SyntheticEvent

This reference guide documents the `SyntheticEvent` wrapper that forms part of React's Event System. See the Handling Events guide to learn more.

Overview

Your event handlers will be passed instances of `SyntheticEvent`, a cross-browser wrapper around the browser's native event. It has the same interface as the browser's native event, including `stopPropagation()` and `preventDefault()`, except the events work identically across all browsers.

If you find that you need the underlying browser event for some reason, simply use the `nativeEvent` attribute to get it. The synthetic events are different from, and do not map directly to, the browser's native events. For example in `onMouseLeave` `event.nativeEvent` will point to a `mouseout` event. The specific mapping is not part of the public API and may change at any time. Every `SyntheticEvent` object has the following attributes:

```
boolean bubbles
boolean cancelable
DOMEventTarget currentTarget
boolean defaultPrevented
number eventPhase
boolean isTrusted
DOMEvent nativeEvent
void preventDefault()
boolean isDefaultPrevented()
void stopPropagation()
boolean isPropagationStopped()
void persist()
DOMEventTarget target
number timeStamp
string type
```

> **Note:**
> As of v17, `e.persist()` doesn't do anything because the `SyntheticEvent` is no longer pooled.

> **Note:**
> As of v0.14, returning `false` from an event handler will no longer stop event propagation. Instead, `e.stopPropagation()` or `e.preventDefault()` should be triggered manually, as appropriate.

Supported Events

React normalizes events so that they have consistent properties across different browsers.

The event handlers below are triggered by an event in the bubbling phase. To register an event handler for the capture phase, append `Capture` to the event name; for example, instead of using `onClick`, you would use `onClickCapture` to handle the click event in the capture phase.

- Clipboard Events
- Composition Events
- Keyboard Events
- Focus Events
- Form Events
- Generic Events
- Mouse Events
- Pointer Events
- Selection Events

- Touch Events
- UI Events
- Wheel Events
- Media Events
- Image Events
- Animation Events
- Transition Events
- Other Events

Reference

Clipboard Events

Event names:

```
onCopy onCut onPaste
```

Properties:

```
DOMDataTransfer clipboardData
```

Composition Events

Event names:

```
onCompositionEnd onCompositionStart onCompositionUpdate
```

Properties:

```
string data
```

Keyboard Events

Event names:

```
onKeyDown onKeyPress onKeyUp
```

Properties:

```
boolean altKey
number charCode
boolean ctrlKey
boolean getModifierState(key)
string key
number keyCode
string locale
number location
boolean metaKey
boolean repeat
boolean shiftKey
number which
```

The `key` property can take any of the values documented in the DOM Level 3 Events spec.

Focus Events

Event names:

```
onFocus onBlur
```

These focus events work on all elements in the React DOM, not just form elements.

Properties:

```
DOMEventTarget relatedTarget
```

onFocus

The `onFocus` event is called when the element (or some element inside of it) receives focus. For example, it's called when the user clicks on a text input.

```
function Example() {
  return (
    <input
      onFocus={(e) => {
        console.log('Focused on input');
      }}
      placeholder="onFocus is triggered when you click this input."
    />
  )
}
```

onBlur

The `onBlur` event handler is called when focus has left the element (or left some element inside of it). For example, it's called when the user clicks outside of a focused text input.

```
function Example() {
  return (
    <input
      onBlur={(e) => {
        console.log('Triggered because this input lost focus');
      }}
      placeholder="onBlur is triggered when you click this input and then you click
outside of it."
    />
  )
}
```

Detecting Focus Entering and Leaving

You can use the `currentTarget` and `relatedTarget` to differentiate if the focusing or blurring events originated from *outside* of the parent element. Here is a demo you can copy and paste that shows how to detect focusing a child, focusing the element itself, and focus entering or leaving the whole subtree.

```
function Example() {
  return (
    <div
      tabIndex={1}
      onFocus={(e) => {
        if (e.currentTarget === e.target) {
          console.log('focused self');
        } else {
          console.log('focused child', e.target);
        }
        if (!e.currentTarget.contains(e.relatedTarget)) {
          // Not triggered when swapping focus between children
          console.log('focus entered self');
        }
      }}
      onBlur={(e) => {
        if (e.currentTarget === e.target) {
          console.log('unfocused self');
        } else {
          console.log('unfocused child', e.target);
        }
        if (!e.currentTarget.contains(e.relatedTarget)) {
          // Not triggered when swapping focus between children
          console.log('focus left self');
        }
      }}
    >
      <input id="1" />
      <input id="2" />
    </div>
  );
}
```

Form Events

Event names:

```
onChange onInput onInvalid onReset onSubmit
```

For more information about the onChange event, see Forms.

Generic Events

Event names:

```
onError onLoad
```

Mouse Events

Event names:

```
onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit
onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave
onMouseMove onMouseOut onMouseOver onMouseUp
```

The `onMouseEnter` and `onMouseLeave` events propagate from the element being left to the one being entered instead of ordinary bubbling and do not have a capture phase.

Properties:

```
boolean altKey
number button
number button
number buttons

number clientX
number clientY
boolean ctrlKey
boolean getModifierState(key)
boolean metaKey
number pageX
number pageY
DOMEventTarget relatedTarget
number screenX
number screenY
boolean shiftKey
```

Pointer Events

Event names:

```
onPointerDown onPointerMove onPointerUp onPointerCancel onGotPointerCapture
onLostPointerCapture onPointerEnter onPointerLeave onPointerOver onPointerOut
```

The `onPointerEnter` and `onPointerLeave` events propagate from the element being left to the one being entered instead of ordinary bubbling and do not have a capture phase.

Properties:

As defined in the W3 spec, pointer events extend Mouse Events with the following properties:

```
number pointerId
number width
number height
number pressure
number tangentialPressure
number tiltX
number tiltY
number twist
string pointerType
boolean isPrimary
```

A note on cross-browser support:

Pointer events are not yet supported in every browser (at the time of writing this article, supported browsers include: Chrome, Firefox, Edge, and Internet Explorer). React deliberately does not polyfill support for other browsers because a standard-conform polyfill would significantly increase the bundle size of `react-dom`.

If your application requires pointer events, we recommend adding a third party pointer event polyfill.

Selection Events

Event names:

```
onSelect
```

Touch Events

Event names:

```
onTouchCancel onTouchEnd onTouchMove onTouchStart
```

Properties:

```
boolean altKey
DOMTouchList changedTouches
boolean ctrlKey
boolean getModifierState(key)
boolean metaKey
boolean shiftKey
DOMTouchList targetTouches
DOMTouchList touches
```

UI Events

Event names:

```
onScroll
```

> **Note**
> Starting with React 17, the `onScroll` event **does not bubble** in React. This matches the browser behavior and prevents the confusion when a nested scrollable element fires events on a distant parent.

Properties:

```
number detail
DOMAbstractView view
```

Wheel Events

Event names:

```
onWheel
```

Properties:

```
number deltaMode
number deltaX
number deltaX
```

```
number deltaY
number deltaZ
```

Media Events

Event names:

```
onAbort onCanPlay onCanPlayThrough onDurationChange onEmptied onEncrypted
onEnded onError onLoadedData onLoadedMetadata onLoadStart onPause onPlay
onPlaying onProgress onRateChange onSeeked onSeeking onStalled onSuspend
onTimeUpdate onVolumeChange onWaiting
```

Image Events

Event names:

```
onLoad onError
```

Animation Events

Event names:

```
onAnimationStart onAnimationEnd onAnimationIteration
```

Properties:

```
string animationName
string pseudoElement
float elapsedTime
```

Transition Events

Event names:

```
onTransitionEnd
```

Properties:

```
string propertyName
string pseudoElement
float elapsedTime
```

Other Events

Event names:

```
onToggle
```