Events

React's event system -- onClick, defining event handlers, onChange, onSubmit, and the event object -- with simple examples.

Beginner 9 min

Last updated: August 18, 2026
Read online: https://www.learnforgex.com/en/topics/events

Events

In Components & Props, we learned how to combine components. Now we start making the UI actually interactive -- responding to a user's clicks, typing, and form submissions. This lesson covers React's event system.

What Is an Event?

An event is something the user does on the page: clicking a button, typing into an input, submitting a form. React gives you ready-made attributes to catch these actions and react to them -- `onClick`, `onChange`, `onSubmit`, and more.

Catching Clicks with onClick

If you give the `onClick` attribute a function, that function runs every time the user clicks the button:

```
function OnClickExample() {
  function handleClick() {
    console.log("Button clicked!");
  }

  return <button onClick={handleClick}>Click</button>;
}
```

We wrote `onClick={handleClick}` -- not `handleClick()`. That difference matters, and we'll look at it in the next section.

Defining an Event Handler Function

A function that reacts to an event is called an **event handler**. You can write it as a named function, or directly inline:

```
function EventHandlerFunctionExample() {
  function sayHello() {
    console.log("Hello!");
  }

  return (
    <div>
      {/* CORRECT: we pass the function itself, we don't call it. */}
      <button onClick={sayHello}>Named Function</button>

      {/* You can also write an inline function. */}
      <button onClick={() => console.log("Hi!")}>Inline Function</button>

      {/* WRONG: if you write sayHello(), the function runs immediately
          as soon as the component renders -- it won't wait for a click. */}
      {/* <button onClick={sayHello()}>Wrong Usage</button> */}
    </div>
  );
}
```

The key rule: write `onClick={sayHello}`, **not** `onClick={sayHello()}`. Writing the second one calls the function immediately, as soon as the component renders -- it doesn't wait for a click, because `sayHello()` CALLS the function and hands its result (`undefined`) to `onClick`. `sayHello` (no parentheses) hands over the function itself, and React calls it for you at the right time.

Catching Input Changes with onChange

`onChange` runs every time an input's value changes -- every time the user types a character:

```
function OnChangeExample() {
  function handleChange(event) {
    console.log("Typed value:", event.target.value);
  }

  return <input type="text" onChange={handleChange} placeholder="Type something..." />;
}
```

The event handler automatically receives an **event object** -- you can read what was just typed with `event.target.value`.

Catching Form Submission with onSubmit

`onSubmit` runs when a `<form>` is submitted -- whether the user presses Enter or clicks a submit button:

```
function OnSubmitExample() {
  function handleSubmit(event) {
    // Without preventDefault, the browser tries to submit the form and
    // the page reloads -- we don't want that in React apps.
    event.preventDefault();
    console.log("Form submitted!");
  }

  return (
    <form onSubmit={handleSubmit}>
      <button type="submit">Submit</button>
    </form>
  );
}
```

`event.preventDefault()` is an important line: without it, the browser falls back to its "default" behavior and reloads the page. In React apps we almost never want that -- we want to decide what happens with our own code, without a page reload.

The Event Object

Every event handler function automatically receives an **event object** from React. It carries information about the event -- what kind of event it was, which element it happened on, and more:

```
function EventObjectExample() {
  function handleClick(event) {
    console.log("Event type:", event.type);
    console.log("Target element:", event.target.tagName);
  }

  return <button onClick={handleClick}>See the Event Object (Check the Console)</button>;
}
```

`event.type` gives you the kind of event ("click", "change", "submit", etc.), and `event.target` gives you the DOM element the event happened on.

Summary and Glossary

In React, you react to events with attributes like `onClick`, `onChange`, and `onSubmit`. Give an event handler the function itself, don't call it (`onClick={f}`, not `onClick={f()}`). Every event handler automatically receives an event object; in forms, `event.preventDefault()` stops the browser's default behavior (reloading the page).

Glossary

Event — Something the user does on the page (clicking, typing, submitting a form, etc.).

Event Handler — A function that runs when an event happens.

Event Object — An object automatically passed to an event handler when an event happens, carrying information about that event.

`preventDefault()` — An event object method that stops the browser's default behavior for an event (like reloading the page on form submission).