Rendering Elements

Elements are the smallest building blocks of React apps

An element describes what you want to see on the screen: - Một phần tử mô tả những gì bạn muốn nó thể hiện trên màn hình.

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

Unlike browser DOM elements, React elements are plain objects, and are cheap to create. React DOM takes care of updating the DOM to match the React elements. – Không giống như các phần tử DOM trên trình duyệt, React elements là các object đơn lẻ và không tốn tài nguyên để tạo. React DOM đảm nhận việc cập nhật DOM để khớp với các phần tử React.

Note:

One might confuse elements with a more widely known concept of "components". We will introduce components in the next section. Elements are what components are "made of", and we encourage you to read this section before jumping ahead. – Người ta có thể nhầm lẫn các elements với một khái niệm được biết đến rộng rãi hơn là "components". Chúng tôi sẽ giới thiệu các components trong phần tiếp theo. Elements là những components cấu tạo thành, và chúng tôi khuyến khích bạn đọc phần này trước khi tiếp tục.

Rendering an Element into the DOM

Let's say there is a `<div>` somewhere in your HTML file: - Giả sử có một ở đâu đó trong file HTML của bạn:

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

We call this a "root" DOM node because everything inside it will be managed by React DOM. – Chúng tôi gọi đây là một node DOM "root" vì mọi thứ bên trong nó sẽ được quản lý bởi React DOM.

Applications built with just React usually have a single root DOM node. If you are integrating React into an existing app, you may have as many isolated root DOM nodes as you like. – Các ứng dụng được xây dựng chỉ với React thường có một node DOM gốc duy nhất. Nếu bạn đang tích hợp React vào một ứng dụng hiện có, bạn có thể có nhiều nút DOM gốc riêng biệt tùy thích.

To render a React element into a root DOM node, pass both to `ReactDOM.render()`: - Để render một React element vào trong một nút DOM gốc, truyền cả hai vào ReactDOM.render():

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

Updating the Rendered Element

React elements are immutable. Once you create an element, you can't change its children or attributes. An element is like a single frame in a movie: it represents the UI at a certain point in time. – Các React element là bất biến. Sau khi tạo một element, bạn không thể thay đổi phần tử con hoặc attributes của nó. Một element là một frame duy nhất trong một movie: nó đại diện cho UI tại một thời điểm nhất định.
With our knowledge so far, the only way to update the UI is to create a new element, and pass it to `ReactDOM.render()`. – Với hiểu biết của chúng ta cho đến nay, cách duy nhất để cập nhật UI là tạo một element mới, và pass nó tới ReactDOM.render().

Consider this ticking clock example:

```
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);
```

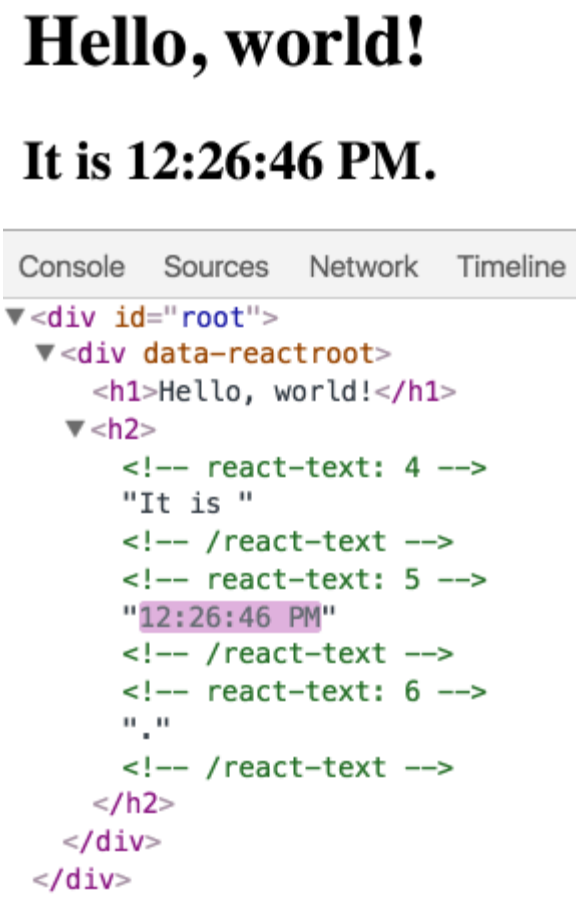
It calls `ReactDOM.render()` every second from a `setInterval()` callback. – Nó gọi ReactDOM.render() mỗi giây từ callback setInterval().

Note:
In practice, most React apps only call `ReactDOM.render()` once. In the next sections we will learn how such code gets encapsulated into stateful components. – Trong thực thế, hầu hết các ứng dụng chỉ gọi ReactDOM.render() một lần. Trong phần tiếp theo chúng ta sẽ tìm hiểu cách code được đóng gói vào trong stateful components.

We recommend that you don't skip topics because they build on each other. – Chúng tôi khuyên rằng bạn đừng bỏ qua các chủ đề vì chúng xây dựng dựa trên nhau.

React Only Updates What's Necessary

React DOM compares the element and its children to the previous one, and only applies the DOM updates necessary to bring the DOM to the desired state. – ReactDOM so sánh element và phần tử con của nó với phần tử trước đó và chỉ áp dụng các bản cập nhật DOM cần thiết để đưa DOM về state mong muốn.
You can verify by inspecting the last example with the browser tools: - Bạn có thể xác minh bằng cách kiểm tra ví dụ cuối cùng với browser tools:



Even though we create an element describing the whole UI tree on every tick, only the text node whose contents have changed gets updated by React DOM. – Mặc dù chúng tôi tạo một phần tử mô tả toàn bộ cây giao diện người dùng trên mỗi lần đánh dấu, nhưng chỉ có nút text có nội dung đã thay đổi được ReactDOM update.

In our experience, thinking about how the UI should look at any given moment, rather than how to change it over time, eliminates a whole class of bugs. – Theo kinh nghiệm của chúng tôi, việc suy nghĩ về giao diện người dùng sẽ trông như thế nào tại bất kỳ

thời điểm nhất định nào, thay vì cách thay đổi nó theo thời gian, sẽ loại bỏ toàn bộ lớp lỗi.