



useImperativeHandle

useImperativeHandle 은 [Ref](#)로 노출되는 핸들을 사용자가 직접 정의할 수 있게 해주는 React Hook입니다.

```
useImperativeHandle(ref, createHandle, dependencies?)
```

- 레퍼런스

- `useImperativeHandle(ref, createHandle, dependencies?)`

- 사용법

- 부모 컴포넌트에 커스텀 Ref 핸들 노출
- 사용자 정의 Imperative 메서드 노출

레퍼런스

useImperativeHandle(ref, createHandle, dependencies?)

컴포넌트의 최상위 레벨에서 useImperativeHandle 을 호출하여 노출할 Ref 핸들을 사용자가 직접 정의할 수 있습니다.

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

function MyInput({ ref }) {
  useImperativeHandle(ref, () => {
    return {
      // ... 메서드를 여기에 입력하세요 ...
    };
  });
}
```

```
} , [] );  
// ...
```

아래에서 더 많은 예시를 확인하세요.

매개변수

- `ref: MyInput` 컴포넌트의 `Prop`으로 받은 `ref` 입니다.
- `createHandle`: 인수가 없고 노출하려는 `Ref` 핸들을 반환하는 함수입니다. 해당 `Ref` 핸들은 어떠한 유형이든 될 수 있습니다. 일반적으로 노출하려는 메서드가 있는 객체를 반환합니다.
- **(선택적)** `dependencies`: `createHandle` 코드 내에서 참조하는 모든 반응형 값을 나열한 목록입니다. 반응형 값은 `Props`, `State` 및 컴포넌트 내에서 직접 선언한 모든 변수와 함수를 포함합니다. [React에 대한 린터Linter를 구성한 경우](#), 모든 반응형 값이 올바르게 의존성으로 지정되었는지 확인합니다. 의존성 목록은 항상 일정한 수의 항목을 가지고 `[dep1, dep2, dep3]`와 같이 인라인으로 작성되어야 합니다. `React`는 각 의존성을 `Object.is` 비교를 사용하여 이전 값과 비교합니다. 리렌더링으로 인해 일부 의존성이 변경되거나, 이 인수를 생략한 경우 `createHandle` 함수가 다시 실행되고 새로 생성된 핸들이 `Ref`에 할당됩니다.

📌 중요합니다!

`React 19` 부터 `ref` 를 `Prop`으로 받을 수 있습니다. `React 18` 또는 그 이전 버전에서는 `ref` 를 받기 위해 `forwardRef` 를 사용해야 합니다.

반환값

`useImperativeHandle` 은 `undefined` 를 반환합니다.

사용법

부모 컴포넌트에 커스텀 `Ref` 핸들 노출

부모 엘리먼트에 `DOM` 노드를 노출하려면 해당 노드에 `ref Prop`을 전달해야 합니다.

```
function MyInput({ ref }) {
  return <input ref={ref} />;
};
```

위의 코드에서 `MyInput`에 대한 `Ref`는 `<input>` DOM 노드를 받게 됩니다. 그러나 대신 사용자 지정 값을 노출할 수 있습니다. 노출된 핸들을 사용자 정의하려면 컴포넌트의 최상위 레벨에서 `useImperativeHandle` 을 호출하세요.

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

function MyInput({ ref }) {
  useImperativeHandle(ref, () => {
    return {
      // ... 메서드를 여기에 입력하세요 ...
    };
  }, []);

  return <input />;
};
```

위의 코드에서 `<input>`에 대한 `ref`는 더이상 전달되지 않습니다.

예를 들어 전체 `<input>` DOM 노드를 노출하지 않고 `focus`와 `scrollIntoView`의 두 메서드만을 노출하고 싶다고 가정해 봅시다. 그러기 위해서는 실제 브라우저 DOM을 별도의 `Ref`에 유지해야 합니다. 그리고 `useImperativeHandle`을 사용하여 부모 컴포넌트에서 호출할 메서드만 있는 핸들을 노출합니다.

```
import { useRef, useImperativeHandle } from 'react';

function MyInput({ ref }) {
  const inputRef = useRef(null);

  useImperativeHandle(ref, () => {
    return {
      focus() {
        inputRef.current.focus();
      },
    };
  });
}
```

```

        scrollIntoView() {
            inputRef.current.scrollIntoView();
        },
    };
}, []);

return <input ref={inputRef} />;
};

```

이제 부모 컴포넌트가 MyInput 에 대한 Ref를 가져오면 focus 및 scrollIntoView 메서드를 호출할 수 있습니다. 그러나 기본 <input> DOM 노드의 전체 액세스 권한은 없습니다.

App.js MyInput.js

🔄 새로고침 ✕ Clear 📌 포크

```

import { useRef } from 'react';
import MyInput from './MyInput.js';

export default function Form() {
    const ref = useRef(null);

    function handleClick() {
        ref.current.focus();
        // 이 작업은 DOM 노드가 노출되지 않으므로 작동하지 않습니다.
        // ref.current.style.opacity = 0.5;
    }
}

```

▼ 자세히 보기

사용자 정의 Imperative 메서드 노출

Imperative Handle을 통해 노출하는 메서드는 DOM 메서드와 정확하게 일치할 필요가 없습니다. 예를 들어, 이 `Post` 컴포넌트는 Imperative Handle을 통해 `scrollAndFocusAddComment` 메서드를 표시합니다. 이렇게 하면 부모 `Page` 에서 버튼을 클릭할 때 댓글 목록을 스크롤하고 입력 필드에 초점을 맞출 수 있습니다.

[App.js](#) [Post.js](#) [CommentList.js](#) [AddComment.js](#)

↻ 새로고침 ✕ Clear 📌 포크

```
import { useRef } from 'react';
import Post from './Post.js';

export default function Page() {
  const postRef = useRef(null);

  function handleClick() {
    postRef.current.scrollAndFocusAddComment();
  }

  return (
    <>
```

▼ 자세히 보기

⚠️ 주의하세요!

Ref를 과도하게 사용하지 마세요. Ref는 Props로 표현할 수 없는 필수적인 행동에만 사용해야 합니다. 예를 들어 특정 노드로 스크롤 하기, 노드에 초점 맞추기, 애니메이션 실행 하기, 텍스트 선택하기 등이 있습니다.

Prop으로 표현할 수 있는 것에 Ref를 사용하지 마세요. 예를 들어 `Modal` 컴포넌트에서 `{ open, close }`와 같은 Imperative Handle을 노출하는 대신 `<Modal isOpen={isOpen} />`과 같은 `isOpen` Prop을 사용하는 것이 더 좋습니다. [Effect](#)를 사용하면 Prop을 통해 명령형 동작(Imperative Behavior)을 노출할 수 있습니다.

< [이전](#)
[useId](#)

[useInsertionEffect](#) > [다음](#)

 Meta Open Source

Copyright © Meta Platforms, Inc

uwu?

React 학습하기

[빠르게 시작하기](#)

[설치하기](#)

[UI 표현하기](#)

[상호작용성 더하기](#)

[State 관리하기](#)

API 참고서

[React APIs](#)

[React DOM APIs](#)

탈출구

커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

더 보기

블로그

React Native

개인 정보 보호

약관

