< React >

1. 리액트란 무엇인가?

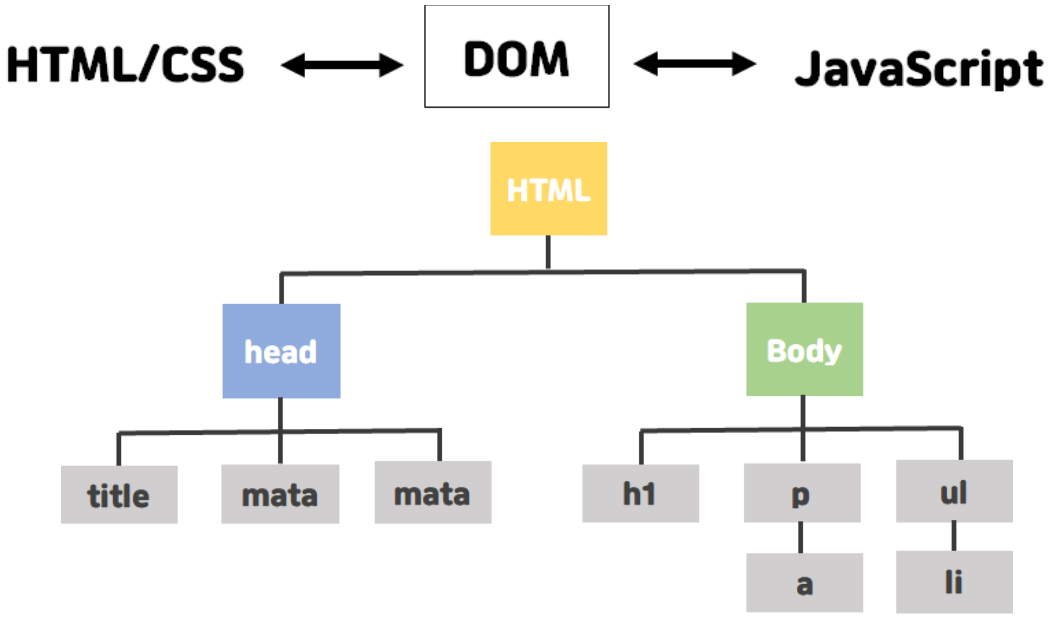
1) 돔 (DOM)

DOM(Document Object Model)은 **웹 페이지를 이루는 태그들을 자바스크립트가 이용할 수 있게 브라우저가 트리구조로 만든 객체 모델을 의미**한다.

DOM(Document Object Model)을 영어 뜻풀이 그대로 하자면 문서 객체 모델을 의미 한다.

문서 객체란 html, head, body와 같은 태그들을 javascript가 이용할 수 있는 (메모리에 보관할 수 있는) 객체를 의미한다.

DOM은 HTML과 스크립팅 언어(Javascript)를 서로 이어주는 역할이다.



HTML의 DOM 트리

2) 가상 돔(Virtual DOM)의 등장배경

요즘 흔히 접하는 큰 규모의 웹 애플리케이션(트위터, 페이스북)은 스크롤바를 내릴수록 수많은 데이터가 로딩되는데, 각 데이터를 표현하는 수 백 개, 수천개의 요소들이 웹 애플리케이션에서 DOM에 직접 접근해 변화를 주다 보니 성능 이슈가 발생하기 시작했다.

근데 이건 정확한 원인이 아니다.

DOM자체는 정말 빠르다. 읽고 쓸 때의 성능은 자바스크립트 객체를 처리할 때의 성능과 비교하여 다르지 않다.

단, **웹 브라우저 단에서 DOM 변화가 일어나면 웹 브라우저가 CSS를 다시 연산하고 레이아웃을 구성하고, 페이지를 리 페인트 즉 렌더링이 일어 나는 이 과정에서 시간이 허비되는 것이다.**

그리고 이 렌더링 과정은 돔이 추가, 삭제 혹은 태그 위치가 변하는 등의 상황에 따라 여러 번 반복하여 발생하기에 빈번히 발생한다.

렌더링이란? 렌더링: 브라우저 로딩 과정 중 스타일 이후의 과정(스타일-> 레이아웃 -> 페인트 -> 합성)

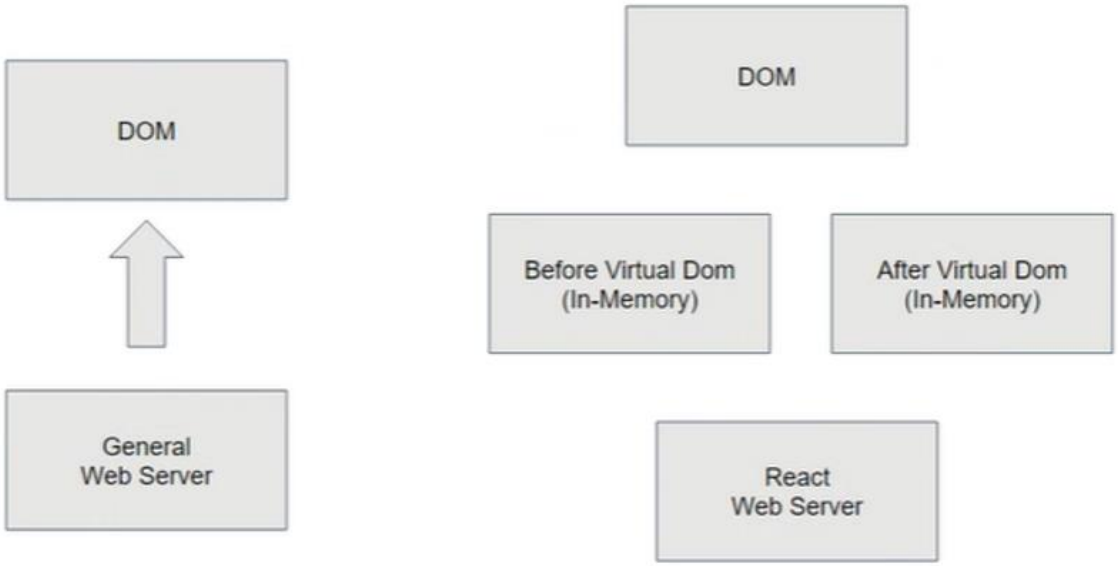
결론은 **속도 적인 부분과 많은 일을 수행하다 버그가 발생, 브라우저가 죽는 일 등등의 일을 개선하고자 가상 돔(Virtual DOM)이 출현했다.**

3) 가상 돔(Virtual DOM) 이란?

Virtual DOM을 사용하면 실제 DOM에 접근하여 조작하는 대신, 이를 추상화한 자바스크립트 객체를 구성하여 사용한다.

DOM의 상태를 메모리에 저장하고, 변경 전과 변경 후의 상태를 비교한 뒤 최소한의 내용만 반영하는 기능이다. 이는 **성능을 향상** 시킨다.

가상 DOM은 DOM의 상태를 메모리 위에 계속 올려 두고, DOM에 변경이 있을 경우 해당 변경을 반영한다.



1. React는 Virtual Dom을 활용하여 실제 바뀐 부분만 Dom에 반영
   => Dom Rendering 성능 향상
2. React는 **단일 방향 이벤트 처리** 모델이다.

4) 리액트가 가상돔을 반영하는 절차

만약, 특정 페이지에서 데이터가 변했다고 가정했을 경우 리액트를 이용해 돔을 업데이트 시키는 절차이다.

4.1) 데이터가 업데이트 되면, 전체 UI를 Virtual DOM에 리렌더링함

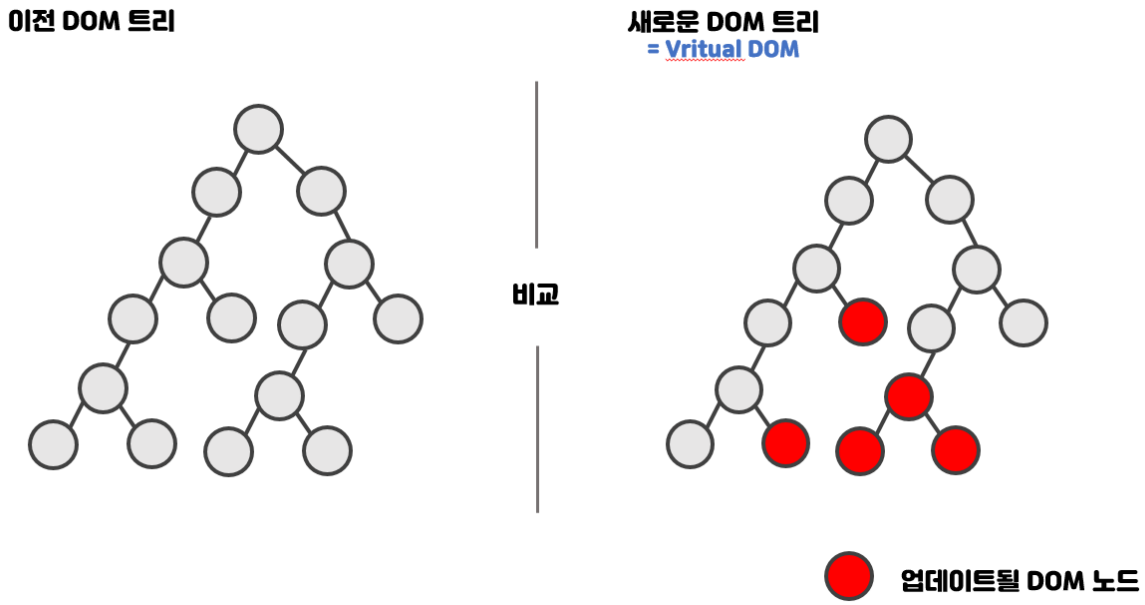
4.2) 이전 Virtual DOM에 있던 내용과 현재의 내용을 비교함 **(가상 돔끼리 비교)**

4.3) **바뀐 부분**만 실제 DOM에 적용이 됨

(컴포넌트가 업데이트 될 때 , 레이아웃 계산이 한번만 이뤄짐)

작은 규모의 레이아웃(리플로우)이 여러 번 발생하는 것보다 큰 규모의 레이아웃이 한 번 발생하는 것은 성능상의 큰 차이를 나타낸다.

리액트는 위와 같은 얕은 비교와 일괄 돔 업데이트 방식을 이용해 성능 향상을 이끄는 것이다.



5) React 의 장점

5.1) 활성화된 오픈 소스 커뮤니티

5.2) Routing 관련하여 압도적인 기능 스펙과 성능 (링크: v5문서 & v6문서)

5.3) State Management의 관리 기술 => Redux 및 Redux Toolkit

5.4) Context API => Redux와 더불어 Context 단위 전체 컴포넌트 라이프 사이클 관리

5.5) 다양한 React Pattern => 기존의 MVC 모델에 강제되는 것이 아닌, 프로젝트의 특징 및 목적에 맞는 개발 디자인 패턴 적용 용이

5.6) Bundlers => jar & maven & gradle 지옥 탈출. Webpack은 사랑입니다.

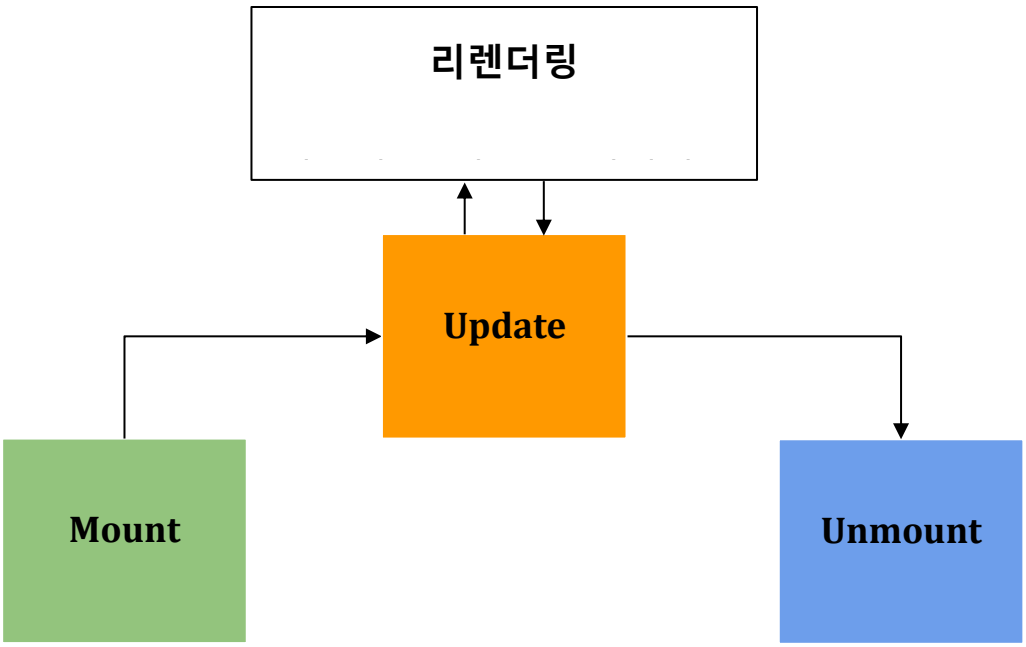
5.7) Forms => reack-hook-form 기반 훌륭한 다국어 처리 및 예외 처리 기술

2. React LifeCycle

React에서 컴포넌트는 여러 종류의 생명주기 메소드를 가지며 이 메소드를 오버라이딩 (상속하여 재정의) 하여 특정 시점에 코드가 실행되도록 설정한다. 리액트 라이프 사이클은 크게 Mount, Update, Unmount 3가지로 나눈다.

클래스 컴포넌트에만 해당되는 내용이며, 함수형 컴포넌트는 Hook을 사용하여 생명주기

에 원하는 동작을 한다.

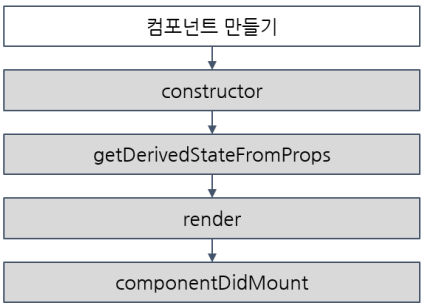


페이지에 컴포넌트가 나타남

페이지에서 컴포넌트가 사라짐

1) Mount (마운트)

DOM이 생성되고 웹 브라우저상에 나타나는 것을 마운트라고 한다. 이때 호출하는 라이프 사이클 메서드는 다음과 같다.



constructor: 컴포넌트를 새로 만들 때마다 호출되는 클래스 생성자 메서드

getDerivedStateFromProps: props에 있는 값을 state에 넣을 때 사용하는 메서드

render: UI를 렌더링하는 메서드

componentDidMount: 컴포넌트가 웹 브라우저상에 나타난 후 호출하는 메서드

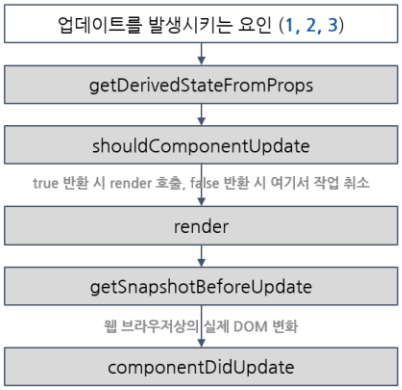
2) Update (업데이트)

컴포넌트는 아래와 같은 총 네 가지 경우에 업데이트 되며 컴포넌트를 업데이트를 할 때 다음 메서드를 호출한다.

1. props 가 바뀔 때 2. state가 바뀔 때

3. 부모 컴포넌트가 리렌더링될 때

4. this.forceUpdate로 강제로 렌더링을 트리거할 때



getDerivedStateFromProps: props로 받아온 것을 state 에 넣어주고 싶을 때 사용하는 메서드 (마운트 과정과 업데이트가 시작되기 전에도 호출됩니다.)

shouldComponentUpdate: 컴포넌트의 리렌더링 여부를 결정하는 메서드로 이 메서드는 true 혹은 false 값을 반환해야 하며 기본적으로 true를 반환합니다. 만약 특정함수 this.forceUpdate() 를 호출한다면 이 과정을 생략하고 바로 render 함수를 호출합니다.

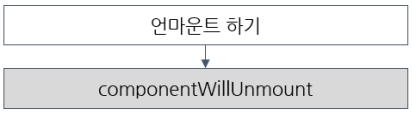
render: 컴포넌트를 리렌더링합니다.

getSnapshotBeforeUpdate: 컴포넌트 변화를 DOM에 반영하기 바로 직전에 호출하는 메서드

componentDidUpdate: 컴포넌트의 업데이트 작업이 끝난 후 호출하는 메서드

3) Unmount(언마운트)

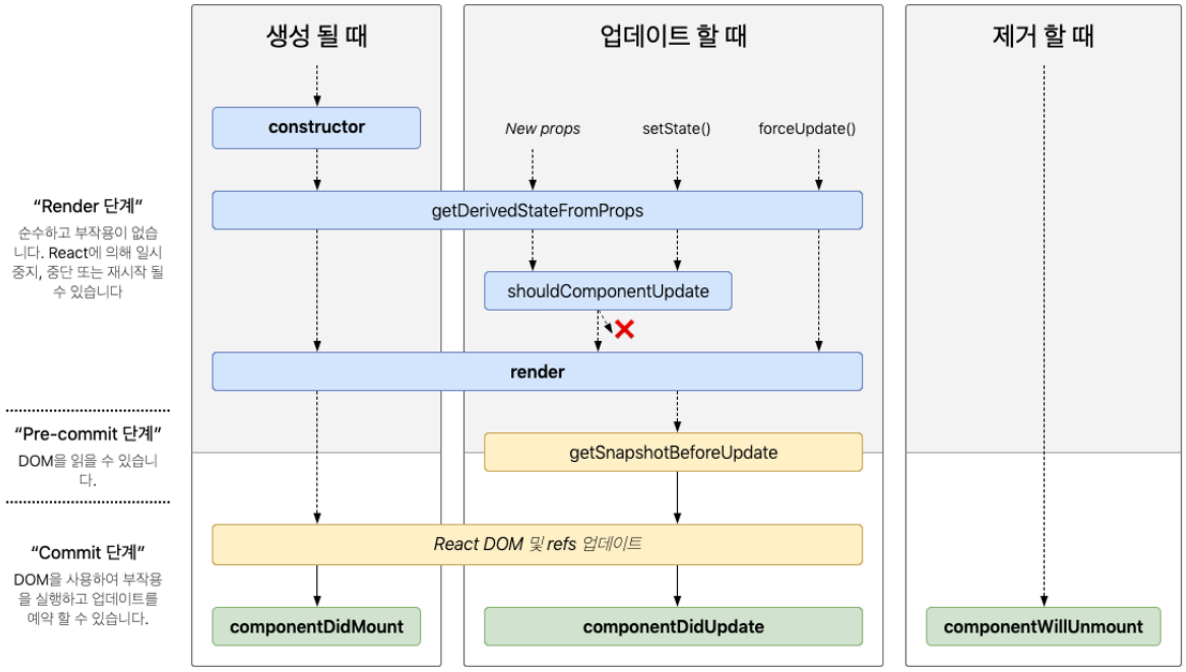
마운트의 반대 과정, 즉 컴포넌트를 DOM에서 제거하는 것을 언마운트라고 한다.



componentWillUnmount: 컴포넌트가 웹 브라우저상에서 사라지기 전에 호출하는 메서드로 등록한 이벤트, 타이머, 직접 생성한 DOM이 있다면 여기서 제거 작업을 해줍니다.

4) 컴포넌트의 라이프 사이클 메서드 흐름

LifeCycle Method는 컴포넌트가 브라우저 상에 나타나고, 업데이트 되고, 사라지게 될 때 호출하는 메서드 들이다. 생명주기 메서드는 클래스형 컴포넌트에서만 사용할 수 있다.



3. React Hooks의 useEffect() 함수

1) Hook란 ?

Hook은 함수 컴포넌트에서 React state와 생명주기 기능(lifecycle features)을 "연동(hook into)"할 수 있게 해주는 함수이다.

class 안에서는 동작하지 않고 함수형 컴포넌트 내부에서 class 없이 React를 사용할 수 있게 해준다.

과거에는 컴포넌트 상태 관리 및 라이프 사이클 제어를 위해 클래스형 컴포넌트를 주로 사용했지만, 2019년 v16.8 부터 리액트 훅 (HOOK)이란 기능이 도입되면서 리액트 공식 문서에서는 함수형 컴포넌트와 훅을 함께 사용할 것을 권장하고 있다.

대표적으로 상태 관리를 할 수 있는 useState, 렌더링 직후 작업을 설정하는 useEffect 등의 기능을 제공한다.

아직 Hook이 class의 모든 사용 사례를 대체하지 못한다. 사용이 드문 클래스형 컴포넌

트 생명주기 메서드인 getSnapshotBeforeUpdate, getDerivedStateFromError, componentDidCatch를 대체할 수 없지만 리액트 공식 문서를 보면 이를 대체할 수 있는 Hook을 곧 추가 개발할 예정이라고 한다.

2) 라이프 사이클 메서드를 대체하는 useEffect()

useEffect란 컴포넌트가 렌더링 될 때마다 특정 작업을 실행할 수 있도록 하는 hook이다.

useEffect는 컴포넌트가 마운트, 업데이트, 언마운트 됐을 때

라이프 사이클 메서드 componentDidmount, getDerivedStateFromProps, componentDidUpdate, componentWillUnmount 처럼 특정 작업을 처리할 수 있다.

Reference

1. https://ko.reactjs.org/docs/hooks-intro.html

2. https://react.vlpt.us/basic/25-lifecycle.html
3. https://born-dev.tistory.com/27
4. https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/
5. https://xiubindev.tistory.com/100
6. https://velog.io/@sukong/REACT-%EB%A6%AC%EC%95%A1%ED%8A%B8%EC%9D%98-%EC%83%9D%EB%AA%85%EC%A3%BC%EA%B8%B0%EC%99%80-useEffect-Hook