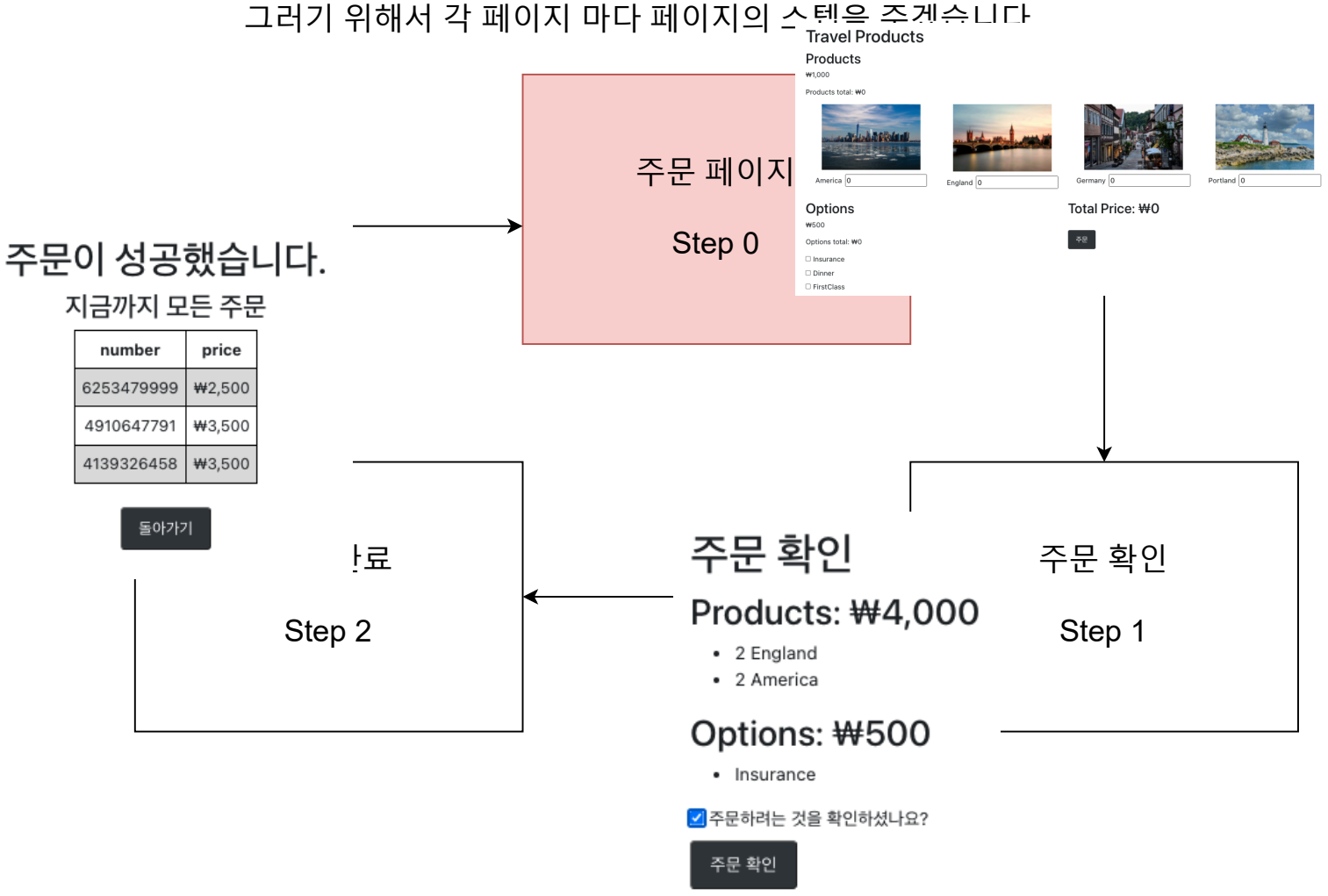
페이지마다 스텝 주기

주문 페이지에서 주문확인 페이지 그리고 주문 완료 페이지로 이동해야 합니다.

그러기 위해서 각 페이지 마다 페이지의 스텝을 주겠습니다



해야 할 일은?

주문 페이지에서 주문하고 주문 버튼 클릭 ===>
주문 확인 페이지에서 체크 박스와 버튼 클릭 ===>
주문 완료 페이지에서 완료 확인

```
test("From order to order completion", async () => {
  // App 컴포넌트 안에는 이미 provider가 Wrap 되어 있습니다.
  render(<App />);
```

테스트 작성

```
// America 여행 상품 2개 추가 합니다.
const americaInput = await screen.findByRole("spinbutton", {
  name: "America",
});
userEvent.clear(americaInput);
userEvent.type(americaInput, "2");

// England 여행 상품 3개 추가 합니다.
const englandInput = screen.getByRole("spinbutton", {
  name: "England",
});
userEvent.clear(englandInput);
userEvent.type(englandInput, "3");

// Insurance 옵션 체크
const insuranceCheckbox = await screen.findByRole("checkbox", {
  name: "Insurance",
});
userEvent.click(insuranceCheckbox);

// 모든 주문을 한 이후 주문 버튼 클릭!
const orderButton = screen.getByRole("button", {
  name: "주문하기",
});
userEvent.click(orderButton);
});
```

테스트 실행

Fail

테스트에 대응하는 실제 코드 작성

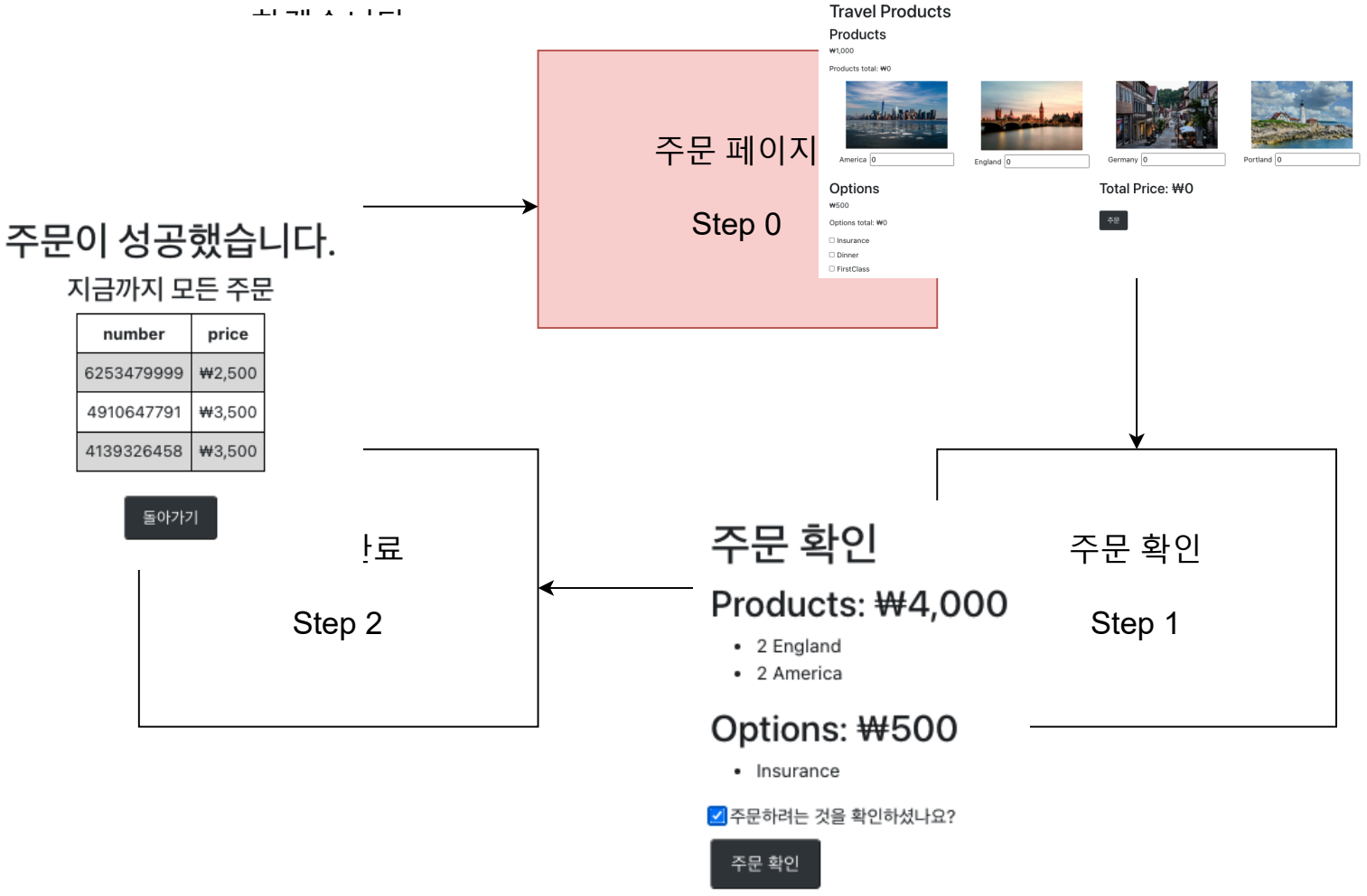
```
export default function App() {
  const [step, setStep] = useState(0);

  return (
    <div style={{ padding: "4rem" }}>
      <OrderContextProvider>
        {step === 0 && <OrderPage setStep={setStep} />}
        {step === 1 && <SummaryPage setStep={setStep} />}
        {step === 2 && <CompletePage setStep={setStep} />}
      </OrderContextProvider>
    </div>
  );
}
```

```
function OrderPage({ setStep }) {
  const [orderDatas] = useContext(OrderContext);
  return (
```

```
  return (
    <div>
      <h1>Travel Products</h1>
      <div>
        <Type orderType="products" />
      </div>
      <div style={{ display: "flex", marginTop: 20 }}>
        <div style={{ width: "50%" }}>
          <Type orderType="options" />
        </div>
        <div style={{ width: "50%" }}>
          <h2>Total Price: {orderDatas.totals.total}</h2> <br />
          <button onClick={() => setStep(1)}>주문하기</button>
        </div>
      </div>
    </div>
  );
}

export default OrderPage;
```

주문 확인 페이지

전 영상에서 주문 페이지에서 주문하기 버튼을 누르는 부분까지 구현했습니다.
이번 영상에서는 주문 확인 페이지에서 주문 확인 버튼을 누르는 부분을



해야 할 일은?

주문 페이지에서 주문하고 주문 버튼 클릭 ===>
주문 확인 페이지에서 체크 박스와 버튼 클릭 ===>
주문 완료 페이지에서 완료 확인

```
////////////////// 주문 확인 페이지 ////////////////
// 제목
const summaryHeading = screen.getByRole("heading", {
    name: "주문 확인" });
expect(summaryHeading).toBeInTheDocument();
```

테스트 작성

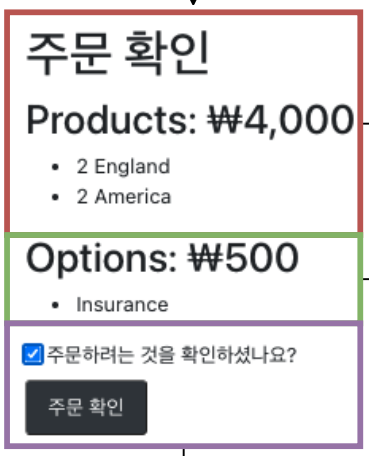
```
// 여행 상품 총 가격
const productsHeading = screen.getByRole("heading", {
  name: "여행 상품: 5000",
});
expect(productsHeading).toBeInTheDocument();

// 옵션 총 가격
const optionsHeading = screen.getByRole("heading", {
  name: "옵션: 500",
});
expect(optionsHeading).toBeInTheDocument();

// 특정 상품 나열
expect(screen.getByText("2 America")).toBeInTheDocument();
expect(screen.getByText("3 England")).toBeInTheDocument();
expect(screen.getByText("Insurance")).toBeInTheDocument();

// 체크 박스 체크
const confirmCheckbox = screen.getByRole("checkbox", {
  name: "주문하려는 것을 확인하셨나요?",
});
userEvent.click(confirmCheckbox);

const confirmOrderButton = screen.getByRole("button", {
  name: "주문 확인",
});
userEvent.click(confirmOrderButton);
```

테스트 실행

Fail

테스트에 대응하는 실제 코드 작성

```
function SummaryPage({ setStep }) {
  const [orderDetails] = useContext(OrderContext);
  const [checked, setChecked] = useState(false);

  const productArray = Array.from(orderDetails.products);
  const productList = productArray.map(([key, value]) => (
    <li key={key}>
      {value} {key}
    </li>
  ));

  return (
    <div>
      <h1>주문 확인</h1>
      <h2>여행 상품: {orderDetails.totals.products}</h2>
      <ul>{productList}</ul>
      <form >
        <input
          type="checkbox"
          checked={checked}
          onChange={(e) => setChecked(e.target.checked)}
```



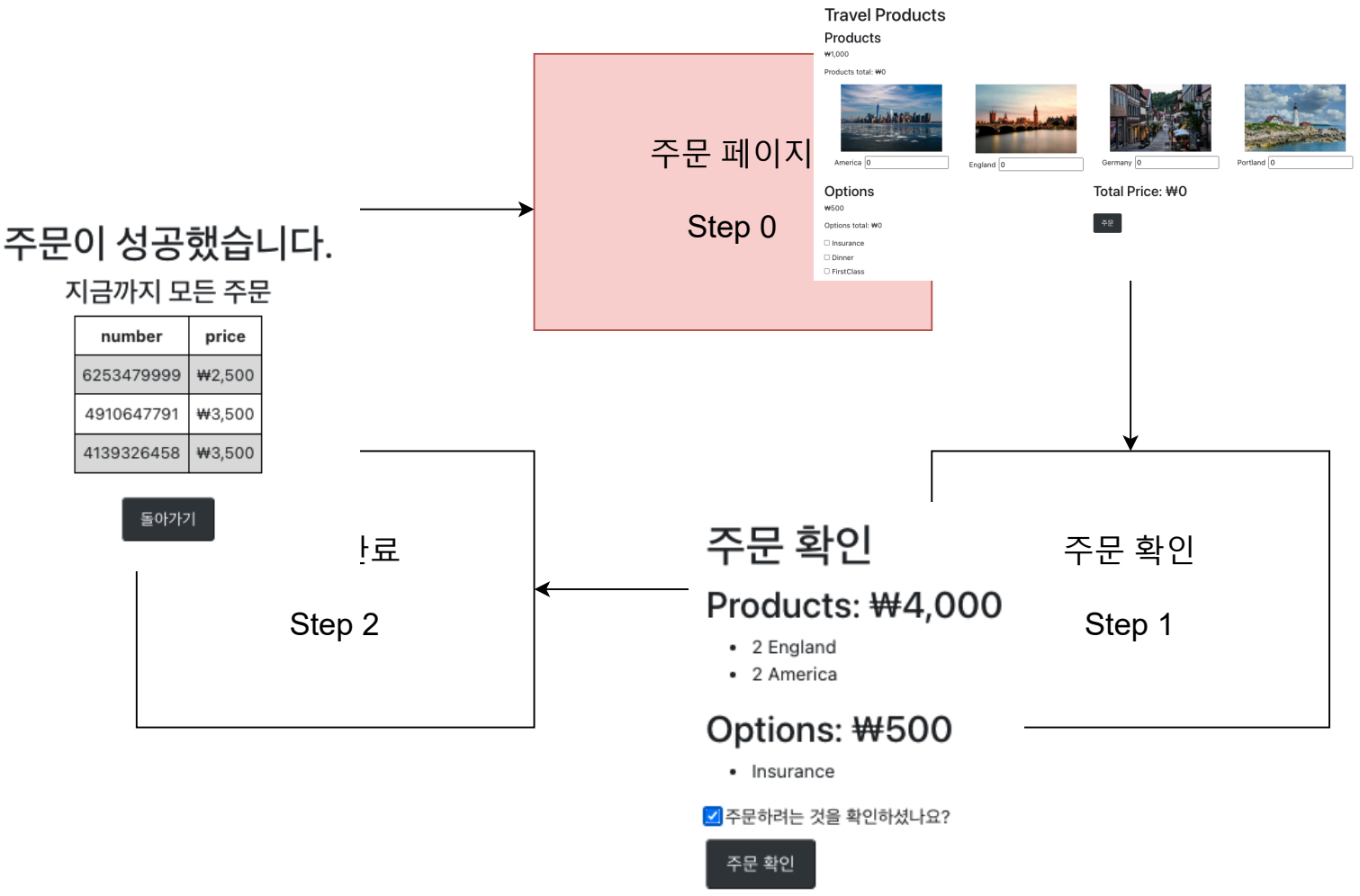

```
        id="confirm-checkbox"
      />{" "}
      <label htmlFor="confirm-checkbox">주문하려는 것을 확인하셨나요?</label>
      <br />
      <button disabled={!checked} type="submit">
        주문 확인
      </button>
    </form>
  </div>
  );
}
```

```
function SummaryPage({ setStep }) {
  const [orderDetails] = useContext(OrderContext);

  const hasOptions = orderDetails.options.size > 0;
  let optionsDisplay = null;

  if (hasOptions) {
    const optionsArray = Array.from(orderDetails.options.keys());
    const optionList = optionsArray.map((key) => <li key={key}>{key}</li>);
    optionsDisplay = (
      <>
        <h2>옵션: {orderDetails.totals.options}</h2>
        <ul>{optionList}</ul>
      </>
    );
  }
  return (
    <div>

      {optionsDisplay}

    </div>
  );
}

export default SummaryPage;
```

```
const SummaryPage = ({setStep}) => {
  const [checked, setChecked] = useState(false);
  const handleSubmit =(event) => {
    event.preventDefault();
    setStep(2)
  }

  return (
    <div>
      <form onSubmit={handleSubmit}>
        <input
          type="checkbox"
          checked={checked}
          onChange={(e) => setChecked(e.target.checked)}
          id="confirm-checkbox"
        />
        <label htmlFor="confirm-checkbox">주문하려는 것을 확인하셨나요?</label>
        <br />
        <button disabled={!checked} type="submit">
          주문 확인
        </button>
      </form>
    </div>
  );
};
```

주문 완료 페이지

마지막 페이지인 주문 완료 페이지를 구현해보겠습니다.



해야 할 일은?

주문 페이지에서 주문하고 주문 버튼 클릭 ===>
주문 확인 페이지에서 체크 박스와 버튼 클릭 ===>
주문 완료 페이지에서 완료 확인

```
////////////////// 주문 완료 페이지 ////////////////
// 백엔드에서 데이터를 가져오는 동안 loading 문구
const loading = screen.getByText(/loading/i);
expect(loading).toBeInTheDocument();

// getByRole이 아닌 findByRole을 사용하는 이유는 주문완료 페이지에 올때
```

테스트 작성

```
// post request를 보내서 async 작업이 이뤄지고 주문이 성공했습니다 문구가 나오기 때문입니다.
const completeHeader = await screen.findByRole("heading", {
  name: "주문이 성공했습니다.",
});
expect(completeHeader).toBeInTheDocument();

// 데이터를 받아온 후에 loading 문구는 사라집니다.
const loadingDisapeared = screen.queryByText("loading");
expect(loadingDisapeared).not.toBeInTheDocument();

// 첫페이지로 버튼 클릭
const firstPageButton = screen.getByRole("button", { name: "첫페이지로" });
userEvent.click(firstPageButton);
```

```
rest.post("http://localhost:5000/order", (req, res, ctx) => {
  let dummyData = [{ orderNumber: 123455676, price: 2000 }];
  return res(ctx.json(dummyData));
}),
```

테스트 실행

Fail

테스트에 대응하는 실제 코드 작성

```
useEffect(() => {
  orderCompleted(orderDatas);
}, [orderDatas]);

const orderCompleted = async (orderDatas) => {
  try {
    let response = await axios.post(
      `http://localhost:5000/order`,
      orderDatas
    );
    setOrderHistory(response.data);
    setLoading(false);
  } catch (error) {
    setError(true);
  }
};

if (error) {
  return <ErrorBanner message="에러가 발생했습니다." />;
}

if (loading) {
  return <div>loading</div>;
} else {
  return (
    <div style={{ textAlign: "center" }}>
      <h2>주문이 성공했습니다.</h2>
      <h3>지금까지 모든 주문</h3>
      <br />
      <button className="rainbow rainbow-1" onClick={() => setStep(0)}>
        첫페이지로
      </button>
    </div>
  );
}
```

```
const orderTable = orderHistory.map((item, key) => (
  <tr key={item.orderNumber}>
    <td>{item.orderNumber}</td>
    <td>{item.price}</td>
  </tr>
```

```
));

if (loading) {
  return <div>loading</div>;
} else {
  return (
    <div style={{ textAlign: "center" }}>
      <h2>주문이 성공했습니다.</h2>
      <h3>지금까지 모든 주문</h3>
      <table style={{ margin: "auto" }}>
        <tbody>
          <tr>
            <th>number</th>
            <th>price</th>
          </tr>
          {orderTable}
        </tbody>
      </table>
```

```
console.error
  Warning: An update to Type inside a test was not wrapped in act(...).

  When testing, code that causes React state updates should be wrapped into act(...):

  act(() => {
    /* fire events that update state */
  });
  /* assert on the output */
```

not wrapped in act 경고

전 영상 마지막에 이러한 경고가 났습니다. 이 경고가 무엇이고 어떻게 해결해야 하는지 알아보겠습니다.

```
console.error
  Warning: An update to Type inside a test was not wrapped in act(...).

  When testing, code that causes React state updates should be wrapped into act(...):

  act(() => {
    /* fire events that update state */
  });
  /* assert on the output */
```

Not wrapped in act 경고는 왜 나는 건가요?

리액트에서 나오는 act 경고는 우리가 컴포넌트에 아무것도 일어나지 않을 것으로 예상하고 있을 때 컴포넌트에 어떤 일이 일어나면 나오는 경고입니다.

원래 컴포넌트에서 무언가가 일어난다고 해주려면 act라는 함수로 감싸주어야 합니다.

```
act(() => {
    /* fire events that update state */
  });
```

이렇게 감싸주면 리액트는 이 컴포넌트에서 어떤 일이 일어날거라고 생

지금까지는 act로 감싸준 적이 없지 않나요?

react-testing-library 내부에 API에 act를 이미 내포하고 있어서 우리가 일부로 act로 감싸서 호출하지 않고 렌더링과 업데이트를 할 수 있습니다. (리액트 콜 스택 안에 있을 때)

```
it("should render and update a counter", () => {
    // 컴포넌트 렌더링
    act(() => {
      ReactDOM.render(<Counter />, container);
    });

    // 컴포넌트 업데이트를 트리거하는 이벤트 발생
    act(() =>
```

```
      button.dispatchEvent(new MouseEvent('click', {bubbles: true}))
    });
  })

  // With react-testing-library
it("should render and update a counter", () => {
    // 컴포넌트 렌더링
    const { getByText } = render(<Counter />);

    // 컴포넌트 업데이트를 트리거하는 이벤트 발생
    fireEvent.click(getByText("Save"));
});
```

그러면 이미 act로 감싸주었는데 지금은 왜 에러가 났나요?

컴포넌트가 비동기 API 호출을 할 때나 렌더링이나 어떠한 것이 업데이트 되기 전에 테스트가 종료 될 때는 따로 act로 감싸주어야하기 때문이다. (리액트 콜 스택 밖에 있을 때)

그래서 이 때는 waitFor API를 이용해서 테스트가 끝나기 전에 컴포넌트가

그러면 act 경고를 어떻게 해결해야하나요?

만들고 있는 앱에서 주문 완료 페이지에 "첫페이지로" 버튼을 누르면

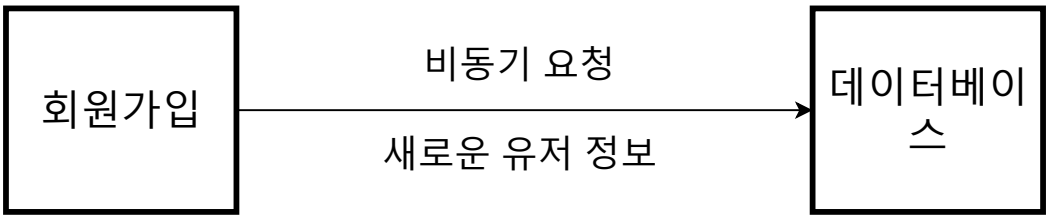
```
// 첫페이지로 버튼 클릭
const firstPageButton = screen.getByRole("button", { name: "첫페이지로" });
userEvent.click(firstPageButton);
```

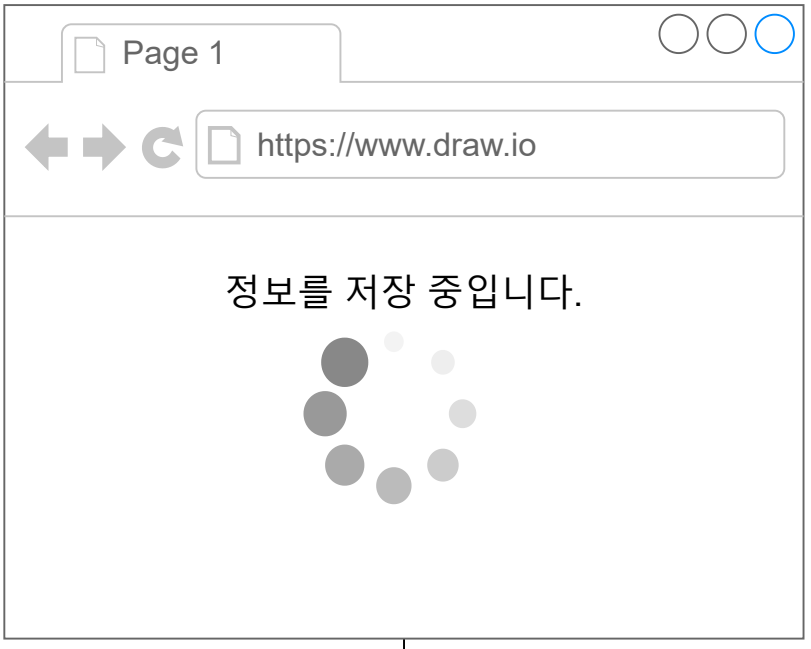
첫 페이지로 갔기 때문에 리액트는 첫페이지에서 어떠한 일이 일어날거라고 생각합니다.(America 여행 상품이나, Insurance 옵션등이 나타나는..) 하지만 테스트 코드에서는 첫 페이지부분으로 가는 버튼을 누르고 바로 테스트가 끝나버립니다. 그래서 리액트가 act 경고를 보여주게 됩

```
// 첫페이지로 버튼 클릭
const firstPageButton = screen.getByRole("button", { name: "첫페이지로" });
userEvent.click(firstPageButton);

await waitFor(() => {
  screen.getByRole("spinbutton", { name: "America" });
});
```

이렇게 첫 페이지에 온 후에 일어날 일들을 wairFor API를 이용해서 넣어주면 경고가 사라지게 됩니다.

다른 예제





저장이 끝나면
정보를 저장 중입니다 문구가 사라져야한다.


```
await waitForElementToBeRemoved(() => screen.getByText("정보를 저장 중입니다."))
```

이 부분을 넣어줘야 경고가 나타나지 않습니다.
리액트는 "정보를 저장 중입니다."가 없어지고 있는걸 예상하고
실제로 테스트에서 없애줘야 경고가 나타나지 않게 됩니다.
waitForElementToBeRemoved는 어떠한 요소(엘리먼트)가 돔에서
사라지는 걸 기다리는 것입니다.

첫 페이지로 돌아갈 때 State Reset!

결제 완료 페이지에서 첫페이지로 버튼을 눌러서 첫 페이지로 돌아갈 때 OrderContext안에 있는 State를 Reset 시켜주겠습니다.

해야 할 일은?

첫 페이지로 돌아올 때 모든 값이 초기화되게 만들기.

테스트 작성

```
// 첫페이지로 버튼 클릭
const firstPageButton = screen.getByRole("button", { name: "첫페이지로" });
userEvent.click(firstPageButton);

// 여행 상품 총 가격 옵션 총 가격이 reset 되었는지 확인
const productsTotal = screen.getByText('상품 총 가격: 0');
expect(productsTotal).toBeInTheDocument();
```

```
const optionsTotal = screen.getByText('옵션 총 가격: 0');
expect(optionsTotal).toBeInTheDocument();

await screen.findByRole("spinbutton", { name: "America" });
await screen.findByRole("checkbox", { name: "Insurance" });
```

테스트 실행

Fail

```
<p>
  상품
   총 가격:
  5000
</p>
<div
  style="display: flex;"
/>
```

테스트에 대응하는 실제 코드 작성

```
const [orderDatas, ,resetOrdeDatas] = useContext(OrderContext);
```

```
function handleClick() {
  // order data를 reset하기
  resetOrdeDatas();
  // 첫페이지로 보내기
  setStep(0)
}


if (loading) {
  return <div>loading</div>;
} else {
  return (
    <div style={{ textAlign: "center" }}>
      <h2>주문이 성공했습니다.</h2>
      <h3>지금까지 모든 주문</h3>
      <table style={{ margin: "auto" }}>
        <tbody>
          <tr>
            <th>number</th>
            <th>price</th>
          </tr>
          {orderTable}
        </tbody>
      </table>
      <br />
      <button className="rainbow rainbow-1" onClick={handleClick}>
        첫페이지로
      </button>
    </div>
  );
}
```

```
  const resetOrderDatas = () => {
    setOrderCounts({
      products: new Map(),
      options: new Map(),
    });
  };

  return [{ ...orderCounts, totals }, updateItemCount, resetOrderDatas];
}, [orderCounts, totals]);
```