ReactDOM

`<script>` タグから React をロードすると、以下のトップレベル API をグローバル変数 `ReactDOM` で使用することができます。npm と ES6 を使用している場合は、 `import ReactDOM from 'react-dom'` と記述できます。npm と ES5 を使用している場合は、 `var ReactDOM = require('react-dom')` と記述できます。

概要

`react-dom` パッケージには、DOM 固有のメソッドが用意されており、アプリケーションのトップレベルで使用したり、必要に応じて React モデルから外れるための避難ハッチとして使用できます。ほとんどのコンポーネントでは、このモジュールを使用する必要はないはずです。

- `render()`
- `hydrate()`
- `unmountComponentAtNode()`
- `findDOMNode()`
- `createPortal()`

ブラウザサポート

React は Internet Explorer 9 以降を含む全ての一般的なブラウザをサポートしていますが、IE 9 や IE 10 といった古いブラウザではいくつかのポリフィルが必要です。

> **補足**
> React は ES5 をサポートしていない古いブラウザをサポートしていませんが、ページ内に es5-shim や es5-sham のようなポリフィルが含まれている場合、古いブラウザでもアプリケーションが動作することがあります。この手段を選択するかどうかは自己責任で判断してください。

リファレンス

`render()`

```
ReactDOM.render(element, container[, callback])
```

渡された `container` の DOM に React 要素をレンダーし、コンポーネントへの参照（ステートレスコンポーネントの場合は `null`）を返します。
React 要素がすでに `container` にレンダーされている場合は更新を行い、最新の React 要素を反映するために必要な DOM のみを変更します。
オプションのコールバックが渡されている場合は、コンポーネントがレンダーまたは更新された後に実行されます。

> **補足:**
> `ReactDOM.render()` は与えられたコンテナの内容を制御します。コンテナ内部のあらゆる既存の DOM 要素は、最初に呼び出された時に置き換えられます。後続する呼び出しでは効率的な更新のために React の DOM 差分アルゴリズムを使用します。
> `ReactDOM.render()` はコンテナノードを変更しません（コンテナの子要素のみ変更します）。既存の子要素を上書きせずにコンポーネントを既存の DOM ノードに挿入することが可能な場合があります。
> `ReactDOM.render()` は現時点ではルートの `ReactComponent` インスタンスへの参照を返します。 しかし、この戻り値を使用する方法は古く、将来のバージョンの React では一部のケースで非同期にコンポーネントをレンダーするようになる可能性があるため、使用は避けるべきです。 ルートの `ReactComponent` インスタンスへの参照が必要な場合は、ルート要素にコールバックの ref を追加することを推奨します。
> サーバで描画されたコンテナをクライアントで再利用するために `ReactDOM.render()` を使用することは非推奨となり、React 17 では削除されます。代わりに `hydrate()` を使用してください。

`hydrate()`

```
ReactDOM.hydrate(element, container[, callback])
```

`render()` と同様ですが、`ReactDOMServer` により HTML コンテンツが描画されたコンテナをクライアントで再利用するために使用されます。React は既存のマークアップにイベントリスナをアタッチしようとします。
React はレンダーされる内容が、サーバ・クライアント間で同一であることを期待します。React はテキストコンテンツの差異を修復することは可能ですが、その不一致はバグとして扱い、修正すべきです。開発用モードでは、React は両者のレンダーの不一致について警告します。不一致がある場合に属性の差異が修復されるという保証はありません。これはパフォーマンス上の理由から重要です。なぜ

なら、ほとんどのアプリケーションにおいて不一致が発生するということは稀であり、全てのマークアップを検証することは許容不可能なほど高コストになるためです。
単一要素の属性やテキストコンテンツがサーバ・クライアント間においてやむを得ず異なってしまう場合（例えばタイムスタンプなど）、要素に `suppressHydrationWarning={true}` を追加することで警告の発生を停止させることが可能です。それは 1 階層下の要素までで機能するものであり、また避難ハッチとして使われるものです。そのため、多用しないでください。テキストコンテンツでない限り、React は修復を試行しようとはしないため、将来の更新まで不整合が残る可能性があります。
サーバとクライアントで異なるものをレンダーしたい場合は、2 パスレンダーを使用できます。クライアント側で異なるものをレンダーするコンポーネントでは、`this.state.isClient` のような state 変数を読み込み、`componentDidMount()` で `true` を設定することができます。こうすると、最初のレンダーパスではサーバ側と同一の内容を描画して不一致を回避しますが、追加のパスが初回レンダーの直後に同期的に発生します。このアプローチでは 2 回レンダーが発生することによりコンポーネントのパフォーマンスが低下しますので、注意して使用してください。
低速な接続下でのユーザー体験に留意することを忘れないでください。JavaScript のコードは初回の HTML の描画より大幅に遅れてロードされる可能性があるため、クライアントでのみ何か異なるものを描画した場合、その変化は不快感を与える可能性があります。しかしうまく実行されれば、サーバ上でアプリケーションの「外枠」を描画し、クライアント上でのみ追加のウィジェットを表示することは有益になるかもしれません。マークアップの不一致の問題を発生させずにこれを実行する方法については、前の段落の説明をご参照ください。

`unmountComponentAtNode()`

```
ReactDOM.unmountComponentAtNode(container)
```

DOM からマウントされた React コンポーネントを削除し、イベントハンドラや state をクリーンアップします。コンテナにコンポーネントがマウントされていない場合、このメソッドを呼び出しても何も行いません。コンポーネントがアンマウントされた場合は `true` を返し、アンマウントすべきコンポーネントが存在しなかった場合は `false` を返します。

`findDOMNode()`

> **補足:**
> `findDOMNode` は内在する DOM ノードにアクセスするために使用される避難ハッチです。ほとんどのケースにおいて、この避難ハッチの使用はコンポーネントの抽象化に穴を開けてしまうためおすすめしません。`StrictMode` では非推奨になっています。

```
ReactDOM.findDOMNode(component)
```

DOM にこのコンポーネントがマウントされている場合、このメソッドは対応するネイティブブラウザの DOM 要素を返します。このメソッドはフォームフィールドの値や DOM の大きさを計測するのに便利です。**ほとんどのケースにおいて、DOM ノードに ref をアタッチすることで `findDOMNode` の使用を避けることができます。**
コンポーネントが `null` や `false` をレンダーする場合、`findDOMNode` は `null` を返します。コンポーネントが文字列をレンダーする場合、`findDOMNode` はその値を含んだテキスト DOM ノードを返します。React 16 以降、コンポーネントは複数の子要素を含むフラグメントを返すことがありますが、その場合 `findDOMNode` は最初の空でない子要素に対応する DOM ノードを返します。

> **補足:**
> `findDOMNode` はマウントされたコンポーネントに対してのみ機能します（つまり、DOM に配置されたコンポーネント）。まだマウントされていないコンポーネントにおいてこのメソッドを呼ぼうとする場合（まだ作成されていないコンポーネントにおける `render()` の中で `findDOMNode()` を呼びだす場合など）、例外がスローされます。
> `findDOMNode` は関数コンポーネントでは使用できません。

`createPortal()`

```
ReactDOM.createPortal(child, container)
```

ポータルを作成します。ポータルは DOM コンポーネントの階層の外側に存在している DOM ノードに対して子要素をレンダーする方法を提供します。

このページを編集する