**1. REALDOM VS VIRTUALDOM 각각 설명하시오**

돔(DOM)이란 Document Object Model의 약자이며, HTML 및 XML 문서를 위한 API이다. 돔은 객체 간의 관계를 지정하지 않았기 때문에 논리적 구조라고 하며, 개발자는 DOM을 사용하여 문서를 작성하고 구조를 탐색하며, 돔에 요소 또는 내용을 추가, 수정, 삭제할 수 있다.

모든 브라우저에는 HTML 문서를 DOM으로 구문 분석하는 DOM Parser(돔 파서)가 있다. 돔 파서는 HTML 페이지를 읽고 해당 데이터를 DOM을 구성하는 객체로 바꾼다. 그리고 논리적으로 DOM Structure Tree로 배열한다.

돔 조작(DOM Manipulation)은 웹 페이지를 수정하기 위해 매우 유용하지만, 문제점도 존재한다. 트리에서 하나의 노드를 추가하거나 제거한다면, 전체 트리를 다시 정렬해야 할 수도 있다. 이것은 비용이 많이 드는 작업이며, 브라우저에서는 시간과 브라우저 리소스가 필요하다.

하나의 노드 추가나 삭제를 하여 웹 페이지 전체의 레이아웃에 영향을 받는 경우, 웹페이지의 일부 또는 전체가 다시 렌더링 될 수 있다. 이런 경우를 Reflow라고 한다. 다시 말하자면 대화식 사이트(interactive site)에서 업데이트한 후에 브라우저가 웹 페이지의 일부 또는 전부를 다시 처리하고 그려야 할 때를 의미한다. 과도한 Reflow를 피하기 위해서는 Dom을 너무 많이 변경하면 안되며, 브라우저에 따라 다른 요소도 브라우저에 영향을 줄 수 있다.

실제 DOM의 변경사항에 대해 DOM에서 수행해야 할 모든 변경 사항을 가상돔(virtual DOM)에서 수행한 다음 실제 DOM에 전달함으로써 위에서 언급한 계산 단계가 줄어든다. 여러 번의 변경사항이 있더라도 모든 변경 사항을 하나로 그룹화하여 한번만 수행한다. 또한, DOM 관리를 자동화하고 추상화하여 직접 할 필요가 없게 해주는 것이다. 또한 전체 DOM Tree를 reload하지 않기 위해 변경한 부분과 변경되지 않는 부분을 직접 할 때는 추적해야 하나 이 또한 가상돔이 자동화 해준다. 마지막으로 DOM 조작 자체를 포기함으로써 DOM을 수정하는 모든 부분 간의 동기화를 피할 수 있다.

출처: <https://code-masterjung.tistory.com/33>

**2. 버츄얼돔은 왜 좋은가. 어떻게 동작하는가. JQUERY의 돔 직접참조에 비해서 무엇이 개선되었는가.**

Virtual DOM 기술은 DOM 조작에 있어 비효율적인 이유에서 나오게 되었다. 기존에는 Javascript 또는 Jquery를 통해 원하는 dom 노드를 찾은 후 변경하는 행위를 했다. 하지만 수천가지의 노드가 존재할 수 있기에, 계산하기 위해 큰 비용을 투자하게 된다. 그 결과, 성능 저하로 인한 페이지 속도 지연이 발생한다.

이러한 배경 속에서 Virtual DOM 이 탄생하게 되었다. Virtual DOM은 코드를 통해 표현할 수 있다. 직접 DOM API를 사용하지 않고, domNode 객체를 활용하게 된다. 이러한 처리는 실제 DOM이 아닌 메모리에 있기 때문에 훨씬 더 빠르다. 즉, 실제 DOM이 아닌 Virtual DOM에 먼저 변경 작업을 해준다. 그리고 변경된 부분만 찾아 실제 DOM에 변경해주는 방식이다. 이로써, 기존 View 방식보다 많은 양의 연산을 줄일 수 있게 된다.

출처: <https://mygumi.tistory.com/190>

**3. 버츄얼돔이 동작하는 DIFF알고리즘에 대해 설명하시오.**

두 개의 트리를 비교할 때, React는 두 엘리먼트의 루트(root) 엘리먼트부터 비교합니다. 이후의 동작은 루트 엘리먼트의 타입에 따라 달라집니다.

엘리먼트의 타입이 다른 경우

두 루트 엘리먼트의 타입이 다르면, React는 이전 트리를 버리고 완전히 새로운 트리를 구축합니다. <a>에서 <img>로, <Article>에서 <Comment>로, 혹은 <Button> to <div>로 바뀌는 것 모두 트리 전체를 재구축하는 경우입니다. 트리를 버릴 때 이전 DOM 노드들은 모두 파괴됩니다. 컴포넌트 인스턴스는 componentWillUnmount()가 실행됩니다. 새로운 트리가 만들어질 때, 새로운 DOM 노드들이 DOM에 삽입됩니다. 그에 따라 컴포넌트 인스턴스는 componentWillMount()가 실행되고 componentDidMount()가 이어서 실행됩니다. 이전 트리와 연관된 모든 state는 사라집니다.

DOM 엘리먼트의 타입이 같은 경우

같은 타입의 두 React DOM 엘리먼트를 비교할 때, React는 두 엘리먼트의 속성을 확인하여, 동일한 내역은 유지하고 변경된 속성들만 갱신합니다. style이 갱신될 때, React는 또한 변경된 속성만을 갱신합니다. DOM 노드의 처리가 끝나면, React는 이어서 해당 노드의 자식들을 재귀적으로 처리합니다.

같은 타입의 컴포넌트 엘리먼트

컴포넌트가 갱신되면 인스턴스는 동일하게 유지되어 렌더링 간 state가 유지됩니다. React는 새로운 엘리먼트의 내용을 반영하기 위해 현재 컴포넌트 인스턴스의 props를 갱신합니다. 이때 해당 인스턴스의 componentWillReceiveProps()와 componentWillUpdate()를 호출합니다. 다음으로 render() 메소드가 호출되고 비교 알고리즘이 이전 결과와 새로운 결과를 재귀적으로 처리합니다.

출처: <https://ko.reactjs.org/docs/reconciliation.html>