DOM 要素

React はパフォーマンスとブラウザ間での互換性のために、ブラウザから独立した DOM システムを実装しています。このことを機に、ブラウザの DOM 実装にあるいくつかの粗削りな部分が取り払われました。

React では、DOM のプロパティと属性（イベントハンドラを含む）全てがキャメルケースで名前付けされる必要があります。例えば、HTML 属性 `tabindex` に React で対応する属性は `tabIndex` です。例外は `aria-*` と `data-*` 属性であり、これらは全て小文字に揃える必要があります。例えば、`aria-label` は `aria-label` のままにできます。

属性についての差異

React と HTML で挙動が異なる属性がいくつか存在します。

checked

`checked` 属性はインプットタイプが `checkbox` または `radio` の `<input>` コンポーネントでサポートされています。コンポーネントがチェックされた状態かどうかの設定に、この属性を使うことができます。これは制御されたコンポーネント (controlled component) を構築する際に役立ちます。`defaultChecked` は非制御コンポーネント (uncontrolled component) において同様の働きをする属性で、そのコンポーネントが最初にマウントされた時に、チェックされた状態かどうかを設定します。

className

CSS クラスを指定するには、`className` 属性を使用してください。このことは `<div>`、`<a>` など全ての標準 DOM 要素と SVG 要素に当てはまります。

React を（一般的ではありませんが）Web Components とともに使用する場合は、代わりに class 属性を使用してください。

dangerouslySetInnerHTML

`dangerouslySetInnerHTML` は、ブラウザ DOM における `innerHTML` の React での代替です。一般に、コードから HTML を設定することは、誤ってあなたのユーザをクロスサイトスクリプティング (XSS) 攻撃に晒してしまいやすいため、危険です。そのため、React では直接 HTML を設定することはできますが、それは危険であることを自覚するために `dangerouslySetInnerHTML` と入力し `__html` というキーを持つオブジェクトを渡す必要があります。例えば：

```
function createMarkup() {
  return {__html: 'First &middot; Second'};
}

function MyComponent() {
  return <div dangerouslySetInnerHTML={createMarkup()} />;
}
```

htmlFor

`for` は JavaScript での予約語であるため、React 要素では代わりに `htmlFor` を使用します。

onChange

`onChange` イベントはあなたが期待しているような挙動をします。フォームフィールドに変更があるたび、このイベントが発生します。React の `onChange` という名前は既存のブラウザの挙動に対しては適切な名前では無く、React はリアルタイムでのユーザ入力を扱うためにこのイベントに依存しているため、React では意図的に既存のブラウザの挙動は使用していません。

selected

`selected` 属性が `<option>` コンポーネントでサポートされています。この属性でコンポーネントが選択されているかを設定することができます。制御されたコンポーネントを構築する際に役立ちます。

style

> **補足**
>
> このドキュメンテーションにあるいくつかの例では `style` を便宜上使用していますが、**`style` 属性を要素のスタイリングの主要な手段として使うことは一般的に推奨されません。**多くの場合、`className` を使って外部の CSS スタイルシートに定義された CSS クラスを参照するべきです。React アプリケーションの中では、`style` は動的に計算されたスタイルをレンダー中に追加するために最もよく使われます。FAQ: Styling and CSS も参照してください。

`style` 属性は CSS 文字列ではなく、キャメルケースのプロパティを持った JavaScript オブジェクトを受け取ります。これは JavaScript での DOM の `style` プロパティとの一貫性があり、より効率的で、XSS 攻撃の対象となるセキュリティホールを防ぎます。例えば：

```
const divStyle = {
  color: 'blue',
```

```
  backgroundImage: 'url(' + imgUrl + ')',
};

function HelloWorldComponent() {
  return <div style={divStyle}>Hello World!</div>;
}
```

ベンダープレフィックスの自動追加は行われないことに注意してください。古いブラウザをサポートするには、対応するスタイルのプロパティを与える必要があります：

```
const divStyle = {
  WebkitTransition: 'all', // ここでは大文字の 'W' です
  msTransition: 'all' // 'ms' は小文字である必要がある唯一のベンダープレフィックスです
};

function ComponentWithTransition() {
  return <div style={divStyle}>This should work cross-browser</div>;
}
```

JavaScript から DOM ノードのプロパティにアクセスする場合（例えば `node.style.backgroundImage`）と一致させるために、スタイルのキー名はキャメルケースです。`ms` 以外のベンダープレフィックスは先頭を大文字にしてください。`WebkitTransition` に大文字 “W” があるのはこのためです。
React はインラインスタイルでの特定の数値プロパティに対して自動的に “px” サフィックスを付け加えます。“px” 以外の単位を使用したい場合は、その単位を付け加えた文字列で値を指定してください。例えば：

```
// 最終的なスタイルは '10px'
<div style={{ height: 10 }}>
  Hello World!
</div>

// 最終的なスタイルは '10%'
<div style={{ height: '10%' }}>
  Hello World!
</div>
```

全てのスタイルプロパティがピクセル指定に変換されるわけではありません。特定のプロパティ（例えば `zoom`、`order`、`flex`）は単位が無いままとなります。単位の無いプロパティの完全なリストはこちらで確認できます。

suppressContentEditableWarning

子要素を持つ要素に `contentEditable` 属性が付与されている場合、それは動作しないため通常は警告が出力されます。この属性は、その警告が出力されないようにします。`contentEditable` を自身で管理している Draft.js のようなライブラリを開発するときでもない限り、この属性は使用しないでください。

suppressHydrationWarning

サーバサイドの React レンダリングを使用している場合、サーバとクライアントが違う内容をレンダーする時に通常は警告が出力されます。しかし、まれに両者の内容が完全に一致することの保証が非常に困難あるいは不可能な場合があります。例えば、サーバとクライアントでは、タイムスタンプは異なることが予想されます。
`suppressHydrationWarning` を `true` に設定した場合、その要素の属性と内容の、サーバとクライアントでの差異について React は警告しません。この機能は単一レベルの深さでのみ動作し、避難ハッチとして使われることが想定されています。そのため、むやみに使用しないでください。この “hydration” 機能の詳細については `ReactDOM.hydrate()` のドキュメンテーションで読むことができます。

value

`value` 属性は `<input>` コンポーネントと `<textarea>` コンポーネントでサポートされています。コンポーネントの値を設定することに使用できます。これは制御されたコンポーネントを構築する際に役立ちます。`defaultValue` は非制御コンポーネントにおいて同様の働きをする属性で、コンポーネントが最初にマウントされた時の値を設定します。

サポートされている全ての HTML 属性

React 16 では、標準あるいは独自の DOM 属性全てが完全にサポートされます。
React は DOM に対して JavaScript 中心に設計された API を常に提供してきました。React コンポーネントは、独自および DOM に関連した props を頻繁に受け取るため、React は DOM API と同様にキャメルケース (`camelCase`) の命名規則を属性の名前付けに使用します。

```
<div tabIndex="-1" />      // DOM API の node.tabIndex と同様に
<div className="Button" /> // DOM API の node.className と同様に
<input readOnly={true} />  // DOM API の node.readOnly と同様に
```

このような props は、これまでドキュメントで述べられてきた特殊な例外を除き、対応する HTML 属性と同様に機能します。
React でサポートされている DOM 属性には、以下が含まれます：

```
accept acceptCharset accessKey action allowFullScreen alt async autoComplete
autoFocus autoPlay capture cellPadding cellSpacing challenge charSet checked
cite classID className colSpan cols content contentEditable contextMenu controls
controlsList coords crossOrigin data dateTime default defer dir disabled
download draggable encType form formAction formEncType formMethod formNoValidate
formTarget frameBorder headers height hidden high href hrefLang htmlFor
httpEquiv icon id inputMode integrity is keyParams keyType kind label lang list
```

```
loop low manifest marginHeight marginWidth max maxLength media mediaGroup method
min minLength multiple muted name noValidate nonce open optimum pattern
placeholder poster preload profile radioGroup readOnly rel required reversed
role rowSpan rows sandbox scope scoped scrolling seamless selected shape size
sizes span spellCheck src srcDoc srcLang srcSet start step style summary
tabIndex target title type useMap value width wmode wrap
```

同様に、全ての SVG 属性を完全にサポートしています：

```
accentHeight accumulate additive alignmentBaseline allowReorder alphabetic
amplitude arabicForm ascent attributeName attributeType autoReverse azimuth
baseFrequency baseProfile baselineShift bbox begin bias by calcMode capHeight
clip clipPath clipPathUnits clipRule colorInterpolation
colorInterpolationFilters colorProfile colorRendering contentScriptType
contentStyleType cursor cx cy d decelerate descent diffuseConstant direction
display divisor dominantBaseline dur dx dy edgeMode elevation enableBackground
end exponent externalResourcesRequired fill fillOpacity fillRule filter
filterRes filterUnits floodColor floodOpacity focusable fontFamily fontSize
fontSizeAdjust fontStretch fontStyle fontVariant fontWeight format from fx fy
g1 g2 glyphName glyphOrientationHorizontal glyphOrientationVertical glyphRef
gradientTransform gradientUnits hanging horizAdvX horizOriginX ideographic
imageRendering in in2 intercept k k1 k2 k3 k4 kernelMatrix kernelUnitLength
kerning keyPoints keySplines keyTimes lengthAdjust letterSpacing lightingColor
limitingConeAngle local markerEnd markerHeight markerMid markerStart
markerUnits markerWidth mask maskContentUnits maskUnits mathematical mode
numOctaves offset opacity operator order orient orientation origin overflow
overlinePosition overlineThickness paintOrder panose1 pathLength
patternContentUnits patternTransform patternUnits pointerEvents points
pointsAtX pointsAtY pointsAtZ preserveAlpha preserveAspectRatio primitiveUnits
r radius refX refY renderingIntent repeatCount repeatDur requiredExtensions
requiredFeatures restart result rotate rx ry scale seed shapeRendering slope
spacing specularConstant specularExponent speed spreadMethod startOffset
stdDeviation stemh stemv stitchTiles stopColor stopOpacity
strikethroughPosition strikethroughThickness string stroke strokeDasharray
strokeDashoffset strokeLinecap strokeLinejoin strokeMiterlimit strokeOpacity
strokeWidth surfaceScale systemLanguage tableValues targetX targetY textAnchor
textDecoration textLength textRendering to transform u1 u2 underlinePosition
underlineThickness unicode unicodeBidi unicodeRange unitsPerEm vAlphabetic
vHanging vIdeographic vMathematical values vectorEffect version vertAdvY
vertOriginX vertOriginY viewBox viewTarget visibility widths wordSpacing
writingMode x x1 x2 xChannelSelector xHeight xlinkActuate xlinkArcrole
xlinkHref xlinkRole xlinkShow xlinkTitle xlinkType xmlns xmlnsXlink xmlBase
xmlLang xmlSpace y y1 y2 yChannelSelector z zoomAndPan
```

独自の属性も、その名前が全て小文字であれば使用できます。

このページを編集する