📌 React Props (속성)

🚀 1. Props란?

📖 개념

- **Props(properties)는 부모 컴포넌트가 자식 컴포넌트에게 데이터를 전달하는 방법**입니다.
- React에서 컴포넌트는 **재사용 가능하고 독립적인 요소**이므로, 부모-자식 간의 데이터 전달이 필요할 때 props를 사용합니다.
- Props는 **읽기 전용(Read-Only)** 이므로, 자식 컴포넌트에서 직접 변경할 수 없습니다.

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}

// 사용 예시
<Greeting name="John" />
```

✅ 부모 컴포넌트에서 `name`이라는 props를 전달하면 자식 컴포넌트에서 이를 사용할 수 있습니다.

💡 2. Props의 주요 특징

✅ Props는 부모 → 자식 단방향 전달 (One-Way Binding)

- 부모 컴포넌트에서 내려준 값은 **자식 컴포넌트에서만 읽을 수 있고 수정할 수 없음**
- 데이터 흐름을 예측 가능하게 유지

✅ Props는 객체 형태로 전달

- Props는 내부적으로 객체이며, 여러 개의 데이터를 한 번에 전달 가능

```
function UserInfo(props) {
  return (
    <div>
      <h2>{props.name}</h2>
      <p>Age: {props.age}</p>
    </div>
  );
}
```

<UserInfo name="Alice" age={25} />

✅ `props.name`, `props.age`를 통해 부모가 전달한 값을 사용 가능

🛠 3. Props 사용법

📌 1) 기본적인 Props 사용

```
function Welcome(props) {
  return <h1>안녕하세요, {props.username}님!</h1>;
}

<Welcome username="철수" />
```

✅ `username`이라는 props를 전달하고, 자식 컴포넌트에서 사용

📌 2) 객체 형태의 Props 전달

```
function UserCard({ user }) {
  return (
    <div>
      <h2>{user.name}</h2>
      <p>Email: {user.email}</p>
    </div>
  );
}

const userData = { name: "홍길동", email: "hong@example.com" };
<UserCard user={userData} />
```

✅ 객체 전체를 props로 전달하여 컴포넌트 내에서 `user.name`, `user.email`로 접근 가능

📌 3) Props의 기본값 설정 (defaultProps)

```
function Greeting(props) {
  return <h1>Hello, {props.name}!</h1>;
}

Greeting.defaultProps = {
  name: "Guest",
};
```

✅ `name`이 전달되지 않았을 경우 기본값인 `Guest`가 사용됨

📌 4) Props의 데이터 타입 검증 (PropTypes)

```
import PropTypes from "prop-types";

function Profile({ name, age }) {
  return (
    <div>
      <h2>{name}</h2>
      <p>나이: {age}</p>
    </div>
  );
}

Profile.propTypes = {
  name: PropTypes.string.isRequired,
  age: PropTypes.number,
};
```

✅ `name`은 반드시 문자열이어야 하고, `age`는 숫자 타입이어야 한다는 규칙 적용

🔥 4. Props를 활용한 컴포넌트 재사용

📌 1) 리스트 데이터 렌더링

```
function ItemList({ items }) {
  return (
    <ul>
      {items.map((item, index) => (
        <li key={index}>{item}</li>
      ))}
    </ul>
  );
}

const fruits = ["사과", "바나나", "포도"];
<ItemList items={fruits} />
```

✅ `props.items`를 사용하여 리스트 데이터를 동적으로 렌더링

📌 2) 자식 컴포넌트에 함수 전달 (콜백 함수 패턴)

```
function Button({ onClick }) {
  return <button onClick={onClick}>클릭</button>;
}
```

```
function ParentComponent() {
  function handleClick() {
    alert("버튼이 클릭되었습니다!");
  }

  return <Button onClick={handleClick} />;
}
```

✅ 부모 컴포넌트에서 함수를 자식 컴포넌트로 전달하여 실행 가능

🎯 5. Props vs State 비교

비교 항목	Props	State
데이터 흐름	부모 → 자식 (단방향)	컴포넌트 내부에서 관리
수정 가능 여부	❌ (불변)	✅ (setState 또는 useState로 변경 가능)
사용 목적	부모 컴포넌트에서 자식으로 데이터 전달	컴포넌트 내부에서 변화하는 값 관리

🎯 6. 적용

✅ **부모 → 자식 데이터 전달 시 props 사용**

✅ **객체 형태의 props를 활용하여 구조적 데이터 전달**

✅ **defaultProps 및 PropTypes로 props의 안정성 강화**

✅ **리스트 데이터 렌더링 시 props로 전달하여 동적 UI 구성**

✅ **부모에서 함수를 props로 전달하여 자식에서 이벤트 처리 가능**