

**Lesson:**

**createRoot() & render()  
methods of react18**

# Topics Covered:

1. Introducing the new root API
  - a. Legacy Root API
  - b. New Root API
2. Example

## Introducing the new root API

A root in React refers to the top-level data structure that renders a tree. In React 18, we will have two root APIs: **the legacy root API and new root API.**

### Legacy Root API

The legacy root API is the existing API called with the ReactDOM.render method. It will create a root running in legacy mode, which is similar to usage in React version 17. If We are using ReactDOM.render() in your React 18 app, we will definitely see the below issue.

A JSX element is converted into a regular JavaScript object during compilation so that it can be used to construct a genuine DOM element. For instance, the JavaScript generated from the JSX code above might be as follows:

The reason is **ReactDOM.render is no longer supported in React 18.**

Earlier, we used to render the component in below way:

The `render()` method of the `react-dom` package is considered legacy starting `react-dom` version 18.

The method is replaced with the `createRoot()` method that is exported from `react-dom/client`.

The `createRoot()` method takes the root element as a parameter and creates a React root.

### **New Root API**

`createRoot()` is a new method introduced in React 18 that allows you to create a separate root for a React tree outside of the main React DOM tree. It takes a single argument, which is a reference to an existing DOM element. It returns a new root object that has a `render()` method for rendering React components into the specified DOM element.

Call `createRoot` to create a React root for displaying content inside a browser DOM element.

React will create a root for the `domNode`, and take over managing the DOM inside it. After we've created a root, we need to call `root.render` to display a React component inside of it:

An app fully built with React will usually only have one `createRoot` call for its root component. A page that uses “sprinkles” of React for parts of the page may have as many separate roots as needed.

**Let’s take an example to understand better how to use `createRoot()`**

**index.html**

**index.js**

**App.js**

**Let's take another example where we can call `createRoot` methods multiple times to create a root for each top-level piece of UI managed by React. We can display different content in each root by calling `root.render`.**

**index.html**

index.js

Component.js