리액트 산출물

태그

React CSR/SSR

CSR

- index.html - 빈 div 태그 `<div id="root"></div>`
- index.js - 빈 div 태그에 App.js를 동적 삽입

```
const root = ReactDOM.createRoot(document.getElementById('
root.render(
    <App />
);
```

- 렌더링 후 화면소스에서 App.js의 소스코드는 볼 수 없음 → SEO 불리

```
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
</body>
```

SSR

- 서버 코드 - renderToString()를 통하여 App.js를 문자열로 반환하여 응답

```
app.get("*", (req,res) => {
    const renderString = renderToString(<App/>);
    const result = html
    .replace(
        '<div id="root"></div>',
        `<div id="root">${renderString}</div>`
    ).replace('__DATA_FROM_SERVER__', JSON.stringify(initi
```

```
    res.send(result);
});
```

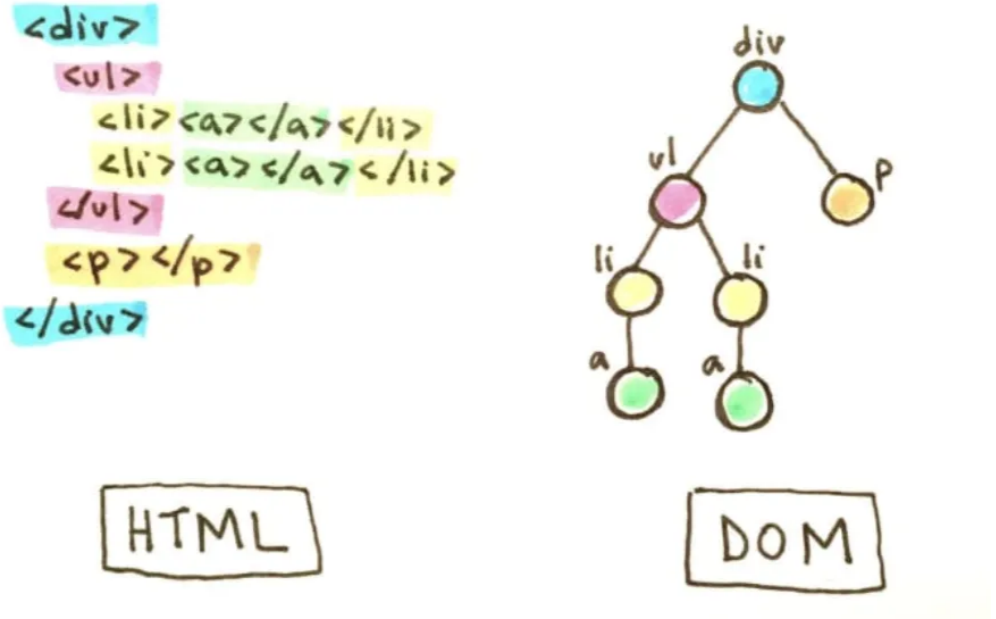
- index.js - render() → hydrate() 변경 (정적 요소에 이벤트 추가)
  - `ReactDom.hydrate(<App />, document.getElementById('root'));`
- 렌더링 후 화면소스에서 App.js의 소스코드를 볼 수 있음 → SEO 유리

```
<body>
    <div id="__next">
        <div id="root">
            <h1>NextJS App</h1>
            <button>count is
            <!-- -->
            0</button>
            <p>Some random content here</p>
        </div>
```

Virtual DOM / Real DOM

Real DOM

- 브라우저에 실제로 렌더링되는 HTML 문서 객체 모델
- 변경사항이 생기는 즉시 웹에 반영됨
- CSSOM, 레이아웃, 이벤트 핸들러와 같은 내부 속성 등 포함



Virtual DOM

- JavaScript 객체로 관리되는 Real DOM의 가벼운 복사본
- Real DOM보다 메모리를 적게 사용 (Real DOM보다 데이터 구조가 간단)
- 필요하지 않은 브라우저 엔진 관련 데이터(스타일, 레이아웃 등)는 포함하지 않음

React Virtual DOM

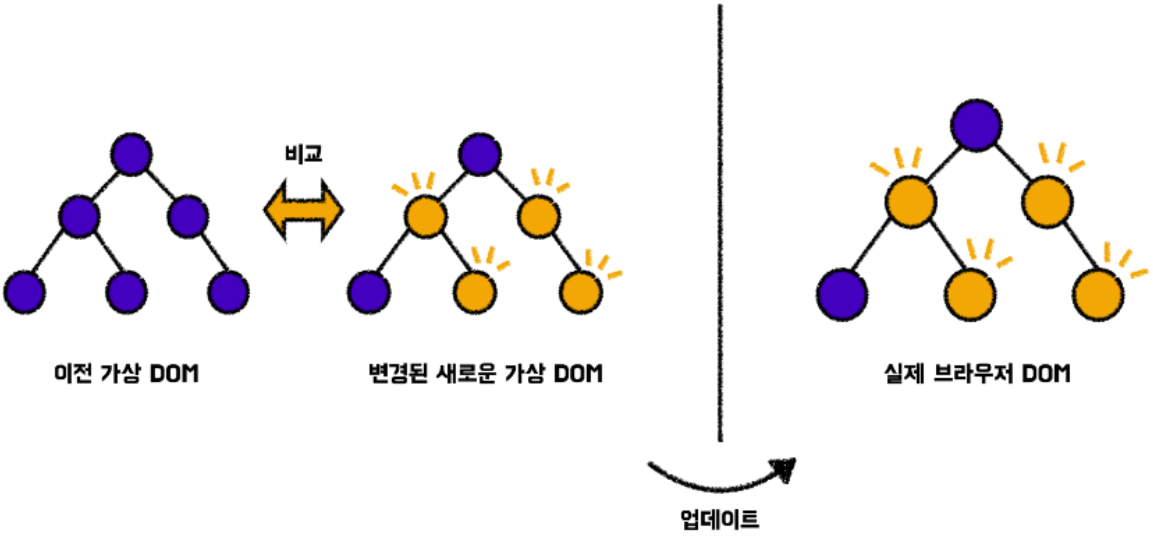
- 2개의 Virtual DOM 사용
  - 현재 Virtual DOM과 업데이트를 적용한 Virtual DOM을 비교(Diffing)하기 위함
  - 두 DOM 간의 차이를 계산(Patching)하고, 필요한 변경사항만 Real DOM에 반영(Batch Update)

리액트의 DOM의 개수는?

Virtual DOM과 Real DOM이 브라우저에 각각 떠 있을 때 메모리 사용량 차이
→ Virtual DOM : 사용량 적음
→ Real DOM : 사용량 많음

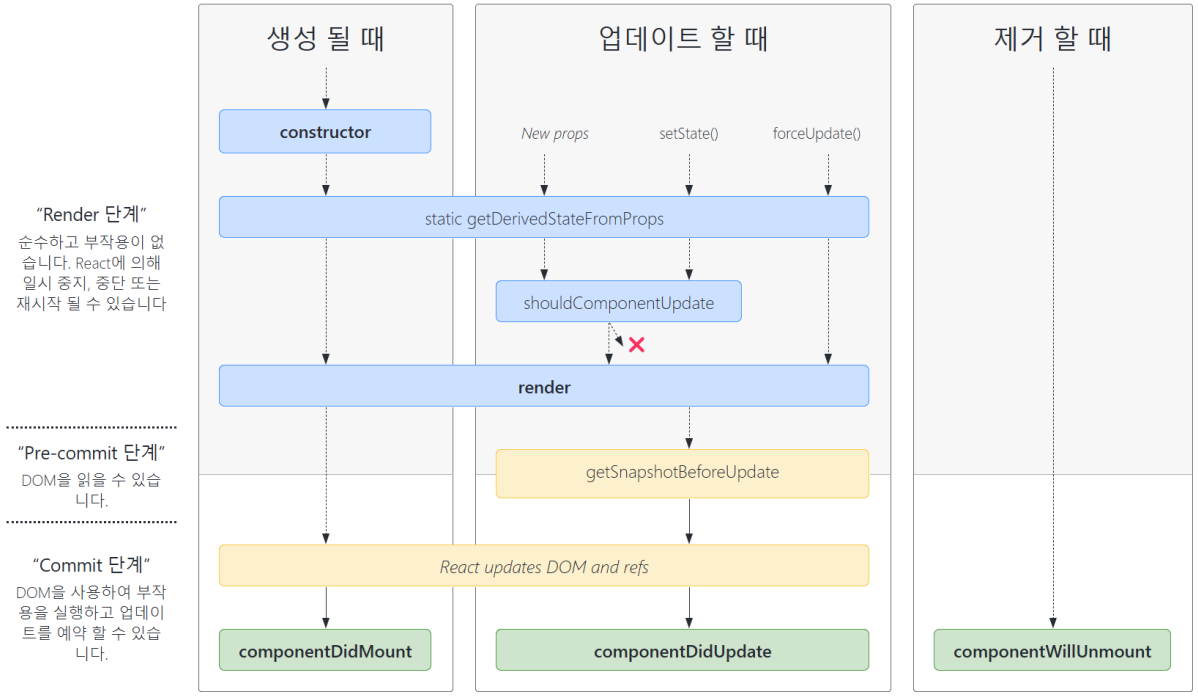
- Virtual DOM과 Real DOM이 브라우저에 각각 떠 있을 때 메모리 사용량 차이
  - Virtual DOM - 사용량 적음
  - Real DOM - 사용량 많음
  - 이유 - Virtual DOM이 Real DOM의 복사본이라고 하더라도, 포함하는 데이터 양이 더 적음
- React(Virtual DOM 2개 + Real DOM 1개)와 Real DOM 1개가 각각 브라우저에 떠 있을 때 메모리 사용량 차이
  - React(Virtual DOM 2개 + Real DOM 1개) - 사용량 많음
  - Real DOM - 사용량 적음

- React의 두 Virtual DOM은 렌더링이 되지 않을 때에도 메모리 상에서 유지되는지 여부
  - 두 Virtual DOM은 계속 메모리에 유지
  - 성능 최적화를 위해 되어있는 설계
- React가 Virtual DOM 2개 + Real DOM 1개를 사용하는 이유
  - 추가적인 메모리 소비를 감수하고도 상태 변화가 많거나 렌더링이 자주 발생하는 상황에서 Real DOM만 사용하는 것보다 효율적



React Life Cycle

- 라이프사이클은 컴포넌트가 생성되고, 업데이트되며, 제거되는 과정에서 호출되는 일련의 단계와 메서드를 말함
- React에서는 이 과정을 통해 개발자가 특정 시점에 원하는 동작을 수행할 수 있도록 제공하며, 컴포넌트의 상태 관리, 외부 API 호출, DOM 조작, 메모리 정리 등을 위해 사용됨



1. Mount 생성

- 컴포넌트가 DOM에 삽입되는 단계로, 초기 렌더링을 준비하고 실행
- 메서드 : constructor(), render(), componentDidMount()

2. Update 업데이트

- 상태(state)나 속성(props)이 변경되어 컴포넌트가 다시 렌더링되는 단계로, 필요한 부분만 DOM을 갱신
- 메서드 : shouldComponentUpdate(), render(), componentDidUpdate()

3. Unmount 제거

- 컴포넌트가 DOM에서 제거되는 단계로, 정리 작업을 수행
- 메서드 : componentWillUnmount()

리액트를 써야하는 이유

- 상호작용이 있는 애플리케이션

    - 자라다 - 사용자와의 상호작용이 있는 애플리케이션으로 사용자에게 만족도 높은 경험을 제공하기 위해 리액트 사용
- 유지보수 인력
    - jQuery과 React 중 요즘 개발자의 선호도
    - 앞으로 jQuery보다 React를 능숙하게 다루는 인력이 더 많아질 것, 새로운 인력에게 기대할 수 있는 기술
    - 한 시스템을 몇십 년 운영한다고 가정했을 때, jQuery를 사용하는 사람은 점점 줄어들 것
- 범용성
    - 멀티플랫폼 환경(냉장고 내 디스플레이 등)에서는 SPA 방식 지향
        - jquery보다 리액트로 SPA를 구현하기 더 쉬움