**1. REALDOM VS VIRTUALDOM 각각 설명하시오**

**DOM : Document Object Model, 문서객체모델**

문서 객체 모델(DOM, Document Object Model)은 XML이나 HTML 문서에 접근하기 위한 일종의 인터페이스입니다.

이 객체 모델은 문서 내의 모든 요소를 정의하고, 각각의 요소에 접근하는 방법을 제공합니다.

DOM은 트리형태로 되어 있어서 특정 node를 찾을 수도 있고 수정하거나 제거 할 수 있습니다.

이러한 DOM은 W3C의 표준 객체 모델이며, 다음과 같이 계층 구조로 표현됩니다.



**단점** : 큰 규모의 웹 어플리케이션(ex) 트위터,페이스북) 에서는 스크롤을 내리다 보면 정말 수많은 데이터가 로딩이 되고 각 데이터를 표현하는 요소도 많아지게 됩니다. 이와같은 요소의 갯수가 몇백개 몇천개 단위로 많아진 상태에서 DOM에 직접 접근하여 변화를 주다보면 성능상 이슈가 발생하게 됩니다.

(레이아웃 구성을 reflow라고 하고, 색상변경 같은 레이아웃에 관계 없는 것들은 repaint라고 합니다.)

**개선방법** : 이러한 HTML마크업을 시각적인 형태로 변환하는 시간이 드는 것은 어쩔 수 없습니다. 따라서 최소한의 DOM 조작을 통해 작업을 처리하는 방식으로 이를 개선 할 수 있습니다.

그중에 DOM작업을 가상화 하여 미리 처리한 다음 한꺼번에 적용할 수 있는 방법이 있습니다.

**출처 :** [**http://tcpschool.com/javascript/js\_dom\_concept**](http://tcpschool.com/javascript/js_dom_concept) **,** [**https://jw910911.tistory.com/41**](https://jw910911.tistory.com/41)

**Virtual DOM**

실제 DOM 문서를 추상화한 개념으로, 변화가 많은 View를 실제 DOM에서 직접 처리하는 방식이 아닌 Virtual Dom과 메모리에서 미리 처리하고 저장한 후 실제 DOM과 동기화 하는 프로그래밍 개념

**장점** : Virtual Dom은 실제 DOM보다 가볍고, 빠른 Rendering이 가능하기 때문에 압도적으로 DOM의 부담을 줄여줍니다. 또한 몇가지의 특수 키워드(ref, key, htmlFor)등 조금씩 다른부분이 존재하지만 실제 DOM과 Virtual Dom은 구조상 큰 차이가 없어 이해하기 편하다.

**2가지 주의해야할 사항**

**언제 DOM을 다시 렌더링 할것인가?**

데이터가 바뀌면 업데이트할 필요가 있습니다.

데이터가 바뀌었는지 어떻게 알까요?

1. 데이터의 모든 값을 체크

2.상태(state)가 바뀌는지 관찰(observe)

만약 바뀐게 없다면 아무것도 하지 않는다.

**어떻게 효율적으로 할 것인가?**

1. 효과적인 비교 알고리즘.

2. DOM을 읽고 쓰는 작업

3. 효율적으로 서브 트리만 업데이트하는 것

위에 나열한 것들은 쉽게 구현하기 힘들다. 그래서 도움을 받을 라이브러리들이 있다. 제일 유명한 것으로 페이스북의 **React JS**

**React**는 자바스크립트 오브젝트에서 가져온, DOM 트리를 흉내낸 가벼운 트리(lightweight tree)를 만든다. 이것으로 HTML을 만들고 HTML 엘리먼트에 append 나 insert 한다. 브라우저가 다시 렌더링 하게 하는 것

출처 : <https://noogoonaa.tistory.com/53> , <https://3794.tistory.com/entry/%EA%B0%80%EC%83%81-DOM-%EC%9D%B4%EB%9E%80>

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

**장점** : 빠르다. 예로들어 100가지의 요소가 한 페이지내에 있다면, 1개가 변경되었을때 기존 DOM은 100가지를 모두 렌더링하고, 가상돔은 바뀐부분만 업데이트 해주기 때문에 속도측면에서 매우 효율적이다.

**동작방식** : React 라이브러리 기준으로 총 3가지의 업데이트 과정이 있습니다.

1. 데이터가 업데이트 되면, 전체 UI를 Virtual DOM에 리렌더링 합니다.

2. 이전 Virtual DOM에 있던 내용과 현재의 내용을 비교합니다.

3. 바뀐 부분만 실제 DOM에 적용이 됩니다.

(컴포넌트가 업데이트 될때 , 레이아웃 계산이 한번만 이뤄짐)

**JQUERY 직접참조에 비해서 개선된점**

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

기존에는 아래와 같이, Javascript 또는 Jquery를 통해 원하는 dom 노드를 찾은 후 변경하는 행위를 했다.

document.getElementById('myId').appendChild(myNewNode) // javascript

$('#myId').append(myNewNode) // jquery

하지만 수천가지의 노드가 존재할 수 있기에, 계산하기 위해 큰 비용을 투자하게 된다.

그 결과, 성능 저하로 인한 페이지 속도 지연이 발생

<ul id='myId'>

<li>Item 1</li>

<li>Item 2</li>

<ul>

위 태그를 참조하려면 기존 직접참조는 API를 통해 노드를 찾은 후 필요한 노드를 추가.

가상 돔에서는 아래와같이

let domNode = {

tag: 'ul'

attributes: { id: 'myId' }

children: [ // where the LI's would go ]

};

domNode 객체를 Virtual DOM으로 보면 된다.

직접 DOM API를 사용하지 않고, domNode 객체를 활용하게 된다.

이러한 처리는 실제 DOM이 아닌 메모리에 있기 때문에 훨씬 더 빨라지게 된다.

domNode.children.push('<li>Item 3</li>');

출처 : <https://jw910911.tistory.com/41> , <https://mygumi.tistory.com/190>

**3. 버츄얼돔이 동작하는 DIFF알고리즘에 대해 설명하시오.**

**Dom diff 알고리즘**

**다른 타입의 요소일 때**

렌더링된 DOM 트리가 전의 것과 비교해서 다른 타입의 요소일 때에, 전의 것은 없어지고 새로운 것을 완전히 새롭게 렌더링한다.



예를 들어, 위의 경우에는 div와 span이 다른 타입이므로 Counter는 언마운트되고, 새로운 Counter가 마운트된다.

**같은 타입의 DOM 엘리먼트일 때**

바뀐 요소가 같은 타임의 DOM 엘리먼트(div, a, img 등)일 때에는 그 엘리먼트의 property의 바뀐 부분만 감지하여 변경한다.



위와 같은 경우에 div의 className만 변경된 것을 감지하고, 전체를 바꾸지 않고 className만 변경한다.

**같은 타입의 컴포넌트 엘리먼트일 때 자식 엘리먼트 순회**

자식 엘리먼트를 순회하면서 차이가 있을 때마다 변경한다.



예를 들어 엘리먼트 트리의 변화가 위와 같을 때에, 첫번째 <li>first</li>를 비교하고, 두번째 <li>second</li>를 비교하고, 세번째에서 <li>third</li>를 삽입할 것이다.

하지만



위와 같이 자식의 첫 번째에 새로운 엘리먼트가 삽입된 경우에 react는 모든 자식 엘리먼트들을 변경함으로써 낮은 퍼포먼스를 보이게 될 것이다.

**Key**

위와 같은 문제를 해결하기 위해서 **react**는 key 어트리뷰트를 지원한다. React는 key를 통해서 엘리먼트들을 비교한다.



위와 같이 key를 심어주면 react는 <li key="0">Something</li>가 새롭게 추가된 엘리먼트임을 인식할 수 있다.

key는 부모 엘리먼트 안에서만 유일하면 된다 (전역에서 유일할 필요는 없다). 안정적이지 못한 값(Math.random() 등)을 사용하면 성능이 저하될 수 있다.

출처 : <http://guswnsxodlf.github.io/react-diff-algorithm>