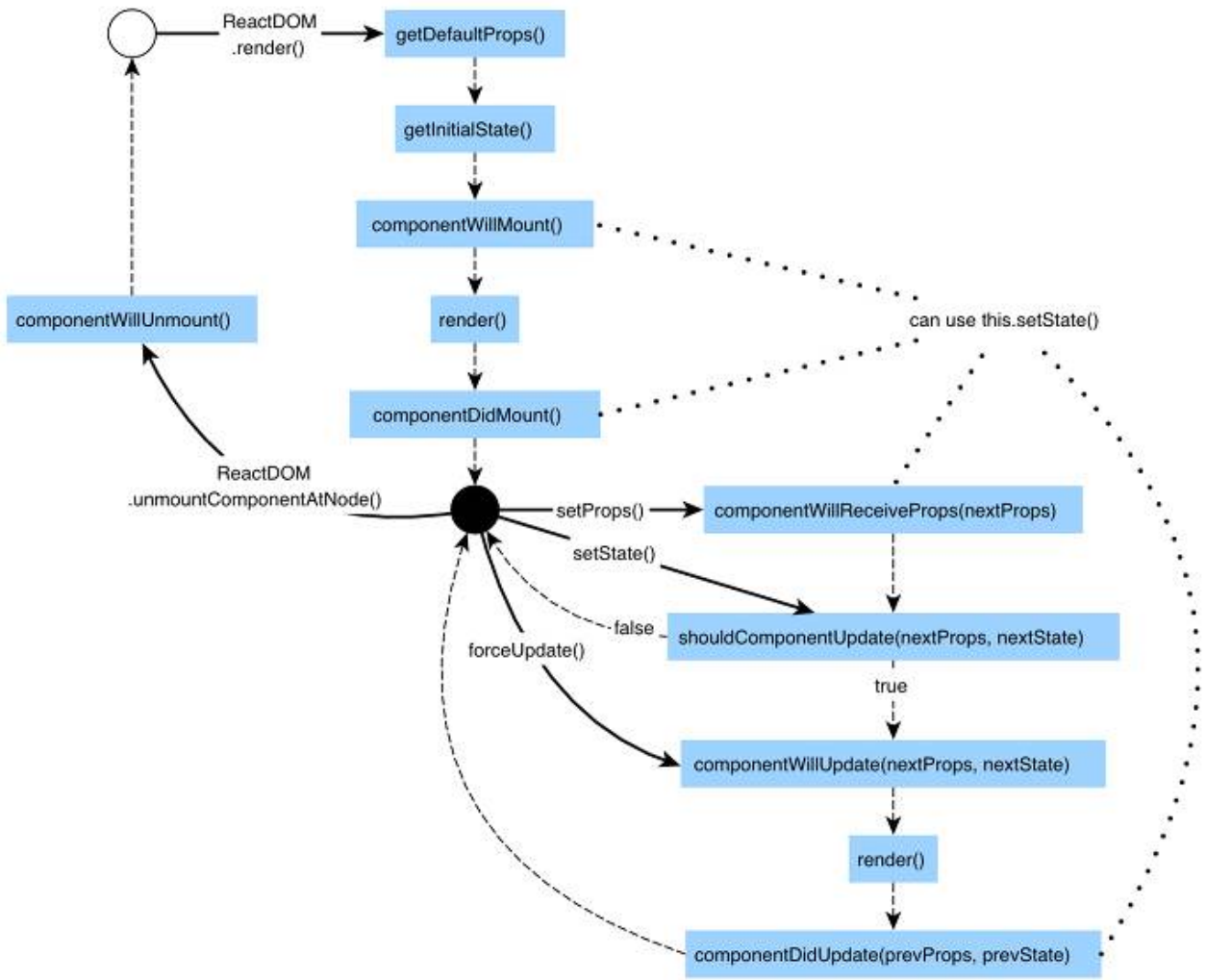
컴포넌트 라이프 사이클

2023.06.29.

01. Class Component Life Cycle
02. Function Component Life Cycle

01. Class Component Life Cycle

2023.06.29.



1. Mount : 컴포넌트가 DOM에 장착되는 것
 ① 저장 - state, context, defaultProps의 정보 저장
 ② ComponentWillMount - 컴포넌트가 장착될 예정 입니다.
 ③ render - ①에서 받은 정보로 엘리먼트를 다 그렸습니다.
 ④ componentDidMount - 정상적으로 장착되었습니다. DOM 접근 가능한 단계 입니다.

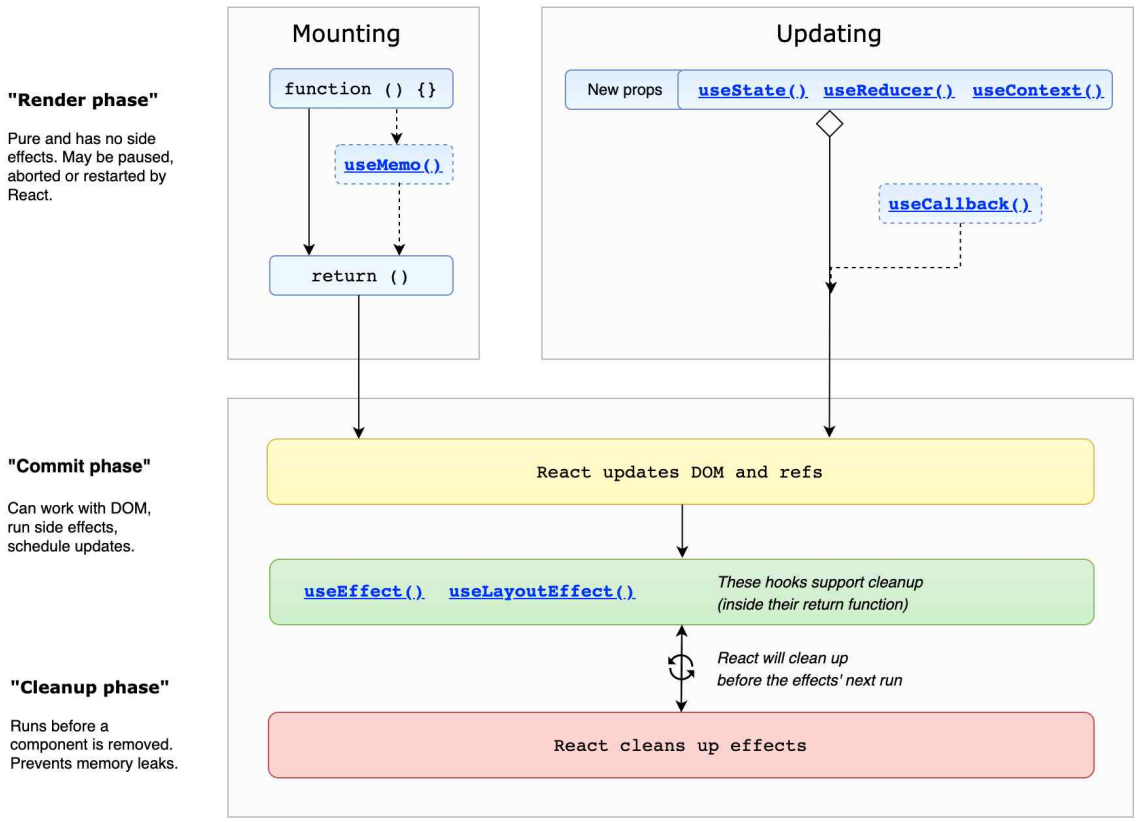
2. Props Update : props가 변경됨을 감지함
 ① componentWillReceiveProps - props를 받을 예정 입니다.
 ② shouldComponentUpdate - 컴포넌트를 업데이트 해야 합니다.
 ③ componentWillUpdate - 컴포넌트가 업데이트 될 예정 입니다.
 ④ re-render - 엘리먼트가 다시 그려졌습니다.
 ⑤ componentDidUpdate - 업데이트 완료

3. State Update : state가 변경됨을 감지 함
 ① shouldComponentUpdate
 ② componentWillUpdate
 ③ render
 ④ componentDidUpdate

4. UnMount : 컴포넌트가 DOM에서 제거됨
 ① componentWillUnmount - 제거예정

02. Function Component Life Cycle

2023.06.29.



1. useEffect()

- 컴포넌트가 랜더링 될 때마다 특정 작업을 실행할 수 있도록 하는 Hook
- component가 mount 됐을 때, component가 unmout 됐을 때, component가 update 됐을 때 특정 작업을 처리할 수 있다.

구문

```
import {useEffect} from 'react';

function 컴포넌트명({프롭스명}) {
  useEffect(function, deps);

  return ( ... );
}
```

- function : 수행하고자 하는 작업
- deps : 배열 형태이며, 배열 안에는 감시하고자 하는 특정 값 또는 빈 배열 dependency

1.1 Component가 mount 됐을 때(처음 나타났을 때)

- 컴포넌트가 화면에 가장 처음 랜더링 될 때 한 번만 실행하고 싶을 경우 deps위치에 빈 배열을 넣는다.

```
useEffect(() => {
  console.log("마운트 될 때만 실행");
}, []);
```

1.2 리랜더링 할 때 마다 실행하고 싶을 경우
- deps 위치의 배열을 생략한다.

```
useEffect(() => {
  console.log("랜더링 될 때 마다 실행");
});
```

1.3 Component가 Update 될 때(특정 props나 state가 바뀔 때)
- deps 위치 배열 안에 검사하고자 하는 값을 넣어준다.
- 마운트 될 때도 실행 된다.

```
useEffect(() => {
  console.log("특정 값이 변경 될 때 마다 실행");
}, [변수, 변수, ...]);
```

1.4 Component가 unmount 될 때(사라질 때) 또는 update 되기 전에
- cleanup 함수 반환(return 문에 포함된 함수 이며 useEffect에 대한 뒷정리 함수라고 한다.)
- unmount 될 때만 cleanup 함수를 실행하고 싶을 때 : 두 번째 파라메타로 빈 배열을 넣는다.
- 특정 값이 update 되기 직전에 cleanup 함수를 실행하고 싶을 때 : deps 배열 안에 검사하고 싶을 값을 넣는다.

```
useEffect(() => {
  console.log("useEffect가 실행 됨");

  return () => {
    cleanup 함수 내용
  };
}, []);
```