# createElement

`createElement`를 사용하면 React 엘리먼트를 생성할 수 있습니다. JSX를 작성하는 대신 사용할 수 있습니다.

```
const element = createElement(type, props, ...children)
```



## 레퍼런스

### `createElement(type, props, ...children)`

`type`, `prop`, `children`를 인수로 제공하고 `createElement`을 호출하여 React 엘리먼트를 생성합니다.

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

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello'
  );
}
```

아래에서 더 많은 예시를 확인하세요.

## 매개변수

- `type`: `type` 인수는 유효한 React 컴포넌트여야 합니다. 예를 들어 태그 이름 문자열 (예: 'div', 'span') 또는 React 컴포넌트(함수, 클래스, `Fragment` 같은 특수 컴포넌트)가 될 수 있습니다.

- `props`: `props` 인수는 객체 또는 `null` 이어야 합니다. `null` 을 전달하면 빈 객체와 동일하게 처리됩니다. React는 전달한 `props` 와 일치하는 프로퍼티를 가진 엘리먼트를 생성합니다. 전달한 `props` 객체의 `ref` 와 `key` 는 특수하기 때문에 생성한 `element` 에서 `element.props.ref` 와 `element.props.key` 는 사용할 수 *없다*는 점에 유의하세요. `element.ref` 또는 `element.key` 로 사용할 수 있습니다.

- **선택사항** `...children`: 0개 이상의 자식 노드. React 엘리먼트, 문자열, 숫자, 포탈, 빈 노드 (`null`, `undefined`, `true`, `false`) 그리고 React 노드 배열을 포함한 모든 React 노드가 될 수 있습니다.

## 반환값

`createElement` 는 아래 프로퍼티를 가지는 React 엘리먼트 객체를 반환합니다.

- `type`: 전달받은 `type`.
- `props`: `ref` 와 `key` 를 제외한 전달받은 `props`.
- `ref`: 전달받은 `ref`. 누락된 경우 `null`.
- `key`: 전달받은 `key` 를 강제 변환한 문자열. 누락된 경우 `null`.

일반적으로 엘리먼트는 컴포넌트에서 반환되거나 다른 엘리먼트의 자식으로 만듭니다. 엘리먼트의 프로퍼티에는 접근할 수 있지만, 엘리먼트 생성 후에는 모든 엘리먼트에 접근할 수 없는 것처럼 대하고 렌더링만 하는 것이 좋습니다.

## 주의 사항

- 반드시 **React 엘리먼트와 그 프로퍼티는 불변하게 취급**해야하며 엘리먼트 생성 후에는 그 내용이 변경되어선 안 됩니다. 개발환경에서 React는 이를 강제하기 위해 반환된 엘리먼트와 그 프로퍼티를 얕게 freeze합니다.

- JSX를 사용한다면 **태그를 대문자로 시작해야만 사용자 컴포넌트를 렌더링할 수 있습니다.** 즉, `<Something />` 은 `createElement(Something)` 과 동일하지만 `<something />` (소문자)은 `createElement('something')` 와 동일합니다. (문자열임을 주의하세요. 내장된 HTML 태그로 취급됩니다.)

- `createElement('h1', {}, child1, child2, child3)` 와 같이 **`children` 이 모두 정적인 경우에만 `createElement` 에 여러 인수로 전달해야 합니다.** `children` 이 동적이라면 전체 배열을 세 번째 인수로 전달해야 합니다. 이렇게 하면 React는 누락된 `키` 에 대한 경고를 표시합니다. 정적 목록인 경우 재정렬하지 않기 때문에 작업이 필요하지 않습니다.

---

# 사용법

## JSX 없이 엘리먼트 생성하기

JSX가 마음에 들지 않거나 프로젝트에서 사용할 수 없는 경우, `createElement` 를 대안으로 사용할 수 있습니다.

JSX 없이 엘리먼트를 생성하려면 `type` , `props` , `children` 와 함께 `createElement` 를 호출합니다

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

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello ',
    createElement('i', null, name),
    '. Welcome!'
  );
}
```

`children` 은 선택 사항이며 필요한 만큼 전달할 수 있습니다. (위 예시에는 3개의 children이 있습니다.) 위 코드는 인사말이 포함된 `<h1>` 를 표시합니다. 비교를 위해 동일한 예시를 JSX로 재작성했습니다.

```
function Greeting({ name }) {
  return (
    <h1 className="greeting">
      Hello <i>{name}</i>. Welcome!
    </h1>
```

```
  );
}
```

자신만의 React 컴포넌트를 렌더링하려면 `'h1'` 같은 문자열 대신 `Greeting` 같은 함수를 `type` 에 전달하세요.

```
export default function App() {
  return createElement(Greeting, { name: 'Taylor' });
}
```

JSX를 사용하면 다음과 같습니다.

```
export default function App() {
  return <Greeting name="Taylor" />;
}
```

`createElement` 를 사용하여 작성한 전체 예시입니다.

App.js

다운로드 새로고침 Clear 포크

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

function Greeting({ name }) {
  return createElement(
    'h1',
    { className: 'greeting' },
    'Hello ',
    createElement('i', null, name),
    '. Welcome!'
  );
}
```

자세히 보기

JSX를 사용하여 작성한 전체 예시입니다.

App.js

다운로드 새로고침 Clear 포크

```
function Greeting({ name }) {
  return (
    <h1 className="greeting">
      Hello <i>{name}</i>. Welcome!
    </h1>
  );
}

export default function App() {
  return <Greeting name="Taylor" />;
}
```

두 코딩 스타일 모두 허용되므로 프로젝트에 맞는 스타일을 사용하면 됩니다. `createElement` 와 비교하여 JSX를 사용할 때의 장점은 어떤 닫는 태그가 어떤 여는 태그에 대응되는지 쉽게 확인할 수 있다는 것입니다.

자세히 살펴보기

### React 엘리먼트란 정확히 무엇인가요?

자세히 보기


이전
Component

다음
createRef

탈출구

## 커뮤니티

행동 강령

팀 소개

문서 기여자

감사의 말

## 더 보기

블로그

React Native

개인 정보 보호

약관