리액트(React)는 **사용자 인터페이스(UI)를 구축하기 위한 JavaScript 라이브러리**입니다. 웹 애플리케이션의 **복잡한 화면**을 효율적으로 개발하고 관리하기 위해 탄생했습니다.

**1. 리액트를 쓰는 이유 (Why Use React?) 💡**

리액트가 현대 웹 개발에서 가장 널리 사용되는 주된 이유는 다음과 같습니다.

**A. 컴포넌트 기반 개발 (Component-Based Architecture)**

리액트는 UI를 \*\*독립적이고 재사용 가능한 부품(컴포넌트)\*\*으로 쪼개서 개발합니다.

* **재사용성:** 헤더, 버튼, 목록 항목 등 하나의 컴포넌트를 만들어두면 앱의 여러 곳에서 그대로 가져다 쓸 수 있습니다.
* **유지보수:** 문제가 생기거나 기능을 추가할 때, 전체 코드가 아니라 해당 컴포넌트만 수정하면 됩니다.

**B. 가상 DOM (Virtual DOM)**

리액트의 핵심 기능 중 하나입니다.

* **성능 최적화:** 웹 브라우저에서 화면을 변경할 때마다 실제 DOM(문서 객체 모델)을 직접 조작하는 것은 매우 느립니다. 리액트는 **가상 DOM**이라는 메모리상의 복사본을 만들어 변경된 부분만 비교하고, **가장 효율적인 방법으로 실제 DOM에 한 번에 반영**합니다. 이로 인해 빠른 화면 업데이트 성능을 제공합니다.

**C. 단방향 데이터 흐름 (One-way Data Flow)**

데이터가 항상 **부모 컴포넌트에서 자식 컴포넌트**로 흐릅니다 (Props).

* **예측 가능성:** 데이터의 흐름이 한 방향으로만 이루어지기 때문에, 코드를 이해하고 디버깅하기 쉬워져 복잡한 애플리케이션의 오류를 줄일 수 있습니다.

**2. 리액트 기초 개념 (React Fundamentals) 🧱**

리액트 개발을 시작하는 데 필요한 가장 기본적인 세 가지 요소는 **JSX, 컴포넌트, State & Props**입니다.

**A. JSX (JavaScript XML)**

JSX는 리액트에서 사용되는 특별한 문법입니다. **JavaScript 코드 내부에 HTML과 유사한 마크업 구조를 작성**할 수 있게 해줍니다.

* **역할:** JSX는 브라우저가 이해하는 일반 JavaScript로 변환(트랜스파일)되어 화면을 그립니다.
* **특징:** HTML처럼 보이지만, JavaScript의 변수나 표현식을 {중괄호} 안에 넣어 사용할 수 있습니다.

JavaScript

// JSX 예시

const name = "React";

const element = <h1>Hello, {name}!</h1>;

// JavaScript 변수 'name'을 HTML 태그 안에 삽입

**B. 컴포넌트 (Components)**

리액트 앱을 구성하는 기본 단위입니다. 두 가지 방식으로 정의됩니다.

| **종류** | **정의 방식** | **특징** |
| --- | --- | --- |
| **함수 컴포넌트 (Function Component)** | 일반 JavaScript 함수로 작성 | 리액트 개발의 주류 방식이며, **Hooks**를 사용하여 상태(State) 및 생명주기 관리가 가능합니다. |
| **클래스 컴포넌트 (Class Component)** | React.Component를 상속받아 작성 | 이전 방식이며, 현재는 Hooks를 사용할 수 없어 잘 쓰이지 않습니다. |

**C. Props (속성)**

Props는 \*\*Properties(속성)\*\*의 약자입니다. 컴포넌트 간에 데이터를 전달할 때 사용됩니다.

* **흐름:** **부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달**하는 유일한 방법입니다.
* **특징:** Props는 한 번 전달되면 \*\*읽기 전용(Read-Only)\*\*이며, 자식 컴포넌트 내부에서는 값을 변경할 수 없습니다 (**불변성**).

**D. State (상태)**

State는 컴포넌트 내부에서 **시간이 지남에 따라 변할 수 있는 데이터**를 의미합니다.

* **역할:** 사용자 입력, API로부터 받은 데이터 등 컴포넌트의 동적인 부분을 관리합니다.
* **업데이트:** State가 변경되면, 리액트가 해당 컴포넌트와 관련된 UI만 **자동으로 재렌더링**합니다.
* **Hooks 사용:** 함수 컴포넌트에서는 useState라는 Hooks 함수를 사용하여 State를 선언하고 업데이트합니다.

JavaScript

// State 예시 (Hooks 사용)

import React, { useState } from 'react';

function Counter() {

// count: 현재 상태 값, setCount: 상태를 업데이트하는 함수

const [count, setCount] = useState(0);

return (

<div>

<p>You clicked {count} times</p>

{/\* setCount를 통해 상태를 변경해야 재렌더링이 일어남 \*/}

<button onClick={() => setCount(count + 1)}>

Click me

</button>

</div>

);

}