

PureComponent

⚠️ 주의하세요!

컴포넌트를 클래스 대신 함수로 정의하는 것을 권장합니다. [마이그레이션 방법](#).

PureComponent 는 [Component](#) 와 비슷하지만 같은 **props**와 **state**에 대해서 다시 렌더링하지 않는다는 점에서 다릅니다. 클래스 컴포넌트를 계속 사용할 수 있지만 새로운 코드에서는 클래스 컴포넌트 사용을 추천하지 않습니다.

```
class Greeting extends PureComponent {  
  render() {  
    return <h1>Hello, {this.props.name}!</h1>;  
  }  
}
```

- [레퍼런스](#)

- [PureComponent](#)

- [사용법](#)

- [클래스 컴포넌트에서 불필요한 재 렌더링 건너뛰기](#)

- [대안](#)

- [PureComponent 클래스 컴포넌트에서 함수 컴포넌트로 마이그레이션 하기](#)

레퍼런스

PureComponent

같은 props와 state에 대해서 다시 렌더링하지 않으려면 `Component` 대신 `PureComponent` 를 extend 해주세요.

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

class Greeting extends PureComponent {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

`PureComponent` 는 `Component` 의 모든 API를 지원하는 `Component` 의 서브클래스입니다. `PureComponent` 를 extend 하는 것은 단순히 props와 state를 비교하는 사용자 `shouldComponentUpdate` 메서드를 정의하는 것과 같습니다.

아래 예시 보기.

사용법

클래스 컴포넌트에서 불필요한 재 렌더링 건너뛰기

React는 일반적으로 부모가 다시 렌더링 될 때마다 자식 컴포넌트도 다시 렌더링 합니다. 하지만 `PureComponent` 를 extend 하여 새 props 및 state가 이전 props 및 state와 같다면 부모가 다시 렌더링 되더라도 자식 컴포넌트는 다시 렌더링 되지 않도록 `Class component`를 최적화할 수 있습니다.

```
class Greeting extends PureComponent {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

React 컴포넌트에는 항상 **순수한 렌더링 로직**이 있어야 합니다. 즉, props, state 및 context가 변경되지 않은 경우 같은 출력을 반환해야 합니다. PureComponent 를 사용하면 컴포넌트가 이 요구 사항을 준수한다고 React에게 알리므로 props 및 state가 변경되지 않는 한 React는 다시 렌더링하지 않습니다. 그러나 사용 중인 context가 변경된다면 컴포넌트는 다시 렌더링 됩니다.

이 예시에서 Greeting 컴포넌트는 name 이 변경될 때마다 다시 렌더링 되지만 (props 중 하나이기 때문에) address 가 변경될 때에는 다시 렌더링 되지 않습니다 (Greeting 에 prop으로 전달되지 않기 때문에).

App.js

다운로드 새로고침 Clear 포크

```
import { PureComponent, useState } from 'react';

class Greeting extends PureComponent {
  render() {
    console.log("Greeting was rendered at", new Date().toLocaleTimeString());
    return <h3>Hello{this.props.name && ', '}{this.props.name}!</h3>;
  }
}

export default function MyApp() {
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
```

▼ 자세히 보기

⚠️ 주의하세요!

컴포넌트를 클래스 대신 함수로 정의하는 것을 권장합니다. [마이그레이션 방법](#).

대안

PureComponent 클래스 컴포넌트에서 함수 컴포넌트로 마이그레이션 하기

새로운 코드에서는 [클래스 컴포넌트](#) 대신 함수 컴포넌트 사용을 권장합니다. PureComponent 를 사용하는 기존 클래스 컴포넌트가 있는 경우 다음과 같이 변환할 수 있습니다.

아래는 기존 코드 입니다.

App.js

📄 다운로드 📌 새로그침 ✕ Clear 🔄 포크

```
import { PureComponent, useState } from 'react';

class Greeting extends PureComponent {
  render() {
    console.log("Greeting was rendered at", new Date().toLocaleTimeString());
    return <h3>Hello{this.props.name && ', '}{this.props.name}</h3>;
  }
}

export default function MyApp() {
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
```

▼ 자세히 보기

이 컴포넌트를 클래스 컴포넌트에서 함수 컴포넌트로 변환할 때 `memo` 로 감싸면 됩니다.

App.js

다운로드 새로고침 Clear 포크

```
import { memo, useState } from 'react';

const Greeting = memo(function Greeting({ name }) {
  console.log("Greeting was rendered at", new Date().toLocaleTimeString());
  return <h3>Hello{name && ', '}{name}!</h3>;
});

export default function MyApp() {
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
  return (
    <>
```

자세히 보기

☐ 중요합니다!

PureComponent 와 달리 `memo` 는 새 state와 이전 state를 비교하지 않습니다. 함수 컴포넌트에서 동일한 state로 `set` 함수를 호출하면 기본적으로 `memo` 없이도 다시 렌더링 되지 않습니다.

< [이전](#)
[isValidElement](#)

 Meta Open Source

Copyright © Meta Platforms, Inc

uwu?

React 학습하기

[빠르게 시작하기](#)

[설치하기](#)

[UI 표현하기](#)

[상호작용성 더하기](#)

[State 관리하기](#)

[탈출구](#)

API 참고서

[React APIs](#)

[React DOM APIs](#)

커뮤니티

[행동 강령](#)

[팀 소개](#)

[문서 기여자](#)

[감사의 말](#)

더 보기

[블로그](#)

[React Native](#)

[개인 정보 보호](#)

[약관](#)

