**클래스형 컴포넌트에서만 쓸 수 있지만, 일단 배워두는 LifeCycle Method**

1. 매뉴얼 : <https://ko.reactjs.org/docs/react-component.html>
2. LifeCycle Method : 컴포넌트가 브라우저상에 나타나고, 업데이트되고, 사라지거나 에러가 났을 때 호출되는 메소드
3. useEffect와 은근히 비슷하다. 물론 작동방식은 많이 다름.
4. 앞으로 쓸 일이 많이 없으므로 너무 시간을 쏟진 말자.
5. 마운트 될 때
   1. constructor : 컴포넌트의 생성자 메소드. 컴포넌트가 만들어지만 가장 먼저 실행됨

constructor(props) {

super(props);

console.log(“constructor”);

}

* 1. getDerivedStateFromProps : props로 받아온 것을 state에 넣어주고 싶을 때 사용함

static getDerivedStateFromProps(nextProps, prevState) {

if(nextProps.color !== prevState.color) {

return { color: nextProps.color };

}

return null;

}

* 다른 생명주기 메소드와 달리 static을 필요로 함
* 이 안에서는 this 를 조회할 수 없음
* 특정 객체를 반환하면 해당 객체 안에 있는 내용들이 컴포넌트의 state가 됨
* null을 반환하면 아무일도 발생하지 않음
* 컴포넌트가 처음 렌더링 되기 전에, 리렌더링 되기전에 매번 실행
  1. render : 컴포넌트를 렌더링하는 메소드
  2. componentDidMount : 첫번째 렌더링을 마치면 호출되는 메소드. 주로 D3, masonry처럼 DOM을 사용해야하는 외부 라이브러리 연동을 하거나 해당 컴포넌트에서 필요로하는 데이터를 요청하기 위해 axios, fetch등을 통하여 ajax요청을 하거나 DOM의 속성일 읽거나 직접 변경하는 작업 수행

1. 업데이트
   1. getDerivedStateFromProps : 컴포넌트의 props나 state가 바뀌었을 때도 호출됨
   2. shouldComponentUpdate : 컴포넌트가 리렌더링 할지말지를 결정하는 메소드  
      최적화시 주로 사용됨
   3. render
   4. getSanpshotBeforeUpdate : 컴포넌트에 변화가 일어나기 직전의 DOM 상태를 가져와서 특정값을 반환하면 그 다음에 발생하는 componentDidUpdate 함수에서 받아와서 사용할 수 있음

* 함수형 컴포넌트 + Hooks를 사용할 때는 이걸 대체할 수 있는 기능이 아직 없음
* 변화되기 직전 DOM의 속성을 확인하고 싶을 때, 클래스형 컴포넌트로 이걸 사용
  1. componentDidUpdate : 리렌더링이 마치고, 화면에 우리가 원하는 변화가 모두 반영되고 난 뒤 호출되는 메소드. getSanpshotBeforeUpdate에서 반환된 값 조회가능

1. 언마운트
   1. componenetWillUnmount

* 컴포넌트가 화면에서 사라지기 직전에 호출됨

componentWillUnmount() {

console.log(“componentWillUnmount”);

}

* 여기서는 주로 DOM에 직접 등록했었던 이벤트를 제거함
* 만약 setTimeout 을 걸은 것이 있다면 여기서 clearTimeout을 통하여 제거함
* 외부라이브러리를 사용한 것이 잇고, 해당 라이브러리에 dispose 기능이 있으면 여기서 호출해줌

