4

2024 GETIT 5기 SW 교육(리액트) 4차시

state, 생명주기, hook

GETIT 5기 운영진 이은새

참고사항

해당 강의자료는 MacOS를 기준으로 제작되었습니다.

도움이 필요하거나 질의사항이 있으시다면
이은새 운영진에게 개인적으로 연락하거나
카카오톡 오픈채팅을 이용해주세요!
GETIT 5기 질문방 : https://open.kakao.com/o/gc7jPrhg

해당 강의자료는 [이인제-소플의 처음 만난 리액트]를 참고하여 제작되었습니다.
5기 부원 외의 강의자료 및 영상 배포는 원칙적으로 금지합니다.

목차

01

state와 생명주기

1

state란?

- 리액트 컴포넌트의 변경가능한 데이터
- state가 변경될때마다 재렌더링됨. 따라서 필요한 경우가 아니라면 state로 정의하지 말아야 함.
  - 쓸데없는 재렌더링으로 성능이 저하될 수 있기 때문
- 쉽게 말해서, 변수이다. 단, 값이 변했을 때 관련 컴포넌트들이 재렌더링됨.
- 변경 가능한 데이터를 state를 통해 관리하는 것

1

왜 state가 있어야 하나?

- 일반적인 변수의 경우, 그 값이 변해도 화면에는 반영되지 않음.
- 하지만 값이 변경될 때 화면이 바뀌어야 하는 경우가 있음
- 때문에 그러한 변수들은 특별히 state라는 객체를 통해 관리하는 것

1

props와 state의 차이점

- props는 함수의 매개변수처럼 컴포넌트에 전달됨
- state는 함수 내에 정의된 변수처럼 관리됨

2

state를 변경하고 싶다면?

- state를 변경하고 싶다면 setState()를 사용해야 함
  - 클래스 컴포넌트의 경우, constructor(생성자)에서 state를 정의하고 setState를 통해 변경
- state는 직접적인 변경이 불가능하기 때문이다.

2

setState

- state를 변경하는 메서드
- 함수 컴포넌트에서는 useState를 많이 사용함
- 비동기적으로 동작한다.

2

동기, 비동기

동기(Synchronous)

– 작업이 순차적으로 진행됨.(이전 작업 끝나야 다음 작업)

–코드 흐름이 직관적이고, 디버깅하기 쉬움

– 블로킹(작업 중단)이 발생한다.

–차례대로 진행되므로 성능 저하가 나타날 수 있음

비동기(Asynchronous)

– 여러 작업이 동시에 진행되는 것(순서 보장x)

–코드의 흐름이 복잡하고 이해하기 어려움

– 블로킹이 발생하지 않는다.

–성능 저하를 방지할 수 있음

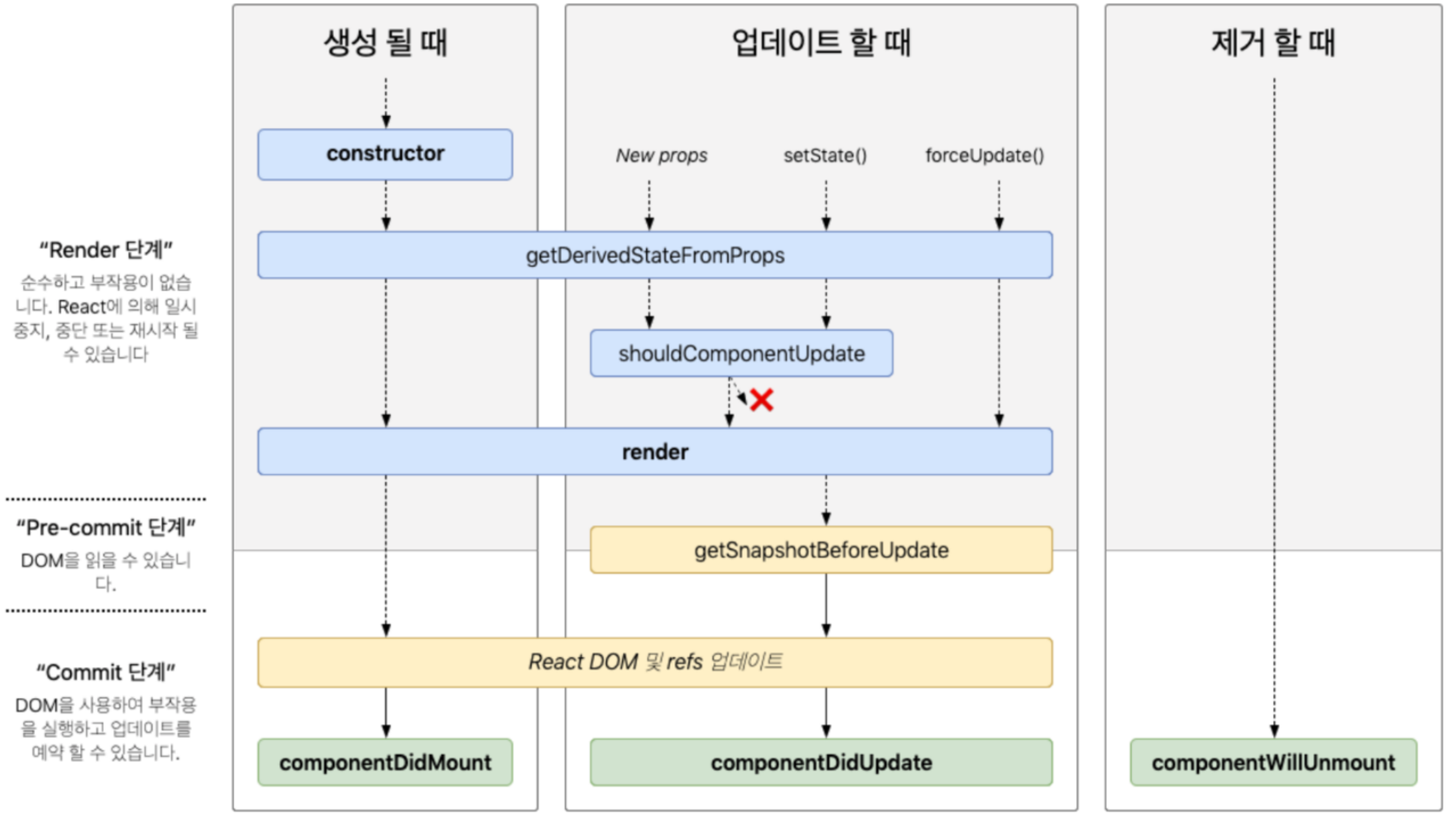
3

생명주기

- 사람이 태어나고 살다가 죽는 것처럼, 컴포넌트도 생성되고 사라집니다.
- 이러한 컴포넌트의 생성~소멸을 '생명주기' 라고 합니다.
- 컴포넌트는 계속 있는 게 아니라, 생성되고 업데이트되다가 사라집니다.

3

생명주기


생성 될 때
업데이트 할 때
제거 할 때
constructor
New props
setState()
forceUpdate()
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React DOM 및 refs 업데이트
componentDidMount
componentDidUpdate
componentWillUnmount
"Render 단계"
순수하고 부작용이 없습니다. React에 의해 일시 중지, 중단 또는 재시작 될 수 있습니다
"Pre-commit 단계"
DOM을 읽을 수 있습니다.
"Commit 단계"
DOM을 사용하여 부작용을 실행하고 업데이트를 예약 할 수 있습니다.

02

hook

1

이전에는 함수 컴포넌트에서 state를 정의하여 사용하거나 컴포넌트의 생명주기에 맞춰 어떤 코드가 실행할 수 없었다.

클래스 컴포넌트에서만 가능했었음.

1

하지만 'hook'이 생기면서

함수 컴포넌트에서도 클래스 컴포넌트에서만

쓰던 기능들을 쓸 수 있게 되었습니다.

2

hook

- 리액트 컴포넌트에서 상태(state)와 생명주기 (lifecycle) 기능 등을 사용할 수 있게 해주는 함수
- 모두 'use'라는 단어로 시작함
  - ex) useState, useEffect, useRef 등
- hook의 도입으로 함수 컴포넌트에서도 이러한 기능을 활용할 수 있게 된 것

2

대표적인 hook

useState

– 컴포넌트에서 상태를 관리하기 위한

useEffect

– 컴포넌트의 생명주기 중 특정 시점에 코드를 실행하기 위한

3

useState

- state를 관리하기 위한 훅

- 예시

```
const [변수명, set변수명] = useState(초깃값);
```

-useState()안에 값을 넣어서 state의 초깃값을 설정

-상태 변수를 바꾸려면 해당 state의 set함수에 값을 넣어서 변경

3

```jsx
import React, { useState } from 'react';

function Counter() {
  // useState를 사용하여 상태 초기값을 0으로 설정하고,
  // 상태와 상태 변경 함수를 반환합니다.
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>현재 카운터 값: {count}</p>
      {/* 버튼을 클릭할 때 상태를 변경합니다. */}
      <button onClick={() => setCount(count + 1)}>카운터 증가</button>
    </div>
  );
}

export default Counter;
```

4

useEffect

– 사이드 이펙트를 수행하기 위한 훅

–컴포넌트에서 제공하는 생명주기 함수인

componentDidMount(), componentDidUpdate(),

componentWillUnmount()의 기능을 하나로 통합해서 제공

– 예시

useEffect(이펙트 함수, 의존성 배열);

–배열에 있는 변수 중에 하나라도 변경되면 이펙트 함수 실행

4

1. 의존성 배열이 빈 배열([])인 경우

–마운트, 언마운트 시에만 이펙트 함수 실행

2. 의존성 배열을 생략한 경우

–렌더링 할때마다 이펙트 함수 실행

3. 의존성 배열에 변수가 있는 경우

–변수들 중 하나가 바뀌면 이펙트 함수 실행

4

1. 의존성 배열이 빈 배열([])인 경우

-마운트, 언마운트 시에만 이펙트 함수 실행

```
import React, { useEffect, useState } from 'react';

function MyComponent() {
  useEffect(() => {
    console.log('컴포넌트가 마운트되었습니다.');

    return () => {
      console.log('컴포넌트가 언마운트되었습니다.');
    };
  }, []); // 마운트 시 한 번 실행, 언마운트 시 클린업 실행

  return <div>My Component</div>;
}
```

4

2. 의존성 배열을 생략한 경우

-렌더링 할때마다 이펙트 함수 실행

```
const [count, setCount] = useState(0);


useEffect(() => {
  console.log('매 렌더링 때 실행');
}); // 의존성 배열을 생략하면 모든 렌더링마다 실행
```

4

3. 의존성 배열에 변수가 있는 경우

-변수들 중 하나가 바뀌면 이펙트 함수 실행

```
const [count, setCount] = useState(0);


useEffect(() => {
  console.log(`value가 ${value}로 변경되었습니다.`);
}, [value]); // value가 변경될 때마다 실행
```

5

hook의 규칙

1. 최상위 레벨에서 호출하기

- 조건문이나 반복문 안에 직접 넣으면 안 됨
- 중첩된 함수 안에서도 X

2. 리액트 컴포넌트에서 호출

- 일반적인 자바스크립트 함수에서 호출 X
- 함수 컴포넌트나 커스텀 훅에서만 사용 가능

과제

- useState와 useEffect를 이용한 과제
- 어떤 식으로 상태 관리하고 생명주기 함수 기능을 활용할 수 있는지
- 컴포넌트 타이머 만들기
- 컴포넌트의 마운트, 업데이트, 언마운트 알아보기

GETIT 5기 SW 교육 4차시 과제

실습 결과 캡쳐 후 노션에 업로드하기

총 1장의 스크린샷을 Notion 페이지에 올려주세요!