Table of Content

Rendering Elements

Elements are the smallest building blocks of React apps. An element describes what you want to see on the screen:

```
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.

One might confuse elements with a more widely known concept of "components". We will introduce components in the next chapter. Elements are what components are "made of", and we encourage you to read this section before jumping ahead.

Rendering and Element into the DOM

Let's say there is a somewhere in your HTML file:

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

We call this a "root" DOM node because everything inside it will be managed by 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.

To render a React element, first pass the DOM element to ReactDOM.createRoot(), then pass the React element to root.render():

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

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.

With our knowledge so far, the only way to update the UI is to create a new element, and pass it to root.render().

Consider this ticking clock example:

```
const root = ReacDOM.createRoot(
    document.getElementById('root');
);

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

setInterval(tick, 1000);
```

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.

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.

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.

Get Download Link