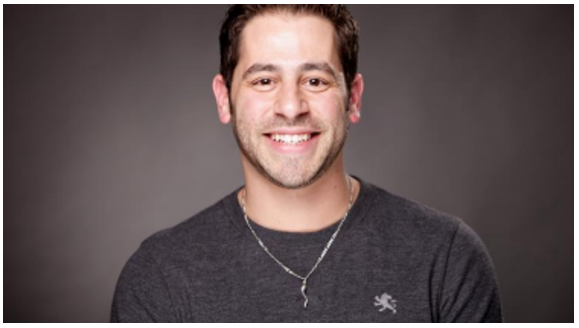
React소개

💡 **FaceBook 에서 개발한 JavaScript 기반 UI 개발 라이브러리**

- React는 페이스북에서 개발한 JavaScript 라이브러리로, 사용자 인터페이스를 만들기 위한 라이브러리입니다.
- React는 사용자 인터페이스의 구축과 관리를 도와주는 선언적이고 효율적인 방법을 제공합니다.
- React는 단일 페이지 애플리케이션(SPA) 및 모바일 애플리케이션에서 특히 뛰어난 성능을 보이며, 컴포넌트기반 아키텍처를 기반으로 하고 있습니다.
- React는 2011년에 페이스북에서 처음으로 도입되었고, 2013년에 오픈소스로 공개되었습니다.
- 그 이후로도 React 커뮤니티는 계속 성장하고 발전하며, 많은 개발자들이 이 라이브러리를 사용하여 현대적이고 효율적인 웹 애플리케이션을 구축하고 있습니다.

React 창시자

React는 페이스북에서 개발되었으며, React를 만든 팀 중 주요한 인물 중 한 명은 Jordan Walke입니다. Jordan Walke는 React를 초기에 개발하고 그 아이디어를 구상한 주요 개발자 중 하나로 알려져 있습니다.



Jordan Walke는 React의 초기 버전을 만들면서 JSX(JavaScript XML) 문법을 도입하고, React 의 핵심 개념 중 하나인 가상 DOM(Virtual DOM)을 도입하는 등의 기술적인 결정을 내린 인물 중 하나로 알려져 있습니다.

DOM

웹브라우저 내부의 html 태그 객체를 DOM (Document Object Model) 이라고 한다.
대부분의 웹브라우저가 C++로 구현되었기 때문에,
DOM 객체도 C++ 클래스로 구현된 객체이다.

웹브라우저가 HTML 문서를 읽으면,
HTML 태그 각각에 대해서 웹브라우저 내부에 DOM 객체가 생성되고,
그 DOM 객체들이 웹브라우저 창에 그려진다.

프론트엔드 앱의 화면

(1) 프론트엔드 앱의 화면에 해당하는 html 태그 DOM 객체들을 처음부터 전부 새로 생성하기
(2) 데이터가 변경되면 화면에서 그 데이터가 출력된 부분의 html 태그 DOM 객체 수정하기

(1) 구현보다 (2) 구현이 훨씬 복잡하고 어렵다.

그래서 (2) 부분을 구현하지 않고 (1) 부분만 구현하고,
데이터가 변경될 때 마다 화면 전체를 새로 생성하는 방식이 훨씬 구현하기 쉽다.

구현하기는 쉽지만 너무 비효율적이고 느리다.
이 비효율을 개선하는 방법은??

Virtual DOM

웹브라우저 내부의 DOM 객체들에는 많은 기능이 구현되어 있어서 무겁고 느리다.

기능이 많아서 무겁고 느린 DOM 객체들을,

> 데이터가 변경될 때 마다 전부 새로 만드는 것은 너무 비효율적이다.

그래서 등장한 아이디어가 Virtual DOM 객체를 활용하는 것이다.

Virtual DOM은 DOM을 흉내내서 만든 평범한 javascript 객체이다.
DOM과는 달리 별 기능이 없어서 가볍고 빠르다.

데이터가 수정될 때마다,

무거운 DOM 객체들 전체를 다시 생성하는 것은 너무 느리지만,
가벼운 Virtual DOM 객체들 전체를 다시 생성하는 것은 빠르다.

React와 개발자의 역할 분담

개발자는

데이터를 웹브라우저 창에 출력하기 위해, 웹페이지 전체 Virtual DOM 객체를 생성하는 코드만 구현한다.
이 구현은 비교적 쉽다.

데이터가 수정되거나 삭제될 때, 해당 Virtual DOM 객체를 찾아서 수정하거나 삭제하는 코드는 구현하지 않는다. 이 구현이 꽤 어려운데 안 해도 되니 다행이다.

React는

데이터가 수정되거나 삭제될 때 마다, 개발자가 구현한 코드를 실행해서,
웹페이지 전체 Virtual DOM 객체들을 다시 생성한다.

데이터가 수정되기 전, 웹페이지 전체 Virtual DOM 객체들과
데이터가 수정된 후, 웹페이지 전체 Virtual DOM 객체들을 비교한다.

이 비교로 어느 부분이 수정되었고 삭제되었는지를 React가 파악한다.
그 수정되고 삭제된 부분에 해당하는 웹브라우저 내부 DOM 객체들을 React가 수정한다.

장점

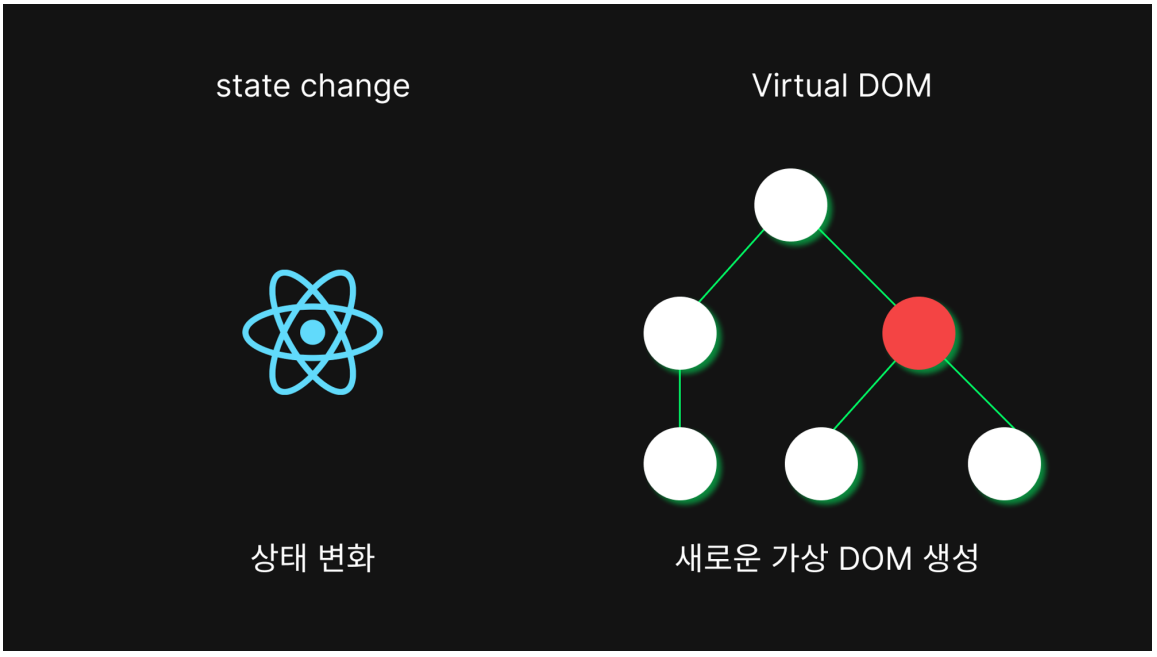
React가 딱 필요한 DOM 객체만 수정 삭제하기 때문에 빠르다.

개발자는, 웹페이지 전체 Virtual DOM 객체 생성 코드만 구현하면 되고,
데이터가 수정 삭제될 때, 해당 Virtual DOM 객체를 찾아서 수정 삭제하는 코드까지 구현할 필요는 없다.
그래서 구현이 쉽다.

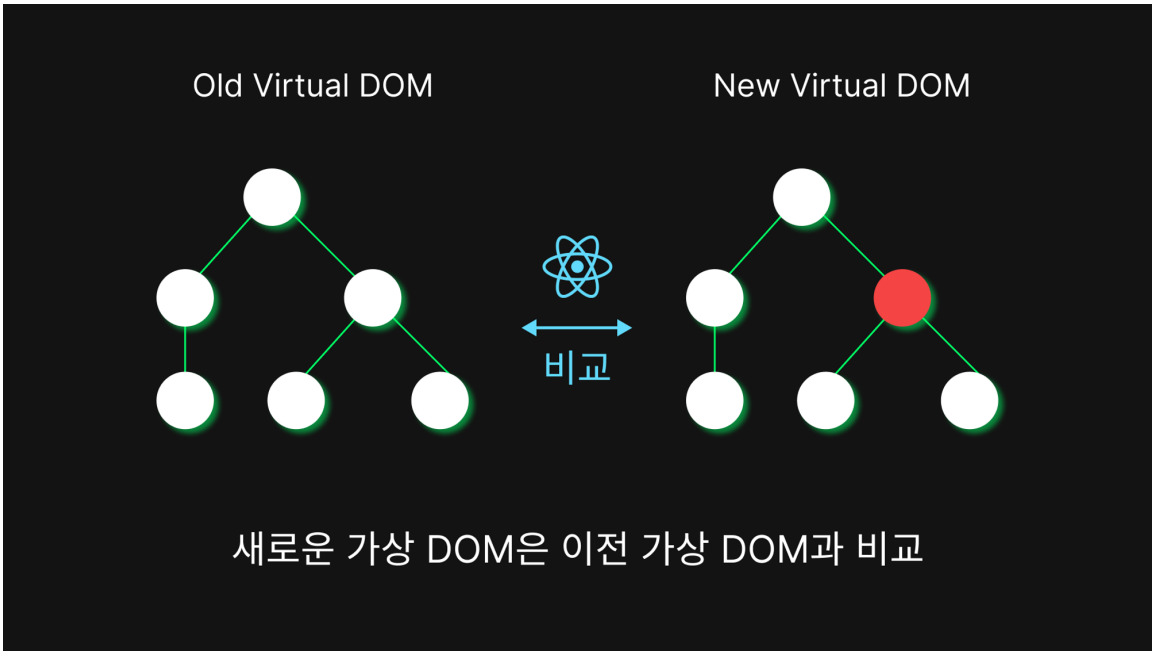
💡 **React는 가상 DOM(Virtual DOM)을 사용하여 속성(attribute) 또는 상태(State)의 변화에 반응하여 UI를 업데이트하는 구조이다.**

Reconciliation (재조정)

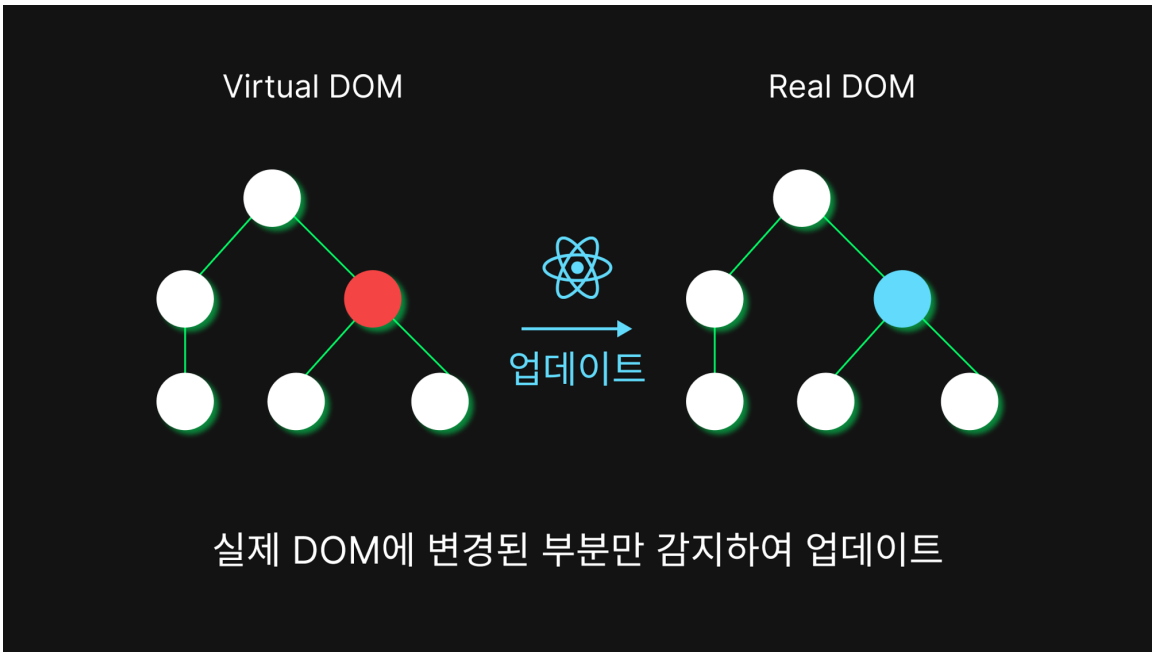
1. **가상 DOM 생성**
   : 상태나 속성이 변경되면 React는 컴포넌트의 새로운 가상 DOM을 생성합니다. 이 가상 DOM은 변경 사항을 반영한 컴포넌트의 트리 구조를 가집니다.



2. **이전 가상 DOM과 비교**
   :새로운 가상 DOM은 이전 가상 DOM과 비교됩니다. 이 비교를 통해 변경된 부분이 식별되고, 실제 DOM에 적용해야 할 최소한의 업데이트가 결정됩니다.



3. **실제 DOM 업데이트**
   : 변경된 부분이 식별되면 React는 이를 바탕으로 실제 DOM을 업데이트합니다. 이때 최소한의 조작만 이루어지므로 성능이 향상됩니다.



생각해보기

💡 **상태 변경을 감지해서 모든 구조를 다시 생성하는 것이 빠를까?**
변경된 부분만 업데이트하는 것이 빠를까?

- Reconciliation (재조정)을 통해 React는 효율적으로 UI를 업데이트하고 성능을 최적화할 수 있습니다.
- 이는 React의 성능 특징 중 하나로, 화면 전체를 다시 그리는 것이 아니라 변경된 부분만을 감지하여 업데이트 하므로써 빠른 화면 갱신을 가능하게 합니다.

리액트 컴포넌트

리액트 개발자는 리액트 컴포넌트를 구현해야 한다.

리액트 컴포넌트를 클래스 형태로 구현하거나 함수 형태로 구현할 수 있는데,
함수 형태로 구현하는 것이 바람직하다.

함수형 리액트 컴포넌트는 Virtual DOM 객체를 생성해서 리턴하는 자바스크립트 함수이다.

리액트 앱은

> 화면을 구성하는 html 태그 Virtual DOM 객체들을 생성해서 리턴하는 함수들로 구성된다.
> (함수형 컴포넌트들로 구성된다)

리액트 앱은 App 함수형 컴포넌트에서 시작한다.

> 즉 App 함수형 컴포넌트의 리액트 앱의 화면에 구성하는 html 태그 Virtual DOM 객체들을 생성해서 리턴하는 함수이다.

주요 개념

1. **컴포넌트 기반**
   : React 애플리케이션은 작고 재사용 가능한 컴포넌트로 구성됩니다. 각 컴포넌트는 자체적으로 상태(state)를 가질 수 있으며, 이러한 컴포넌트를 조합하여 전체 애플리케이션을 구축합니다.
2. **가상 DOM (Virtual DOM)**
   : React는 가상 DOM을 사용하여 브라우저에 렌더링되기 전에 가상으로 모든 변경 사항을 처리합니다. 이를 통 해 효율적으로 화면을 갱신하고 성능을 최적화할 수 있습니다.
3. **JSX (JavaScript XML)**
   : React에서는 JSX라는 문법을 사용하여 JavaScript 코드 안에서 XML과 유사한 문법으로 UI를 작성할 수 있습니다. JSX는 간결하고 가독성이 높은 코드를 작성할 수 있도록 도와줍니다.
4. **단방향 데이터 바인딩**
   : React는 단방향 데이터 바인딩을 지원하여 데이터의 흐름이 단일 방향으로 유지됩니다. 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달할 수 있으나, 그 반대는 자동으로 이루어지지 않습니다.
5. **상태 관리**
   : React 애플리케이션에서는 컴포넌트의 상태를 통해 동적인 데이터를 다룰 수 있습니다. 상태는 컴포넌트 내에서 변경이 가능하며, 상태의 변화에 따라 자동으로 화면이 갱신됩니다.
6. **라우팅**
   : React는 라우터를 통해 다양한 페이지 간의 전환이나 SPA에서의 라우팅을 쉽게 처리할 수 있습니다. React Router는 React 애플리케이션에서 라우팅을 구현하는 데 사용되는 널리 사용되는 라이브러리 중 하나입니다.

주요 학습 요소

JSX (JavaScript XML)

- JSX는 JavaScript의 확장 문법으로, JavaScript 코드 안에서 XML 형식의 마크업을 작성할 수 있게 합니다.

- React에서 컴포넌트의 UI를 정의하는데 주로 사용되며, 가독성이 높고 코드 작성이 편리하도록 도와줍니다.

```
const element = <h1>Hello, World!</h1>;
```

컴포넌트

- 컴포넌트는 UI를 독립적이고 재사용 가능한 부분으로 나누는 개념입니다.
- React 애플리케이션은 여러 컴포넌트들이 조합되어 전체 UI를 형성합니다.

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

Props (속성)

- Props는 부모 컴포넌트에서 자식 컴포넌트로 데이터를 전달하는 데 사용되는 매개변수입니다.
- 컴포넌트는 props를 통해 부모로부터 전달받은 데이터를 사용할 수 있습니다.

```
<Welcome name="John" />
```

State (상태)

- State는 컴포넌트가 자체적으로 유지하고 관리하는 데이터입니다.
- 상태가 변경되면 React는 컴포넌트를 리렌더링하고 변경된 상태를 반영합니다.

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }
```

```
  // ...
}
```

라이프사이클(생명주기)

- 컴포넌트가 생성, 갱신, 제거되는 과정을 라이프사이클이라고 합니다.
- 주요 단계에 따라 라이프사이클 메서드를 사용하여 작업을 수행할 수 있습니다.

```
class ExampleComponent extends React.Component {
  componentDidMount() {
    // 컴포넌트가 마운트된 후 실행되는 코드
  }
  // ...
}
```

Hooks

- Hooks는 함수 컴포넌트에서 상태와 생명주기 기능을 사용할 수 있게 해주는 함수입니다.
- useState, useEffect 등이 널리 사용되며, 함수형 컴포넌트에서 상태와 라이프사이클 기능을 사용할 수 있게 합니다.

```
import React, { useState, useEffect } from 'react';

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

  useEffect(() => {
    // 컴포넌트가 마운트 또는 갱신된 후 실행되는 코드
  }, [count]);

  // ...
}
```

이벤트 핸들링

- React에서는 이벤트 핸들링이 일반적인 HTML과 유사하지만, camelCase 표기법을 사용합니다.
- 핸들러 함수를 만들고 JSX에서 이벤트에 연결할 수 있습니다.

```
function handleClick() {
  alert('Button Clicked!');
}

<button onClick={handleClick}>Click me</button>
```

조건부 렌더링

- JSX에서 조건에 따라 다른 컴포넌트나 엘리먼트를 렌더링할 수 있습니다.
- 삼항연산자나 && 연산자 등을 사용하여 조건부 렌더링을 구현할 수 있습니다.

```
function Greeting({ isLoggedIn }) {
  return isLoggedIn ? <UserGreeting /> : <GuestGreeting />;
}
```

React & React Native

두 기술 모두 Facebook에서 개발되었으며, React에서 사용되는 기술과 개념을 공유합니다.

React Native는 React를 기반으로 하여 JavaScript 및 React를 사용하여 네이티브 모바일 애플리케이션을 만들 수 있도록 하는 것이 주요 목표입니다.

이러한 유사성으로 인해 React로 작성된 웹 애플리케이션과 React Native로 작성된 모바일 애플리케이션 간의 코드 재사용이 일부 가능합니다.

React

💡 **웹 애플리케이션 개발 라이브러리** https://ko.legacy.reactjs.org/

- **웹 애플리케이션 개발을 위한 라이브러리** : React는 주로 웹 애플리케이션을 개발하기 위한 라이브러리로 사용됩니다.
- **컴포넌트 기반 아키텍처** : React는 컴포넌트 기반 아키텍처를 기반으로 한다. 이는 UI를 재사용 가능하고 독립적인 조각으로 나누어 개발하는데 도움을 준다.

React로 개발한 웹사이트 : Facebook, Instagram

React Native

💡 **모바일 애플리케이션 개발 라이브러리** https://reactnative.dev/

- 모바일 애플리케이션 개발을 위한 프레임워크: React Native React의 확장으로, 주로 iOS 및 Android 모바일 애플리케이션을 개발하기 위한 프레임워크로 사용됩니다.
- 네이티브 모바일 앱 개발: React Native는 JavaScript와 React를 사용하여 네이티브(원래의 플랫폼에서 실행되는) 모바일 애플리케이션을 만들 수 있도록 지원합니다.
- 컴포넌트 기반 아키텍처의 확장: React Native도 React처럼 컴포넌트 기반 아키텍처를 사용하며, React에서 사용하는 것과 유사한 개념과 API를 제공합니다.

React Native로 개발한 앱 : Facebook, Instagram
