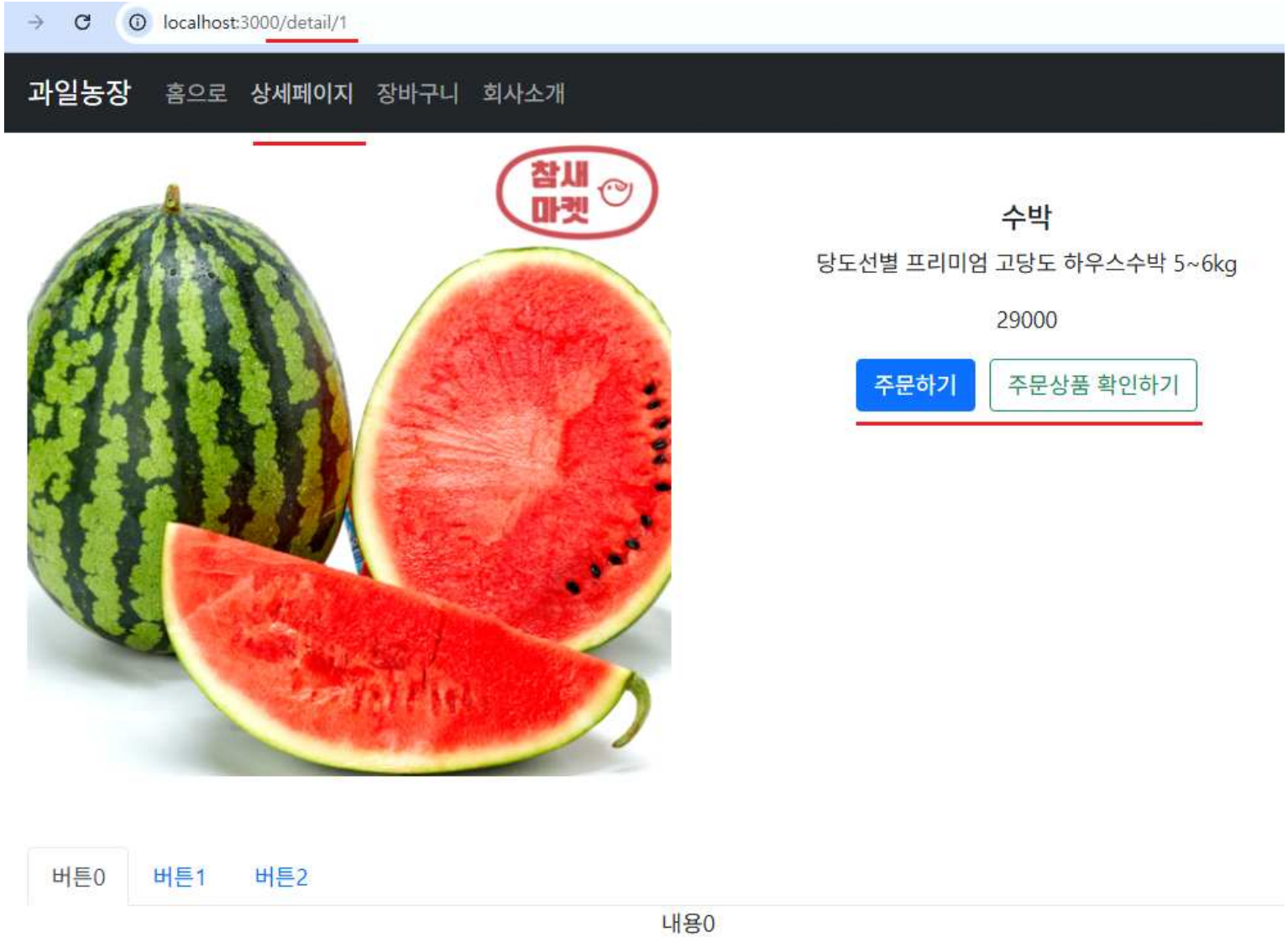
```
            </div>
          </div>
        </div>
      </>
    );
}

export default Cart;
```

Cart.js 전체 코드

```
import { Table } from "react-bootstrap";
import { useDispatch, useSelector } from "react-redux";
// import store, {changeName, increase} from './store.js'
import { addCount, decreaseCount, deleteItem, sortName } from "../store.js";
import { Button } from "react-bootstrap";
import { Link } from "react-router-dom";



function Cart() {

    // Redux에서 전체 state를 구조분해할당으로 가져옴
    const { user: { name, age }, cart} = useSelector((state) => state);

    // dispatch는  store.js 로 요청보내주는 함수
    let dispatch = useDispatch();

    const smallProdcuctStyle = {
      border: "1px solid #ddd",
      width: "100px",
      height: "80px",
      cursor: "pointer",
    };

    let textverticalAlign = {
      verticalAlign: "middle",
      textAlign: "center",
    };

    return (
      <>
```

```
<div class="container">
  <div class="row">
    <div class="col-sm-12" style={{ textAlign: "center" }}>
      {/* 사용자 이름과 나이 보여주기 */}
      <h5 style={{ padding: "50px" }}>
        {/* {name} {age}의 장바구니/ */}
        {name}의 장바구니
      </h5>

      <Table>
        <thead>
          <tr>
            <th>id</th>
            <th>상품이미지</th>
            <th>상품명</th>
            <th>수량</th>
            <th>변경하기</th>
          </tr>
        </thead>
        <tbody>
          {/* 장바구니에 있는 상품 목록 출력 */}
          {cart.map(({ id, imgurl, name, count }, i) => (
            <tr key={i}>
              <td style={textverticalAlign}>{id}</td>
              {/* 이미지 클릭 시 해당 상품 상세 페이지로 이동 */}
              <td>
                <Link to={`/detail/${id}`}>
                  <img  src={`img/${imgurl}`} style={smallProdcuctStyle} />
                </Link>

              </td>
              {/* 상품명, 수량 보여주기 */}
              <td style={textverticalAlign}>{name}</td>
              <td style={textverticalAlign}>{count}</td>

              <td style={textverticalAlign}>
                {/* 상품수량  + 버튼 */}
                <Button onClick={() => { dispatch(addCount(id)); }} variant="outline-success" style={{ marginRight: "10px" }}> + </Button>
```

```
                        {/* 상품수량  - 버튼 */}
                        <Button onClick={() => { dispatch(decreaseCount(id)); }} variant="outline-warning" style={{ marginRight: "10px" }} > - </Button>

                        {/* 상품 삭제 버튼 */}
                        <Button onClick={() => { dispatch(deleteItem(id)); }} variant="outline-danger" > 상품삭제 </Button>
                      </td>
                    </tr>
                  ))}
                </tbody>
              </Table>


              {/* 이름순으로 정렬하는 버튼 */}
              <Button variant="outline-primary" onClick={() => { dispatch(sortName()); }} > 이름순정렬 </Button>{" "}
            </div>
          </div>
        </div>
      </>
    );
}

export default Cart;
```

Detatil. js 상세페이지에서 주문하기와 주문상품확인하기(장바구니) 코드 추가	
구현 화면	

<table>
<tr><th>순서</th><th>작업환경</th><th colspan="2">명령어</th></tr>
<tr><td rowspan="2">5</td><td rowspan="2">Detail.js</td><td>상단 import 추가
빨간색 코드 추가</td><td>import React from 'react';
import { useParams } from "react-router-dom";
import { Nav } from 'react-bootstrap'
import { useState, useEffect } from 'react'; // 한 줄로 합침
import { addItem } from '../store.js'
import { useDispatch } from 'react-redux'
import { Button } from 'react-bootstrap'
import { Link } from 'react-router-dom';</td></tr>
<tr><td>return 위</td><td>let { paramId } = useParams();// paramId는 문자열로 오기 때문에 숫자로 바꿔줘야 한다.

// const { imgUrl, title, content, price } = props.fruit[paramId];
//let item = props.fruit.find(f => f.id === parseInt(paramId)); // 조건에 맞는 요소(객체)를 찾아 반환
//const { imgUrl, title, content, price } = item;

let [tap, setTap] = useState(0);

// React Hook은 조건 없이 컴포넌트 최상단에서 호출되어야 함
let [fade2, setFade2] = useState('');

// dispatch 정의 추가
const dispatch = useDispatch();

useEffect(()=>{
setFade2('end')
return ()=>{
setFade2('')
}
},[])

// 상품 유효성 체크 (이건 Hook 호출 이후에 실행되어야 함)
let selproduct = props.fruit.find((x) => x.id === Number(paramId));</td></tr>
</table>

			// 혹은 조건문(if) 아래에서 호출하면 안 됨 (React가 Hook 순서 기억을 못함)
			if (!selproduct) {
			return 해당 상품이 존재하지 않습니다. ;
			}
			
			const { id, imgUrl, title, content, price } = selproduct;
			
			console.log("내가 선택한 상품은: " + id + " " + title);

		빨간색으로 코드수정
		
		<button className="btn btn-danger">주문하기</button>
		
		주문하기 버튼을
		옆의 빨간색 코드로 대체
			return (
			<div className={"container start " + fade2}>
			<div className="row" style={{ marginBottom: "50px" }}>
			
			
			
			<h4 className="pt-5">{title}</h4>
			 {content}
			 {price}
			<Button
			variant="primary"
			onClick={() => {
			// dispatch(addItem({id : 1, imgurl : 'fruit1.jpg', name : 'Grey Yordan', count : 1}))
			
			dispatch(
			addItem({
			id: id,
			imgurl: imgUrl.replace("img/", ""),
			name: title,
			count: 1,
			})
			);
			}}
			style={{ marginRight: "10px" }}
			>
			주문하기
			</Button>

			<Link to="/cart"> <Button variant="outline-success"> 주문상품 확인하기 </Button> </Link> <Nav variant="tabs" defaultActiveKey="link0"> <Nav.Item> <Nav.Link eventKey="link0" onClick={() => { setTap(0); }}>버튼0</Nav.Link> </Nav.Item> <Nav.Item> <Nav.Link eventKey="link1" onClick={() => { setTap(1); }}>버튼1</Nav.Link> </Nav.Item> <Nav.Item> <Nav.Link eventKey="link2" onClick={() => { setTap(2); }}>버튼2</Nav.Link> </Nav.Item> </Nav> <TabContent tap={tap} />); }

	Detail.js
	전체 코드(참고)
			import React from 'react';
			import { useParams } from "react-router-dom";
			import { Nav } from 'react-bootstrap'
			import { useState, useEffect } from 'react'; // 한 줄로 합침
			import { addItem } from '../store.js'
			import { useDispatch } from 'react-redux'
			import { Button } from 'react-bootstrap'
			import { Link } from 'react-router-dom';
			
			function Detail(props) {
			let { paramId } = useParams();// paramId는 문자열로 오기 때문에 숫자로 바꿔줘야 한다.
			let [tap, setTap] = useState(0);

```
// React Hook은 조건 없이 컴포넌트 최상단에서 호출되어야 함
let [fade2, setFade2] = useState('');

// dispatch 정의 추가
const dispatch = useDispatch();



useEffect(()=>{
  setFade2('end')
  return ()=>{
    setFade2('')
  }
},[])

// 상품 유효성 체크 (이건 Hook 호출 이후에 실행되어야 함)

let selproduct = props.fruit.find((x) => x.id === Number(paramId));

// 혹은 조건문(if) 아래에서 호출하면 안 됨 (React가 Hook 순서 기억을 못함)
if (!selproduct) {
 return <div>해당 상품이 존재하지 않습니다.</div>;
}

const { id, imgUrl, title, content, price } = selproduct;

console.log("내가 선택한 상품은: " + id + " " + title);



return (
    <div className={'container start ' + fade2}>
        <div className="row">
            <div className="col-md-6">
              <img src={'/' + imgUrl} width="100%"  alt={title} />
            </div>
            <div className="col-md-6">
                <h4 className="pt-5">{title}</h4>
                <p>{content}</p>
```

```
                    <p>{price}</p>
                    <Button
                    variant="primary"
                    onClick={() => {
                      //   dispatch(addItem(   {id : 2,   imgurl : 'shoes1.jpg', name : 'Grey Yordan', count : 1}))

                      dispatch(
                        addItem({
                          id: id,
                          imgurl: imgUrl.replace("img/", ""),
                          name: title,
                          count: 1,
                        })
                      );
                    }}
                    style={{ marginRight: "10px" }}
                  >
                    주문하기
                  </Button>

<Link to="/cart">
  <Button variant="outline-success"> 주문상품 확인하기 </Button>
</Link>


        </div>
  </div>

  <Nav variant="tabs"   defaultActiveKey="link0" style={{marginTop:"50px"}}>
        <Nav.Item>
        <Nav.Link   onClick={()=>{ setTap(0) }} eventKey="link0">버튼0</Nav.Link>
        </Nav.Item>
        <Nav.Item>
        <Nav.Link onClick={()=>{ setTap(1) }} eventKey="link1">버튼1</Nav.Link>
        </Nav.Item>
        <Nav.Item>
        <Nav.Link   onClick={()=>{ setTap(2) }} eventKey="link2">버튼2</Nav.Link>
        </Nav.Item>
  </Nav>
```

```
                <TabContent tap={tap}/>
            </div>
        )
}


function TabContent({tap}){
    let [fade, setFade] = useState('')
     useEffect( ()=>{
       setTimeout(()=>{ setFade('end')},10)
       return ()=>{
            setFade('')
       }
     } ,[tap])

    return (
      <div className={'start ' + fade}>
        { [<div>내용0</div>, <div>내용1</div>, <div>내용2</div>][tap] }
      </div>
    )
  }

export default Detail;
```

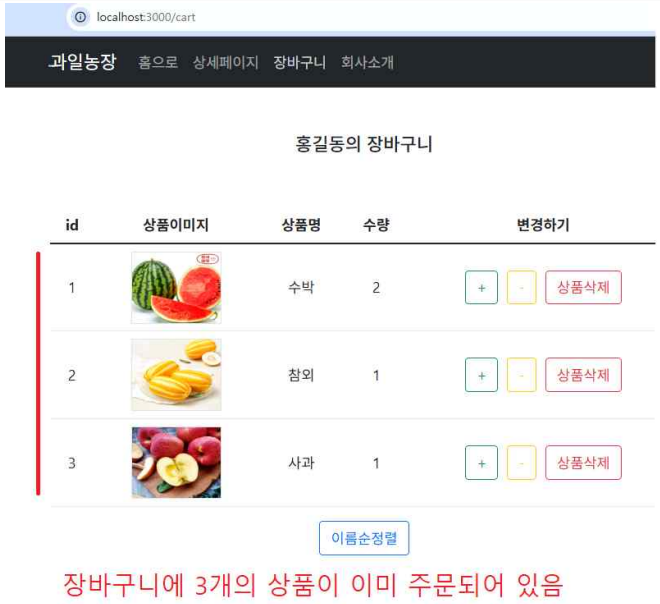
App.js에 장바구니 페이지 추가하기

<table>
<tr><th>순서</th><th>작업환경</th><th>명령어</th></tr>
<tr><td>6</td><td>App.js에서</td><td>
<table>
<tr><td>상단 import 확인</td><td>import Cart from "./components/Cart";</td></tr>
<tr><td>Cart 라우터 확인</td><td><Route path="/detail/:paramId" element={<Detail fruit={fruit} />} /> //아래

<Route path="/cart" element={<Cart/>}/></td></tr>
</table>
</td></tr>
<tr><td>7</td><td>브라우저에서 확인</td><td>상품 선택후 상세 페이지에서 주문하기 버튼을 누른후 주문상품 확인하기 버튼 클릭
장바구니 사이트로 가서 상품이 추가 주문되었는지 확인</td></tr>
</table>

브라우저에서 확인

메인화면에서 장바구니 클릭
장바구니에 이미 3개의 상품이 들어있는지 확인



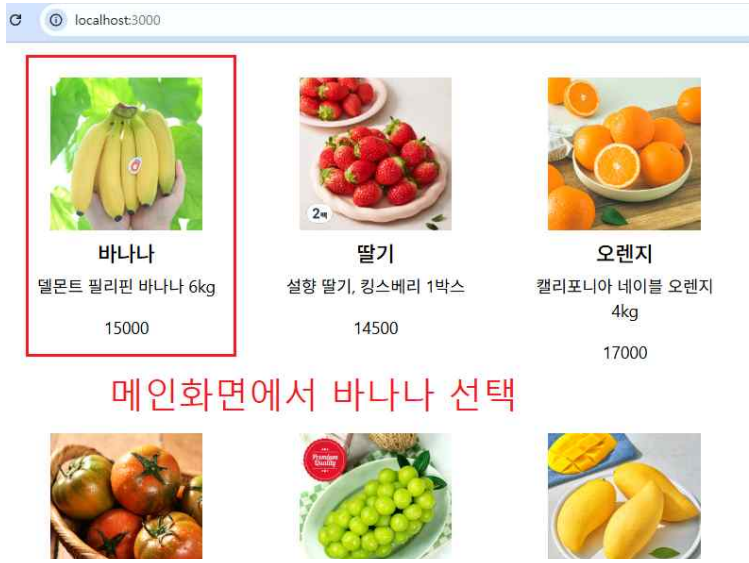
메인 화면으로 돌아와서 수박상품을 클릭

수박상품에서 주문하기 버튼클릭

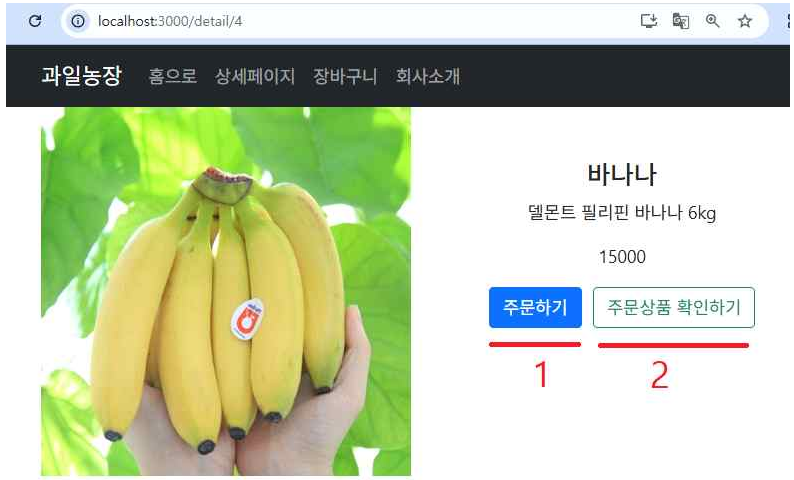
주문하기 버튼 클릭후 주문상품 확인하기 버튼 클릭

수박 상품 수량이 2 + 1 = 3인지 확인

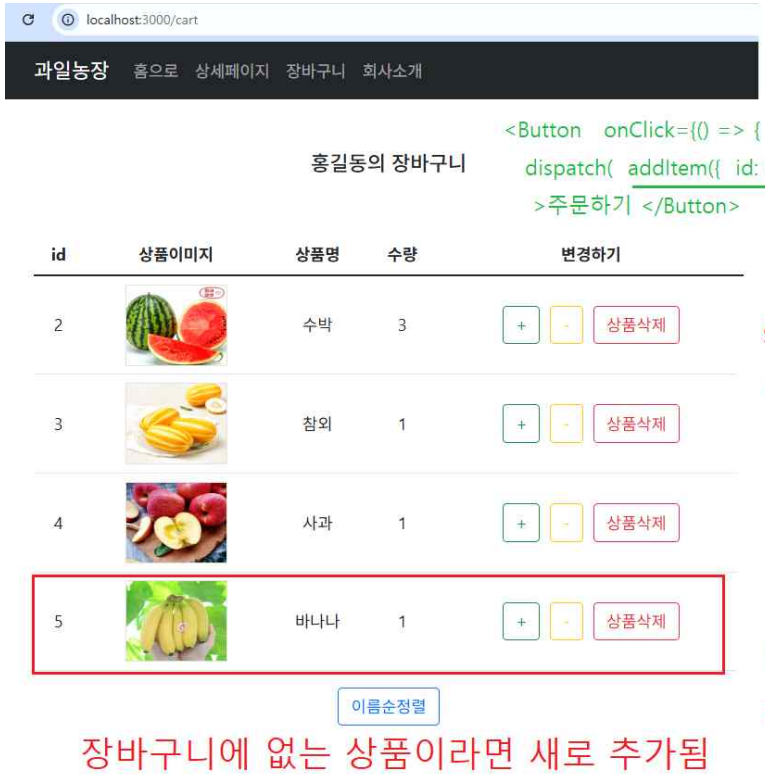
메인화면으로 다시 돌아가서 장바구니에 없는 바나나 상품 선택



바나나 상세페이지로 가서 1 주문하기 버튼클릭, 2 주문상품 확인하기 버튼클릭



바나나가 장바구니에 추가되었는지 확인	다른 버튼도 클릭했을 때 잘 작동하는지 확인



장바구니에 없는 상품이라면 새로 추가됨

detail.js 상품상세페이지 (인자값만 줄수있다)

```
<Button  onClick={() => {
  dispatch( addItem({  id: id,   imgurl: imgUrl.replace("img/", ""),   name: title, count: 1, })); }}
  >주문하기 </Button>
```

store.js(리덕스파일) - 파라미터만 줄수있다

```
addItem(state, action) {
  let num = state.findIndex((a) => a.id ===  action.payload.id);
  if (num !== -1) {
    state[num].count++;
  } else {
    state.push(action.payload);
  }
},
```

장바구니에 상품 추가하기, 이미 있으면 수량만 +1, 없으면 로 추가

리듀서 문법 이해 (코드분석)

자바 스크립트 객체

```
let user = {
  name: '홍길동',
  age: 20,

  // 이름을 '손오공'으로 바꾸는 메서드
  changeName: function() {
    this.name = '손오공';
  },

  // 나이를 전달받은 값만큼 증가시키는 메서드
  increaseAge: function(value) {
    this.age += value;
  }
};
```

자바 객체

```
public class User {
    String name = "홍길동";
    int age = 20;

    void changeName() {
        this.name = "손오공";
    }

    void increase(int value) {
        this.age += value;
    }
}
```

name	슬라이스의 이름, 라벨
initialState:	처음 시작할 때 저장된 값 (기본값)
reducers	데이터를 변경하는 함수들

createSlice	상태와 그 상태를 바꾸는 액션 생성 함수들을 정의
configureStore	여러 리듀서를 결합하여 Redux store를 설정

Redux Toolkit 버전

```
import { configureStore, createSlice } from '@reduxjs/toolkit';
let user = createSlice({
  name: 'user',  // 슬라이스 이름, 라벨
  initialState: { name: '홍길동', age: 20 },  // 데이터 초기값 저장(기본값)
  reducers: {   // 안에 함수 모아두기, 데이터를 변경하는 함수들
    changeName(state) { //상태를 매개변수로 받음
      state.name = '손오공';
    },
    increase(state, action) {  //변수, 함수
      state.age += action.payload;
    }
  }
});
//  user 슬라이스에서 액션 함수들을 각각 가져오기
//  다른 파일에서 액션을 디스패치(dispatch) 할 때 사용됨
export let { changeName, increase } = user.actions;

// 'user' 슬라이스 전체를 다른 곳에서 사용할 수 있도록 내보내기
export default user;

// Redux store 등록
const store = configureStore({
  reducer: {  // user라는 상태를 담당하는 리듀서를 등록
    user: userReducer,  // userReducer는 user상태를 관리하는 리듀서
  },
});

export default store;   // 설정한 store를 내보내서 다른 곳에서 사용할 수 있도록 함
```

리덕스를 사용하는 코드 쪽에서 Detail.js(상세페이지), Cart.js(장바구니)

```
// App.js
import { useDispatch, useSelector } from 'react-redux';
import store, { changeName, increase } from './store.js';

// 컴포넌트 안에서 사용
const dispatch = useDispatch();
const user = useSelector((state) => state.user);

// 버튼 클릭 시 이름 변경
<button onClick={() => dispatch(changeName())}>이름 바꾸기</button>
```

import { useDispatch, **useSelector** } from "react-redux"; //리덕스 사용
import **store**, {changeName, increase} from './store.js' // **Provider연결**, 함수사용,	

useDispatch	Redux에 액션을 보내는 함수
useSelector	Redux 상태를 읽어오는 **변수**

비유	
store	앱 전체 데이터(state)를 보관하는 곳(장바구니)
store	커다란 창고
state	창고 안에 들어 있는 물건들
dispatch	물건을 바꿔달라고 요청하는 것
reducer	요청을 받아서 실제로 물건을 바꿔주는 작업자

Redux Toolkit 기본 문법 정리
createSlice({ name, initialState, reducers })
initialState
reducers
state
action.payload
export { 액션함수명 } = slice.actions
configureStore({ reducer: { ... } })
slice.reducer

14 Styled-Components로 CSS 파일 없이 스타일 관리

<table>
<tr><th>순서</th><th>작업환경</th><th colspan="2">명령어</th></tr>
<tr><td>1</td><td>Terminal</td><td colspan="2">종료 후 명령어로 설치
npm install styled-components

위의 코드로 안깔리면 최신버전으로 다시 깔기 styled-components의 최신버전이라는 뜻
npm install styled-components@latest</td></tr>
<tr><td rowspan="2">2</td><td rowspan="2">Detail.js

`` backtick 기호를 이용해서
CSS 스타일

대문자로 만들어야 됨</td><td>상단</td><td>Detail함수 안에</td></tr>
<tr><td>import styled from 'styled-components'

let Banner = styled.div`<br>padding : 20px;<br>color : gray;<br>`;

let BannerBtn = styled.button`<br>color : white;<br>font-size:30px;<br>width:100%;<br>padding : 100px 100px;<br>border:1px solid #ccc;<br>background-image:url("../img/banner.jpg");<br>background-size:cover;<br>background-position:center;<br>`;</td><td>function Detail(props) {

return (
<div className={'container start ' + fade2}>

<Banner>
<BannerBtn>과일농장의 맛과 건강을 선물하세요.</BannerBtn>
</Banner>

)
}</td></tr>
<tr><td></td><td>styled로 만든 컴포넌트 사용 예</td><td>.import styled from 'styled-components';

const Wrapper = styled.div`<br>padding: 20px;<br>background-color: lightgray;<br>`;</td><td>왼쪽 코드처럼 Wrapper처럼 styled로 만든
컴포넌트 이름은 반드시 대문자로 시작해야 됨
소문로 시작하면 리액트가 HTML 태그라고 본다.</td></tr>
</table>

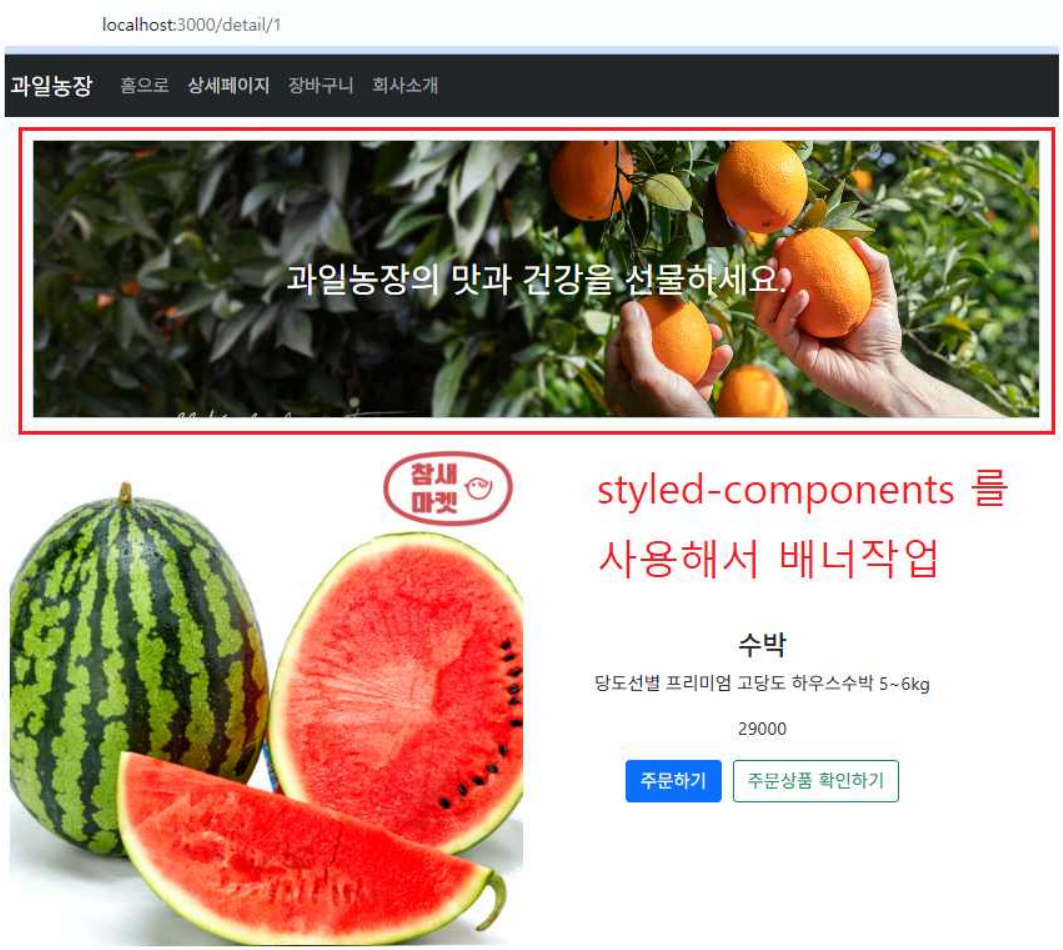
	전체 코드	import React from 'react'; import { useParams } from "react-router-dom"; import { Nav } from 'react-bootstrap' import { useState, useEffect } from 'react'; // 한 줄로 합침 import { addItem } from '../store.js' import { useDispatch } from 'react-redux' import { Button } from 'react-bootstrap' import { Link } from 'react-router-dom'; import styled from 'styled-components' let Banner = styled.div` padding : 20px; color : gray; `; let BannerBtn = styled.button` color : white; font-size:30px; width:100%; padding : 100px 100px; border:1px solid #ccc; background-image:url("../img/banner.jpg"); background-size:cover; background-position:center; `; function Detail(props) { let { paramId } = useParams();// paramId는 문자열로 오기 때문에 숫자로 바꿔줘야 한다. let [tap, setTap] = useState(0); // React Hook은 조건 없이 컴포넌트 최상단에서 호출되어야 함 let [fade2, setFade2] = useState(''); // dispatch 정의 추가 const dispatch = useDispatch();

```
useEffect(()=>{
   setFade2('end')
   return ()=>{
      setFade2('')
   }
},[])

// 상품 유효성 체크 (이건 Hook 호출 이후에 실행되어야 함)

let selproduct = props.fruit.find((x) => x.id === Number(paramId));

// 혹은 조건문(if) 아래에서 호출하면 안 됨 (React가 Hook 순서 기억을 못함)
if (!selproduct) {
 return <div>해당 상품이 존재하지 않습니다.</div>;
}

const { id, imgUrl, title, content, price } = selproduct;

console.log("내가 선택한 상품은: " + id + " " + title);



return (
     <div className={'container start ' + fade2}>

          <Banner>
              <BannerBtn>과일농장의 맛과 건강을 선물하세요.</BannerBtn>
          </Banner>

          <div className="row">
              <div className="col-md-6">
                <img src={'/' + imgUrl} width="100%"  alt={title} />
              </div>
              <div className="col-md-6">
                   <h4 className="pt-5">{title}</h4>
                   <p>{content}</p>
                   <p>{price}</p>
                   <Button
```

```
                variant="primary"
                onClick={() => {
                  //   dispatch(addItem(   {id : 1,   imgurl : 'fruit1.jpg', name : 'Grey Yordan', count : 1}))

                  dispatch(
                    addItem({
                      id: id,
                      imgurl: imgUrl.replace("img/", ""),
                      name: title,
                      count: 1,
                    })
                  );
                }}
                style={{ marginRight: "10px" }}
              >
                주문하기
              </Button>

<Link to="/cart">
  <Button variant="outline-success"> 주문상품 확인하기 </Button>
</Link>


        </div>
  </div>

  <Nav variant="tabs"   defaultActiveKey="link0" style={{marginTop:"50px"}}>
        <Nav.Item>
        <Nav.Link   onClick={()=>{ setTap(0) }} eventKey="link0">버튼0</Nav.Link>
        </Nav.Item>
        <Nav.Item>
        <Nav.Link onClick={()=>{ setTap(1) }} eventKey="link1">버튼1</Nav.Link>
        </Nav.Item>
        <Nav.Item>
        <Nav.Link   onClick={()=>{ setTap(2) }} eventKey="link2">버튼2</Nav.Link>
        </Nav.Item>
  </Nav>
```

```
                <TabContent tap={tap}/>
            </div>
        )
}


function TabContent({tap}){
    let [fade, setFade] = useState('')
     useEffect( ()=>{
       setTimeout(()=>{ setFade('end')},10)
       return ()=>{
            setFade('')
       }
     } ,[tap])

    return (
       <div className={'start ' + fade}>
         { [<div>내용0</div>, <div>내용1</div>, <div>내용2</div>][tap] }
       </div>
    )
  }

export default Detail;
```

브라우저에서 확인



컴포넌트 이름은 반드시 대문자로 시작해야 됨
소문로 시작하면 리액트가 HTML 태그라고 본다.

styled-components의 장점과 단점

<table>
<tr><td rowspan="2">styled-components의
장점과 단점</td><td>장점</td><td>단점</td></tr>
<tr><td>1 CSS 따로 안 써도 됨
JS 안에서 스타일 바로 씀

2 스타일 충돌 없음
다른 컴포넌트랑 섞이지 않음

3 필요한 스타일만 로딩됨
속도 조금 더 빠름</td><td>1 JS 파일이 복잡해짐
코드 보기 어려워짐

2 스타일 재사용 불편할 수 있음
여러 곳에 쓸 땐 import 해야 함

3 디자이너랑 협업 어려움
디자이너가 문법 모르면 힘듦</td></tr>
<tr><td colspan="3">개발자에겐 편한데, 협업이나 재사용은 살짝 귀찮을 수 있다</td></tr>
<tr><td>모듈화를 하면
해당js
파일에만 적용</td><td colspan="3">
<table>
<tr><td>CSS 모듈화란</td><td>CSS를 JS 파일별로 나눠서 적용되게 만드는 방법
다른 파일에 영향 안 줌! (스타일 충돌 없음)</td></tr>
<tr><td>문법</td><td>컴포넌트명.module.css</td></tr>
<tr><td>파일이름</td><td>App.js → App.module.css
Detail.js → Detail.module.css</td></tr>
<tr><td colspan="2">CSS가 해당 컴포넌트에만 적용됨
다른 파일에 간섭 안 함 (안전함)</td></tr>
</table>
</td></tr>
</table>

15 쇼핑몰을 Git에 배포 - 체크사항 (터미널에서 에러가 나면 안됨)

하위경로 없음	하위경로 배포	
//dino-21은 git id 자신의 id로 교체 https://sinaboro.github.io/	https://dino-21.github.io/shop/	
	package.json 마지막 라인에 추가	}, "homepage": "https://sinarobo.github.io/shop" }
	index.js basename 추가	<BrowserRouter basename='/shop'> <App /> </BrowserRouter>
	App.js	이미지가 안떠도 서버에 올라가면 이미지가 경로에 맞게 나옴
	상세페이지 Detail.js	
	터미널에서 확인 경로가 바뀌어도 라우터가 잘되는지	npm start 로컬호스트확인 http://localhost:3000/shop

터미널	npm run build	작업 프로젝트 폴더 내에 **build** 라는 폴더가 하나 생성됩니다. 여러분이 짰던 코드를 전부 .html .css .js 파일로 변환해서 하나의 파일로 담아줍니다. build 폴더 안에 안에 있는 내용을 모두 서버에 올리면 됩니다. index.html이 메인페이지입니다.
git New repository name	하위경로 없음: **sinaboro.github.io** 하위경로 배포: **shop**	
Repository 생성 후	build 폴더 내의 파일을 전부 드래그 앤 드롭	
Github pages 설정	source 부분을 None이 아니라 main 저장	
배포 주소 확인	하위경로 없음: **https://sinarobo.github.io/** 하위경로 배포: **https://sinaboro.github.io/shop/**	