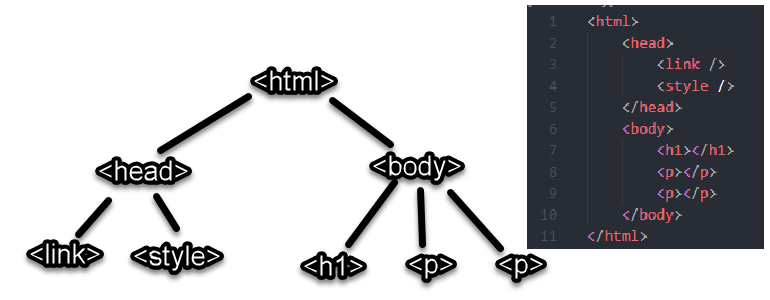
**7월1일**

1. **REALDOM VS VIRTUALDOM 각각 설명하시오.**

**DOM(Real/Broswer DOM)**

돔(DOM)이란 Document Object Model의 약자이며, HTML 및 XML 문서를 위한 API이다. 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 개체로 바꾼다. 그리고 논리적으로 DOM Structure Tree로 배열한다.

  
DOM 트리에서 각 HTML 요소를 Node(노드) 라고 한다. 위에서 말했듯이 DOM을 통해 HTML 문서에 액세스할 수 있으며 조작할 수도 있다. 그리고 DOM Tree에서 노드를 변경하거나 업데이트할 때마다 웹페이지에 반영된다. 돔 조작 시 자바스크립트를 사용해서 <div> 태그의 색상을 업데이트한다고 하면 해당 DOM node 객체에 접근하고 색상 속성을 업데이트하면 된다. 이때는 트리의 나머지 노드에 영향을 미치지 않는다.

그러나 트리에서 하나의 노드를 추가하거나 제거한다면, 전체 트리를 다시 정렬해야 할 수도 있다. 이것은 비용이 많이 드는 작업이며, 브라우저에서는 시간과 브라우저 리소스가 필요하다. 예를 들어 DOM에 5가지 리스트(<li></li>)가 추가된다고 하면, 하나의 리스트마다 새 노드가 DOM에 추가되어 트리가 매번 업데이트된다. 총 5개의 업데이트가 추가되는 것이다. 같은 예로 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. 과도한 Reflow를 피하기 위해서는 Dom을 너무 많이 변경하면 안되며, 브라우저에 따라 다른 요소도 브라우저에 영향을 줄 수 있다. 하지만 대부분의 Javascript 프레임 워크가 DOM을 필요한 것보다 훨씬 많이 업데이트한다. 이러한 현상은 곧 속도 저하로 이어진다. 속도를 향상시키고자 Virtual DOM이라는 것이 등장하였다.

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

**Virtual DOM**

Virtual DOM을 이용하면 실제 DOM에 접근하여 조작하는 대신에, 이를 추상화시킨 자바스크립트 객체를 구성하여 사용한다. 여기서 추상화시킨 자바스크립트 객체는 실제 DOM의 가벼운 사본을 말한다.

Virtual DOM은 쉽게 말해서 DOM의 상태를 메모리에 저장하고 변경 전과 변경 후의 상태를 비교한 뒤 최소한의 내용만 반영하므로 실제 DOM을 조작하는 것보다 성능이 더 향상될 수 있다.

Virtual DOM은 DOM의 상태를 메모리 위에 계속 올려두고 DOM에 변경이 있을 때만 해당 변경을 반영한다.

대표적으로 Virtual DOM을 이용하는 자바스크립트 라이브러리 중에 React에 대해 소개해보려고 한다.

React에서 데이터가 변하여 브라우저 상의 실제 DOM에 업데이트하는 과정은 3가지 절차를 밟는다.

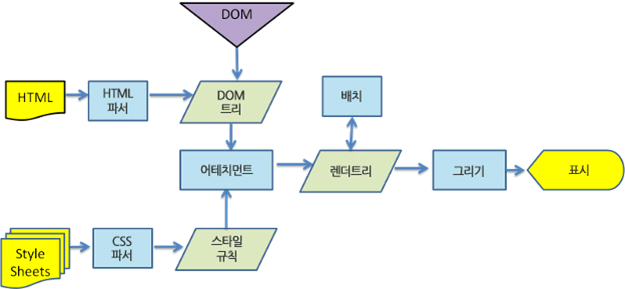
1. 데이터가 업데이트되면, 전체 UI를 Virtual DOM에 리렌더링한다.
2. 이전 Virtual DOM에 있던 내용과 현재의 내용을 비교한다.
3. 바뀐 부분만 실제 DOM에 적용이 된다.

(컴포넌트가 업데이트 될 때, 레이아웃 계산이 한번만 이뤄진다.)

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

**일반적인 웹에서의 DOM 컨트롤**

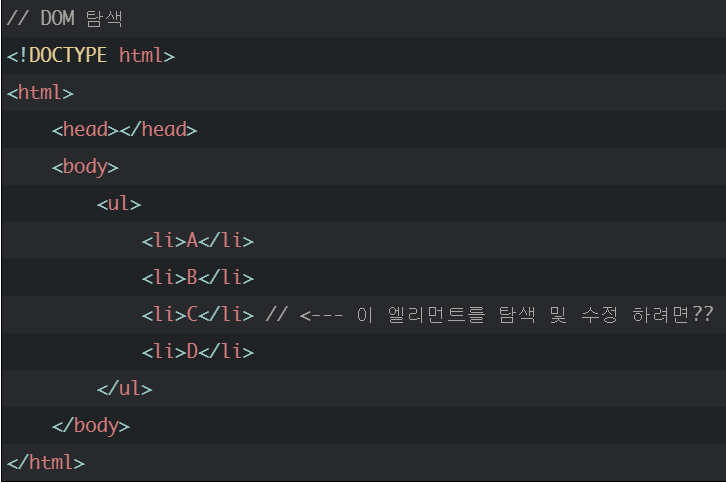
JS, jQuery를 사용하는 프로젝트에서 동적인 효과를 위해 직접 DOM을 탐색, 조작할때  
브라우저는 생각보다 많은 값을 치루게 된다.

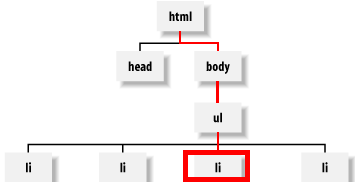


웹킷의 렌더링 동작과정

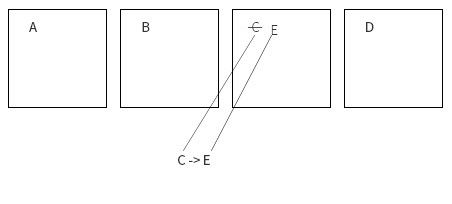
(웹킷이란, 브라우저 렌더링 엔진을 말한다.)

값을 탐색해서 수정하는 경우를 살펴보자





li 값을 찾으려면 html, head, body, ul 그리고 모든 li를 탐색한 후에야 3번째 li를 찾을 것이다.  
탐색이 끝나면 브라우저는 해당 부분을 수정하고 해당부분을 다시 그린다



해당값을 탐색하고 수정하고 다시그리고… 반복.. 이러한 작업에 브라우저는 엄청난 부담을 가지게 된다.

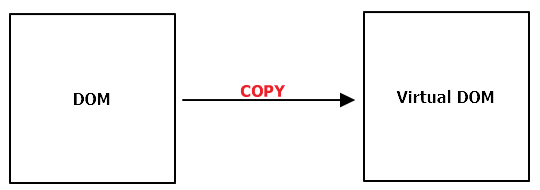


엘리먼트가 많을때 DOM에 변경이 일어나는 경우 브라우저가 치루는 비용은 점점 증가하게 되고 기존까지의 웹에서는 조금이라도 지불하는 값을 줄이고자 각자의 노하우를 통해 최적화를 수행하였다.

이러한 최적화의 노력에도 불구하고 엘리먼트가 많은 웹페이지 (무한 스크롤링 페이지같은?)에서 최적화는 한계를 가지고 있어 Virtual DOM 같은 최적화 방법이 제안되었고 리액트는 Virtual DOM을 사용하고 있다.

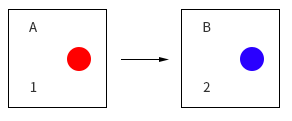
Virtual DOM은 리액트에서 나온 개념은 아니다. 사용자가 직접 Virtual DOM을 생성해서 최적화를 수행하는 방법도 있고, 이 방법이 React를 사용하는 것보다 더 빠른 속도로 최적화시킬 수 있지만 과정이 매우 복잡하다. 이 복잡한 과정을 React가 우리 대신에 제공해준다.

React는 내부적으로 DOM을 복사해서 새로운 가상의 DOM을 생성한다.

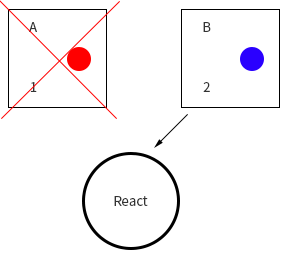


이러한 작업을 React가 대신해줘요!

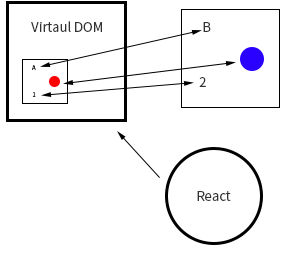
아래와 같이 수정하는 상황을 가정해보자



Virtual DOM을 사용할 때 브라우저는 변경하는 부분을 통째로 렌더링해서 React에게 넘긴다.



변경사항이 react에게 전달되면 Virtual Dom을 이용하여 변경된 내역을 빠르게 탐색하고   
변경된 내역만을 업데이트시킨다.



참고 : <http://hong.adfeel.info/frontend/react/react-%EB%A9%94%EB%AA%A8/>

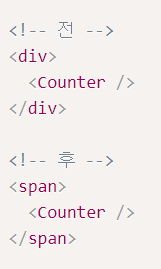
1. **버츄얼돔이 동작하는 DIFF알고리즘에 대해 설명하시오.**

React를 사용할 때, render()함수는 react의 새로운 엘리먼트를 반환한다. state나 prop이 바뀌어서 render()단계에서 다시 새로운 엘리먼트를 반환한다. 여기서 새롭게 렌더링이 될 때, react는 **DOM diff 알고리즘**을 이용해 효율적으로 UI를 업데이트한다.

**DOM diff 알고리즘**

다른 타입의 요소일 때

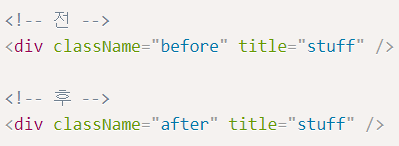
렌더링된 DOM 트리가 전의 것과 비교해서 다른 타입의 요소일 때에, 전의 것은 없어지고 새로운 것을 완전히 새롭게 렌더링한다.



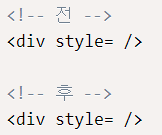
예를 들어, 위의 경우에는 div와 span이 다른 타입이므로 Counter는 언마운트되고, 새로운 Counter가 마운트된다.

같은 타입의 DOM 엘리먼트일 때

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



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

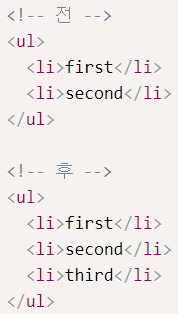


위와 같이 style의 color만 바뀐 경우에 react는 fontWeight를 포함한 style의 전체를 바꾸지 않고, color만 변경한다.

같은 타입의 컴포넌트 엘리먼트일 때

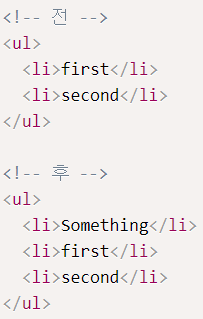
자식 엘리먼트 순회

React는 자식 엘리먼트를 순회하면서 차이가 있을 때마다 변경한다.



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

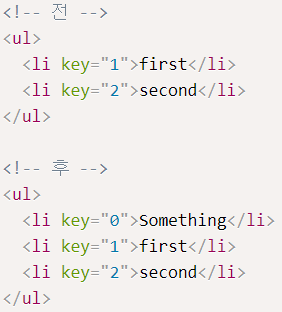
하지만 다음 예제를 보자.



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 react는 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.

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