**Creating and nesting components**

React apps are made out of *components*. A component is a piece of the UI (user interface) that has its own logic and appearance. A component can be as small as a button, or as large as an entire page. React components are JavaScript functions that return markup:

function MyButton() {

return (<button>I'm a button</button>);

}

export default function MyApp() {

return (<div> <h1>Welcome to my app</h1> <MyButton /> </div>

);

}

Notice that <MyButton /> starts with a capital letter. That’s how you know it’s a React component. React component names must always start with a capital letter, while HTML tags must be lowercase.

**Writing markup with JSX**

The markup syntax above is called *JSX*. It is optional, but most React projects use JSX for its convenience. JSX is stricter than HTML. You have to close tags like <br />. Your component also can’t return multiple JSX tags. You have to wrap them into a shared parent, like a <div>...</div> or an empty <>...</> wrapper.

**Adding styles**

In React, you specify a CSS class with className. It works the same way as the HTML class attribute. React does not prescribe how you add CSS files. In the simplest case, you’ll add a <link> tag to your HTML.

**Displaying data**

JSX lets you put markup into JavaScript. Curly braces let you “escape back” into JavaScript so that you can embed some variable from your code and display it to the user.

**Using Hooks**

Functions starting with use are called *Hooks*. useState is a built-in Hook provided by React. You can also write your own Hooks by combining the existing ones.Hooks are more restrictive than other functions. You can only call Hooks *at the top* of your components (or other Hooks). If you want to use useState in a condition or a loop, extract a new component and put it there.