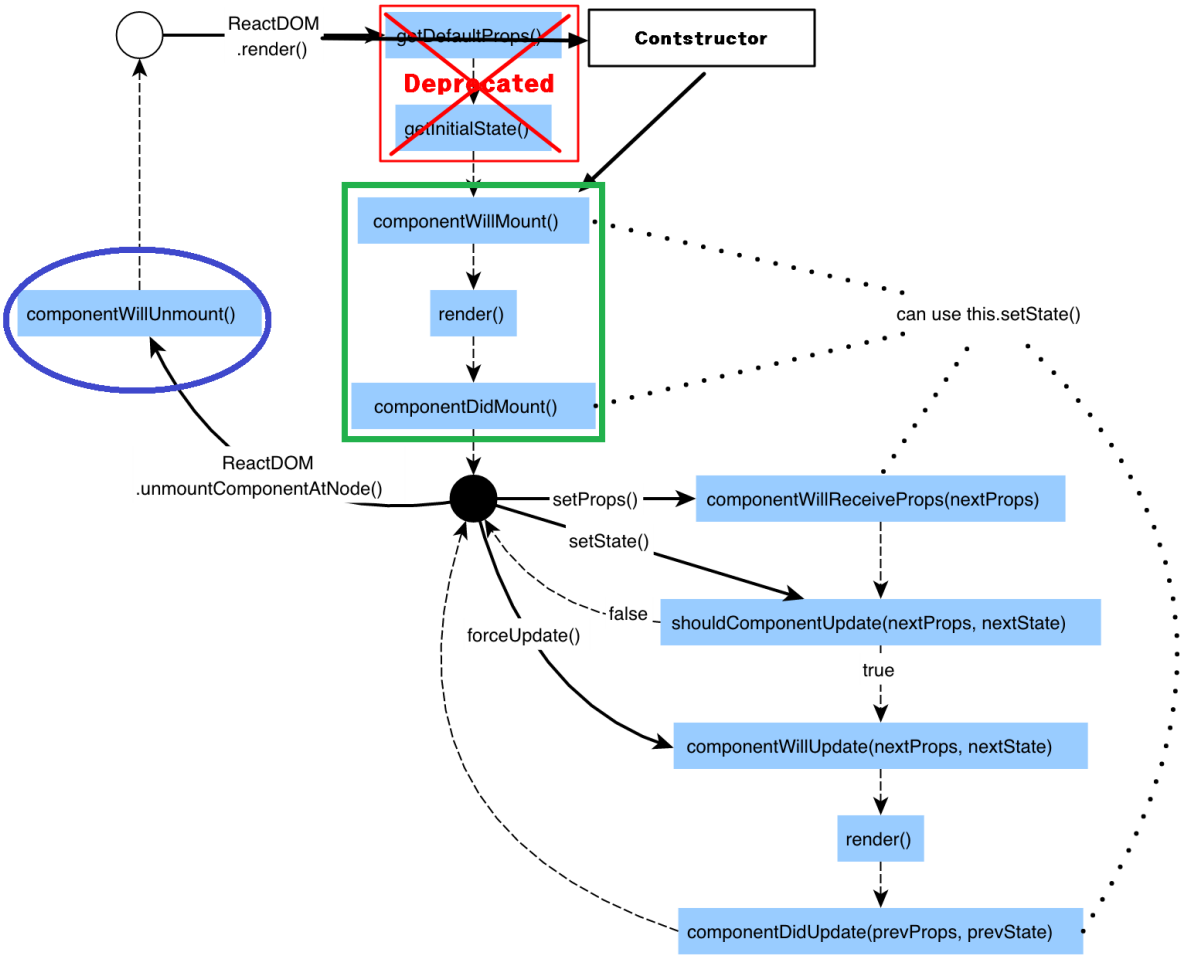
6. React_LifeCycle_Functional_Component_useEf (March 9th, 21)



class 컴포넌트에서,

componentWillMount, render, componentDidMount가 컴포넌트가 등장할 때 쓰이는 한 셋트라고 한다면,

componentWillUnmount는 **컴포넌트가 소멸할 때 쓰이는** method다.

functional 컴포넌트에서 이런 componentWillUnmount를 사용하는 방법에 대해서 이야기한다.

functional 컴포넌트가 소멸할 때 componetWillUnmount와 같은 기능을 사용하기 위해서,

useEffect에 이를 위한 return 함수를 설정한다.

```
function FuncComp(props){
  var numberState = useState(props.initNumber);
  var number = numberState[0];
  var setNumber = numberState[1];
  // var dateState = useState((new Date()).toString());
  // var _date = dateState[0];
```

```
  // var setDate = dateState[1];
  var [_date, setDate] = useState((new Date()).toString());

  useEffect(function(){
    console.log('%cfunc => useEffect'+(++funcId), funcStyle);
    document.title = number + ' : ' + _date;

    return function(){    #functional 컴포넌트의 componentWillUnmount기능을 하는
                           함수를 만들어서 useEffect의 return으로 설정
    }
  });

  console.log('%cfunc => render'+(++funcId), funcStyle);

  return(
    <div className="container">
      <h2>function style component</h2>
      <p>Number : {number}</p>
      <p>Date : {_date}</p>
      <input type="button" value="random" onClick={
          function(){
            setNumber(Math.random());
          }
        }></input>

      <input type="button" value="date" onClick={
          function(){
            setDate((new Date()).toString());
          }
        }></input>
    </div>
  );
}
```

```
func => useEffect3
▶ src\App.js                    react devt
  Line 1:8:  'logo' is defined but neve
vars
func => render4
func => useEffect componentWillUnmount
func => useEffect6
```

random버튼을 눌러서 state를 변화시키면, render가 되고, component에 대해 useEffect가 한 번 더 실행되기 전에 componentWillUnmount가 실행되어 컴포넌트를 정리할 수 있는 기회를 제공하고, 이걸 **CleanUp** 이라고 부른다.

정리(clean-up)를 이용하는 Effects

https://ko.reactjs.org/docs/hooks-effect.html