要素のレンダー

要素とは React アプリケーションの最小単位の構成ブロックです。
要素は画面上に表示したいものの説明書きです：

```
const element = <h1>Hello, world</h1>;
```

ブラウザの DOM 要素とは異なり、React 要素はプレーンなオブジェクトであり、安価に作成できます。React DOM が React 要素に合致するように DOM を更新する作業を担当します。

> **補足:**
> 要素のことを、より広く知られている概念である “コンポーネント” と混同する人もいるかもしれません。コンポーネントについては次の章で説明します。要素とはコンポーネントを “構成する” ものです。次に進む前にこの章を読んでいくことをお勧めします。

要素を DOM として描画する

HTML ファイルの中に `<div>` 要素があったとしましょう：

```
<div id="root"></div>
```

この中にあるもの全てが React DOM によって管理されることになるので、“ルート” DOM ノードと呼ぶことにしましょう。
React だけで構築されたアプリケーションは、通常ルート DOM ノードをひとつだけ持ちます。既存のアプリに React を統合しようとしている場合は、独立したルート DOM ノードを好きなだけ持つことができます。
React 要素をルート DOM ノードにレンダリングするには、その 2 つを `ReactDOM.render()` に渡します：

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

Try it on CodePen
このコードはページに “Hello, world” を表示します。

レンダリングされた要素の更新

React 要素はイミュータブルです。一度要素を作成すると、その子要素もしくは属性を変更することはできません。要素は映画の中のひとつのフレームのようなものであり、それは特定のある時点の UI を表します。
ここまでで分かる通り、UI を更新する唯一の方法は、新しい要素を作成して `ReactDOM.render()` に渡すことです。
以下の秒刻みで動く時計の例について考えます：

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```

Try it on CodePen
上記のコードでは `setInterval()` のコールバックから `ReactDOM.render()` を毎秒呼び出しています。

> **補足:**
> 実際には大抵の React アプリケーションは `ReactDOM.render()` を一度しか呼び出しません。次の章では上記のようなコードをどのようにステート付きコンポーネントへとカプセル化するのかを学びます。
> トピックはお互いを基礎として構成されているため、読み飛ばさないことをお勧めします。

React は必要な箇所のみを更新する

React DOM は要素とその子要素を以前のものと比較し、DOM を望ましい状態へと変えるのに必要なだけの DOM の更新を行います。

このことは、最後の例をブラウザツールで調査すれば確認できます：

Hello, world!

It is 12:26:46 PM.

毎秒ごとに UI ツリー全体を表す要素を作成しているにも関わらず、内容が変更されたテキストノードのみが React DOM により更新されます。

私達の経験上、時間の経過によりどのように UI が変更されるかを考えるよりも、任意の時点において UI がどのように見えるべきかを考えることで、あらゆる類のバグを排除することができます。

このページを編集する