



# isValidElement

isValidElement 는 값이 React 엘리먼트인지 확인합니다.

```
const isElement = isValidElement(value)
```

- [레퍼런스](#)
  - [isValidElement\(value\)](#)
- [사용법](#)
  - [어떤 것이 React 엘리먼트인지 확인하기](#)

## 레퍼런스

### isValidElement(value)

isValidElement(value) 를 호출하여 value 가 React 엘리먼트인지 확인합니다.

```
import { isValidElement, createElement } from 'react';

// ✅ React 엘리먼트
console.log(isValidElement(<p />)); // true
console.log(isValidElement(createElement('p'))); // true

// ❌ React 엘리먼트가 아님
console.log(isValidElement(25)); // false
console.log(isValidElement('Hello')); // false
console.log(isValidElement({ age: 42 })); // false
```

아래에서 더 많은 예시를 확인하세요

## 매개변수

`value`: 확인하려는 `value` 입니다. 모든 종류의 값이 될 수 있습니다.

## 반환값

`isValidElement` 는 `value` 가 `React` 엘리먼트인 경우 `true` 를 반환합니다. 그렇지 않으면 `false` 를 반환합니다.

## 주의 사항

- `createElement` 가 반환한 **JSX 태그**와 객체는 `React` 엘리먼트로 간주합니다. 예를 들어, 42와 같은 숫자는 유효한 `React` 노드 (컴포넌트에서 반환될 수 있지만)이지만, 유효한 `React` 엘리먼트는 아닙니다. `createPortal` 로 만들어진 배열과 `portal`도 `React` 엘리먼트로 간주하지 않습니다.

## 사용법

### 어떤 것이 `React` 엘리먼트인지 확인하기

어떤 값이 `React` 엘리먼트인지 확인하려면 `isValidElement` 를 호출해 보세요.

`React` 엘리먼트는 다음과 같습니다.

- **JSX tag**를 작성하여 생성된 값
- `createElement` 를 호출하여 생성된 값

`React` 엘리먼트의 경우 `isValidElement` 는 `true` 를 반환합니다.

```
import { isValidElement, createElement } from 'react';

// ✅ JSX 태그는 React 엘리먼트입니다.
console.log(isValidElement(<p />)); // true
console.log(isValidElement(<MyComponent />)); // true

// ✅ createElement가 반환하는 값은 React 엘리먼트입니다.
console.log(isValidElement(createElement('p'))); // true
```

```
console.log(isValidElement(createElement(MyComponent))); // true
```

문자열, 숫자, 임의의 객체 및 배열과 같은 값들은 React 엘리먼트가 아닙니다.

이 경우 `isValidElement` 는 `false` 를 반환합니다.

```
// ❌ 이것들은 React 엘리먼트가 *아닙니다*.
console.log(isValidElement(null)); // false
console.log(isValidElement(25)); // false
console.log(isValidElement('Hello')); // false
console.log(isValidElement({ age: 42 })); // false
console.log(isValidElement([<div />, <div />])); // false
console.log(isValidElement(MyComponent)); // false
```

`isValidElement` 가 필요한 경우는 매우 드뭅니다. 주로 “엘리먼트만” 허용하는 다른 API를 호출할 때와 (`cloneElement` 가 하는 것처럼) 인수가 React 엘리먼트가 아닌 경우 오류를 피하고 싶을 때 유용합니다.

`isValidElement` 확인을 추가 해야 하는 구체적인 이유가 없는 한 이 확인은 필요하지 않을 수 있습니다.

[📖 자세히 살펴보기](#)

## React 엘리먼트 vs React 노드

[자세히 보기](#)

[< 이전](#)  
[forwardRef](#)

[다음 >](#)  
[PureComponent](#)

---

 Meta Open Source

Copyright © Meta Platforms, Inc

uwu?

## React 학습하기

[빠르게 시작하기](#)

[설치하기](#)

[UI 표현하기](#)

[상호작용성 더하기](#)

[State 관리하기](#)

[탈출구](#)

## 커뮤니티

[행동 강령](#)

[팀 소개](#)

[문서 기여자](#)

[감사의 말](#)

## API 참고서

[React APIs](#)

[React DOM APIs](#)

## 더 보기

[블로그](#)

[React Native](#)

[개인 정보 보호](#)

[약관](#)

