1. 리액트의 시작

* HTML / JS 와의 비교

<h2 id="number">0</h2>

<div>

<button id="increase">+1</button>

<button id="decrease">-1</button>

</div>

* 위는 HTML 구성 : id를 사용하여 각 DOM을 선택한 뒤, 원하는 이벤트가 발생하면 DOM의 특정 속성을 바꿔 주어야 함.

const number = document.getElementById('number');

const increase = document.getElementById('increase');

const decrease = document.getElementById('decrease');

increase.onclick = () => {

const current = parseInt(number.innerText, 10);

number.innerText = current + 1;

};

decrease.onclick = () => {

const current = parseInt(number.innerText, 10);

number.innerText = current - 1;

};

* 위는 JavaScript
* 대부분의 경우 웹 애플리케이션의 규모가 커지면, DOM을 직접 건드리면서 작업을 하면 코드가 난잡해지기 쉽다. 처리해야할 이벤트도 다양해지고, DOM 도 다양해지게 된다면 이에 따라 업데이트를 하는 규칙도 많이 복잡해진다.
* 리액트를 사용해보자.
* 리액트의 발상 : 어떠한 상태가 바뀌었을 때, 그 상태에 따라 DOM을 어떻게 업데이트 할 규칙을 정하는 것이 아닌 **다 날려버리고 처음부터 모든걸 새로 만들어서 보여줘 볼까?**
* **Virtual DOM**을 사용 : 리액트는 상태가 업데이트 되면, 업데이트가 필요한 곳의 UI를 Virtual DOM을 통해 렌더링, 그후 리액트 개발 팀이 만든 매우 효율적인 **비교 알고리즘**을 통해 실제 브라우저에 보여지고 있는 DOM과 비교를 한 후, **차이가 있는 곳을 감지**하여 실제 DOM에 패치
* 업데이트를 어떻게 할지에 대한 고민이 필요 없어짐, 빠른 성능을 지킬 수 있게 됨.