

6강

조건부 렌더링 & 모달 구현

(UI 토글)





- 1. State로 UI 제어하기**
- 2. 조건부 렌더링(Conditional Rendering)**
- 3. props로 모달 제어하기**



➤ 핵심 개념

True/false

1. UI 상태 제어(state)

- 모달 열림/닫힘, 탭 선택, 드롭다운 열림 등은 모두 boolean이나 `selectedId` 같은 state로 관리한다.
- 예: `const [modalOpen, setModalOpen] = useState(false)` — `true`면 모달 보임, `false`면 숨김.

2. 조건부 렌더링(Conditional Rendering)

- 삼항연산자 `condition ? <A/> : ` 또는 `condition && <A/>`로 특정 컴포넌트를 보이게/감춘다.
- 렌더링 자체를 아예 하지 않기 때문에 성능/구조상 유리할 때가 많다.

3. props로 데이터 전달

- 모달 내부에서 상세 정보를 보여주려면 부모가 `selectedItem` 같은 데이터를 props로 전달한다.
- 닫기 함수(`onClose`)도 `props`로 내려서 모달 내부 버튼에서 부모 state를 변경하게 한다.



➤ State로 UI 제어하기

- React에서는 '화면 상태'도 데이터로 관리한다..

<예시 코드 >

```
import { useState } from 'react';

export default function ModalExample() {
  const [isOpen, setIsOpen] = useState(false); // 초기에는 모달이 닫힘

  return (
    <div>
      <button onClick={() => setIsOpen(true)}>모달 열기</button>
      {isOpen && <div className="modal">모달 내용</div>}
      {isOpen && <button onClick={() => setIsOpen(false)}>닫기</button>}
    </div>
  );
}
```

모달 열기



모달 열기

모달 내용

닫기



모달 열기

💡 핵심

- isOpen이 **true**면 모달 표시, **false**면 숨김
- “상태(state)”가 UI를 직접 제어
- “DOM을 조작하지 않고” 데이터로 화면이 바뀜



➤ 조건부 렌더링 (Conditional Rendering)

- 조건에 따라 JSX를 보여줄지 말지 결정한다.

<예시 코드 >

```
import { useState } from 'react';
export default function LoginView() {
  const [isLoggedIn, setIsLogin] = useState(false);
  return (
    <div>
      {isLoggedIn ? <h1>Welcome Back!</h1> : <h1>Please Login</h1>}
      <button onClick={() => setIsLogin(!isLoggedIn)}>
        {isLoggedIn ? '로그아웃' : '로그인'}
      </button>
    </div>
  );
}
```

💡 핵심

- if문 대신 JSX 안에서 간단히 조건 분기 가능
- 렌더링을 "안 하는 것"이므로 깔끔하고 성능에 유리

Please Login

로그인



Welcome Back!

로그아웃



➤ props로 데이터 전달

- 부모 state를 자식이 직접 바꾸면 안 되기 때문에, 함수를 내려준다

<예시 코드 >

```
import { useState } from 'react';
function Child({ onToggle }) {
  return <button onClick={onToggle}>모달 열기</button>;
}

export default function Parent() {
  const [open, setOpen] = useState(false);
  return (
    <div>
      <Child onToggle={() => setOpen(true)} />
      {open && (
        <div>
          <h2>모달창입니다.</h2>
          <button onClick={() => setOpen(false)}>닫기</button>
        </div>
      )}
    </div>
  );
}
```

💡 핵심

- open 상태를 부모가 관리합니다.
 - setOpen(true) → 모달 열림
 - setOpen(false) → 모달 닫힘
- 자식은 단지 버튼을 눌렀을 때 **onToggle**을 실행
- onToggle이 부모로부터 전달된 함수이기 때문에, 자식은 "**모달을 열어라**"라는 신호만 보내는 역할을 한다.



➤ 데이터 흐름

Parent (setOpen) → Child (onToggle props로 전달) → 클릭 시 setOpen(true)

- 📢 상태(state) 는 부모가 갖고 있고,
- 📢 상태를 바꾸는 함수를 자식에게 내려보내서,
- 📢 자식이 클릭 이벤트로 부모의 상태를 바꾸는 구조

이 패턴을 “상태 끌어올리기(lifting state up)” 라고도 부른다.

자식끼리 직접 상태를 주고받지 않고, 부모를 통해 제어하는 게 React의 기본 원리



➤ 여기서 잠깐!

- 📢 기존의 Blog.jsx 파일의 블로그 글 수정 / 삭제하기에 글 제목을 클릭하면 글 제목 수정 할 수 있도록 수정한다.
- 📢 모달 창은 [글수정][닫기]버튼으로 구현다
- 📢 [글수정]버튼을 클릭하면 Prompt 창이 뜨고, 수정할 데이터를 입력하면 글의 내용이 수정되도록 작성한다.
- 📢 자식 컴포넌트인 Model 을 생성하고 Props를 이용해 부모 Blog 컴포넌트에서 데이터를 받아 화면에 출력하도록 구현한다.



➤ 결과 화면

남자코트 추천👍0

11월 1일 발생

삭제

강남 우동맛집👍0

11월 1일 발생

삭제

파이썬독학👍0

11월 1일 발생

삭제

글 발행

남자코트 추천👍0

11월 1일 발생

삭제

강남 우동맛집👍0

11월 1일 발생

삭제

파이썬독학👍0

11월 1일 발생

삭제

글 발행

파이썬독학

날짜: 11월 1일

상세내용: 여기에 내용을 넣어보세요

글수정 닫기

남자코트 추천👍0

11월 1일 발생

삭제

강남 우동맛집👍0

11월 1일 발생

삭제

Java 독학👍0

11월 1일 발생

삭제

글 발행

Java 독학

날짜: 11월 1일

상세내용: 여기에 내용을 넣어보세요

글수정 닫기

localhost:5173 내용:
새 제목을 입력하세요

확인 취소



➤ 여기서 잠깐!: 컴포넌트 분리 하기

조건

- ① 로그인/회원가입 두 가지 폼 존재
- ② 버튼 클릭 시 화면 전환 (조건부 렌더링)
- ③ 입력 완료 시 상위 컴포넌트에 메시지 전달 (props 활용)
- ④ 성공 시 상단에 "환영합니다, 000님!" 표시

⑤ 컴포넌트 구조

<AuthApp />

└ <LoginForm />


└ <SignupForm />

└ <Message /> (props로 결과 표시)

- ⑥ src 폴더안에 -> stateUI 폴더 생성 -> member 폴더 생성 -> AuthApp.jsx, LoginForm.jsx, SignupForm.jsx, Message.jsx 파일을 생성하여 작성하시오.
- ⑦ src 폴더안에 -> App.jsx에 작성한 AuthApp.jsx를 import하여 실행 하시오.




<초기 출력 화면 >

로그인/회원가입 

<이름, 비밀번호 입력 없을 때>

localhost:5173 내용:
입력해주세요!

<회원 가입 하고 , 로그인 하기>

로그인/회원가입 

환영합니다, kjb1045님!

심화 연습 문제





문제1] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

- ① 영화 제목 + 리뷰 입력 → 목록에 추가
- ② 각 영화는 "상세보기" 버튼 클릭 시 모달로 세부 내용 표시
- ③ 리뷰 없으면 "아직 작성된 리뷰가 없습니다." 출력
- ④ 모달은 props를 통해 부모 → 자식으로 데이터 전달

⑤ 컴포넌트 구조

<MovieApp /> // 부모

└ <MovieList /> // 영화 목록

└ <MovieItem /> // 각 영화 항목

└ <MovieModal /> // 선택된 영화 상세정보 모달

- ⑥ src 폴더안에 -> stateUI 폴더 생성 -> movie 폴더 생성 -> MovieApp.jsx, MovieList.jsx, MovieItem.jsx, MovieModal.jsx 파일을 생성하여 작성하시오.
- ⑦ src 폴더안에 -> App.jsx에 작성한 MovieApp.jsx를 import하여 실행 하시오.



<초기 출력 화면>

영화 리뷰 관리 앱

<추가 화면>

영화 리뷰 관리 앱

-

사랑과 영혼 상세보기

-

포켓몬 상세보기

<상세보기 화면>

사랑과 영혼

인생영화



문제2] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

- ① 상품 목록 map() 렌더링
- ② "담기" 클릭 시 장바구니 수량 증가
- ③ "장바구니 보기" 클릭 시 모달 오픈
- ④ 장바구니가 비어있으면 "비어있습니다" 출력
- ⑤ **컴포넌트 구조**

<ShopApp /> // 부모

└ <ProductList /> // 상품 목록

└ <ProductItem /> // 각 상품

└ <CartModal /> // 장바구니 모달

아래 오브젝트 배열을 이용해 작성하세요

```
const [products] = useState([
  { id: 1, name: '노트북', price: 1200000 },
  { id: 2, name: '마우스', price: 25000 },
  { id: 3, name: '키보드', price: 50000 },
]);
```

- ⑥ src 폴더안에 -> stateUI 폴더 생성 -> shop 폴더 생성 -> ProductList.jsx, ProductItem.jsx, CartModal.jsx 파일을 생성하여 작성하시오.
- ⑦ src 폴더안에 -> App.jsx에 작성한 ProductList.jsx를 import하여 실행 하시오.



<초기 출력 화면>

쇼핑몰

장바구니 보기

- 노트북 - 1200000원 담기
- 마우스 - 25000원 담기
- 키보드 - 50000원 담기

<노트북 담기>

쇼핑몰

장바구니 보기

- 노트북 - 1200000원 **담기**
- 마우스 - 25000원 담기
- 키보드 - 50000원 담기

<장바구니 보기>

쇼핑몰

장바구니

- 노트북 x 1

담기

- 키보드 - 50000원 담기

<노트북 두 번 담기>

쇼핑몰

장바구니

- 노트북 x 2

담기

- 키보드 - 50000원 담기



문제3] 다음의 조건에 만족하도록 React를 작성 하시오.

조건

- ① 여행지, 날짜, 메모 입력 → 목록 추가
- ② 일정 클릭 시 상세보기 모달
- ③ "편집" 버튼 클릭 시 수정 모드
- ④ props로 부모 → 자식으로 선택된 일정 전달

⑤ 컴포넌트 구조

```
<TripPlanner />  
├ <TripList />  
├ <TripItem />  
└ <TripModal /> (조건부 렌더링, props 사용)
```

- ⑥ src 폴더안에 -> stateUI 폴더 생성 -> trip 폴더 생성 -> TripPlannerApp.jsx, TripList.jsx, TripItem.jsx, TripModal.jsx 파일을 생성하여 작성하시오.
- ⑦ src 폴더안에 -> App.jsx에 작성한 TripPlannerApp.jsx를 import하여 실행 하시오.