# 合成イベント (SyntheticEvent)

このリファレンスガイドでは、React のイベントシステムの一部を構成する `SyntheticEvent`（合成イベント）ラッパについて説明します。詳細については、イベント処理ガイドを参照してください。

---

## 概要

イベントハンドラには、`SyntheticEvent` のインスタンスが渡されます。これはブラウザのネイティブイベントに対するクロスブラウザ版のラッパです。`stopPropagation()` と `preventDefault()` を含む、ブラウザのネイティブイベントと同じインターフェイスを持ちつつ、ブラウザ間で同じ挙動をするようになっています。
何らかの理由で実際のブラウザイベントが必要な場合は、単に `nativeEvent` 属性を使用するだけで取得できます。すべての `SyntheticEvent` オブジェクトは以下の属性を持っています。

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

> **補足**
> v0.14 以降、イベントハンドラから `false` を返してもイベントの伝播は止まりません。代わりに、`e.stopPropagation()` または `e.preventDefault()` を手動で呼び出す必要があります。

### イベントのプール

`SyntheticEvent` はプールされます。つまり、`SyntheticEvent` オブジェクトは再利用され、すべてのプロパティはイベントコールバックが呼び出された後に `null` で初期化されます。 これはパフォーマンス上の理由からです。 そのため、非同期処理の中でイベントオブジェクトにアクセスすることはできません。

```
function onClick(event) {
  console.log(event); // => null で初期化されるオブジェクト
  console.log(event.type); // => "click"
  const eventType = event.type; // => "click"

  setTimeout(function() {
    console.log(event.type); // => null
    console.log(eventType); // => "click"
  }, 0);

  // これは動作しません。this.state.clickEvent は null 値のみを持つオブジェクトとなります。
  this.setState({clickEvent: event});

  // イベントプロパティをエクスポートすることは可能です。
  this.setState({eventType: event.type});
}
```

> **補足**
> 非同期処理の中でイベントのプロパティにアクセスしたい場合は、`event.persist()` をイベント内で呼び出す必要があります。これにより、合成イベントがイベントプールの対象から除外され、イベントへの参照をコードで保持できるようになります。

---

## サポートするイベント

React は異なるブラウザ間でも一貫したプロパティを持つようにイベントを正規化します。
以下のイベントハンドラはイベント伝搬のバブリングフェーズで呼び出されます。キャプチャフェーズのイベントハンドラを登録するには、イベント名に `Capture` を追加します。たとえば、キャプチャフェーズでクリックイベントを処理するには `onClick` の代わりに `onClickCapture` を使用します。

- クリップボードイベント

- コンポジションイベント
- キーボードイベント
- フォーカスイベント
- フォームイベント
- マウスイベント
- ポインタイベント
- 選択イベント
- タッチイベント
- UI イベント
- ホイールイベント
- メディアイベント
- 画像イベント
- アニメーションイベント
- 遷移イベント
- その他のイベント

---

## リファレンス

**クリップボードイベント**
イベント名：

```
onCopy onCut onPaste
```

プロパティ：

```
DOMDataTransfer clipboardData
```

---

**コンポジションイベント**
イベント名：

```
onCompositionEnd onCompositionStart onCompositionUpdate
```

プロパティ：

```
string data
```

---

**キーボードイベント**
イベント名：

```
onKeyDown onKeyPress onKeyUp
```

プロパティ：

```
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
```

`key` プロパティは DOM Level 3 Events spec に記載されている任意の値を取ることができます。

**フォーカスイベント**
イベント名：

```
onFocus onBlur
```

これらのフォーカスイベントは、フォーム要素だけでなくすべての React DOM 要素で動作します。
プロパティ：

```
DOMEventTarget relatedTarget
```

---

**フォームイベント**
イベント名：

```
onChange onInput onInvalid onSubmit
```

onChange イベントの詳細については、Forms を参照してください。

---

**マウスイベント**
イベント名：

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

`onMouseEnter` と `onMouseLeave` イベントは通常のバブリングとは異なり、（ポインタが）出て行った要素から入ってきた要素に伝播し、キャプチャフェーズを持ちません。
プロパティ：

```
boolean altKey
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
```

---

**ポインタイベント**
イベント名：

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

`onPointerEnter` と `onPointerLeave` イベントは通常のバブリングとは異なり、（ポインタが）出て行った要素から入ってきた要素に伝播し、キャプチャフェーズを持ちません。
プロパティ：
W3 spec に定義されている通り、ポインタイベントは下記のプロパティを持つマウスイベントの拡張です。

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

クロスブラウザサポートについての補足：
すべてのブラウザでポインタイベントがサポートされているわけではありません（この記事の執筆時点でサポートされているブラウザは、Chrome、Firefox、Edge、および Internet Explorer です）。標準に準拠したポリフィルは `react-dom` のバンドルサイズを大幅に増加させるため、React は意図的にその他ブラウザのためのポリフィルを提供しません。
アプリケーションでポインタイベントが必要な場合は、サードパーティのポインタイベントポリフィルを追加することをお勧めします。

---

**選択イベント**
イベント名：

```
onSelect
```

---

**タッチイベント**
イベント名：

```
onTouchCancel onTouchEnd onTouchMove onTouchStart
```

プロパティ：

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

---

**UI イベント**
イベント名：

```
onScroll
```

プロパティ：

```
number detail
DOMAbstractView view
```

---

**ホイールイベント**
イベント名：

```
onWheel
```

プロパティ：

```
number deltaMode
number deltaX
number deltaY
number deltaZ
```

---

**メディアイベント**
イベント名：

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

---

**画像イベント**
イベント名：

```
onLoad onError
```

---

**アニメーションイベント**
イベント名：

```
onAnimationStart onAnimationEnd onAnimationIteration
```

プロパティ：

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

---

**遷移イベント**
イベント名：

```
onTransitionEnd
```

プロパティ：

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

---

**その他のイベント**
イベント名：

```
onToggle
```

このページを編集する