**Learn React: Your First React Component Explained -** [Aphinya Dechalert](https://medium.com/@PurpleGreenLemon?source=post_page-----bd8fecbfaf23--------------------------------) Nov 5, 2021

Part 3 of Learn React from Scratch Series

React apps are made out of **components**. But what is a component?

A component in React is a small and reusable chunk of code that renders some HTML. Here is an example of what a simple React component looks like:

import React from 'react';   
import ReactDOM from 'react-dom'; class AnIttyBittyComponent extends React.Component {   
 render(){   
 return <h1>Hello! I am a squirrel</h1>   
 }   
} ReactDOM.render(   
 <AnIttyBIttyComponent />,   
 document.getElementById('app')   
);

Let’s break the above React component down, line by line.

**Understanding React and ReactDOM**

The first line — import React from 'react'; - brings in the React module from your React library. It also creates an Object called React which you can tap into pre-written methods.

For example, createElement() is a method that belongs in the React object. createElement() is also how React renders things into HTML. When you're using JSX, JSX compiles and transforms your code into a React.createElement() call.

**Note:**You can write React ‘code’ without JSX, but it means that you will need to format everything to fit with createElement() method requirements.

The next line after import React from 'react'; is import ReactDOM from 'react-dom';.

This line imports methods that are available from react-dom and makes it accessible through the Object named ReactDOM.

When you are rendering your React component, you are doing it via ReactDOM.

**Note:** the DOM is not something that is new or exclusive to React. It is part of HTML and lets you hook into different parts of the document. [Here’s a piece about DOMs](https://www.dottedsquirrel.com/shadow-dom-javascript/) I wrote a while back if you want to learn more.

**Creating a Component Class**

JavaScript is [object-oriented by design](https://www.dottedsquirrel.com/object-oriented-javascript/). A React component you write is an extension of React’s Component class. This means that when you create a new React component, you are actually creating a *subclass* of React.Component.

Here is the syntax of how to do this:

class AnIttyBittyComponent extends React.Component {}

The idea of subclassing is beyond the scope of this piece but here’s a [comprehensive piece I wrote a few years back on object-oriented JavaScript (what it means and how it works)](https://www.dottedsquirrel.com/object-oriented-javascript/) If you’re new to JavaScript and JavaScript OOP, the piece might help you out.

The name of the above component is AnIttyBittyComponent and it is now a subclass of React.Component. When we subclass, we have access to all the methods that are available in the parent class. Because React.Component is now our parent class, we have access to a method called render(). We can use render() to return our JSX.

**Note**: It is a JavaScript convention to write all class names in UpperCamelCases. You don’t have to, but following convention makes it easier to quickly identify what kind of content type you’re dealing with.

render() is a function that always return something. In our case, it's a block of HTML code.

class AnIttyBittyComponent extends React.Component { render(){ return <h1>Hello! I am a squirrel</h1> } }

render() is a compulsory method required by all React.Component subclasses.

**How to call your React component**

After you’ve written your first React component, you now have access to it in your React project in this format: < YourComponentName />

To render it on the DOM, you can do so via ReactDOM.render().

ReactDOM.render() takes two parameters - what it's supposed to render and where to render it.

ReactDOM.render(   
 <MyIttyBittyComponent />,   
 document.getElementById('app')   
);

And that’s basically it for how a React component class works.

**React FAQ:**

**Can we declare more than one component in a single file?**

Yes! All you have to do is create another component class extension. Here’s a code snippet example:

class SomeComponent extends React.Component{   
 render(){}   
} class AnotherComponent extends React.Component{   
 render(){}   
}

**How do you import multiple objects from a library in React/JavaScript?**

Use the {}. Here is the syntax example:

import {Router, Switch} from 'react-router';

**Is component render() method the same as ReactDOM render()?**

No. Despite having the same name, they belong to different classes. This means that they are different and are completely unrelated.

**What else can React.Component class do?**

In addition to render() method, there is also the constructor() method. constructor() is invoked before the component gets added to the DOM and sets up the initialization of states and method bindings.

Another method that React.Component class offers is componentDidMount() and componentDidUpdate(). componentDidMount() runs right after the component is inserted into the DOM tree and is only invoked once. componentDidUpdate() runs every time the component updates.

What do we do with these methods? Well, you can put certain functions and additional method calls inside them to run based on the above-described situation.

**Can we return multiple elements inside a React.Component render() function?**

Yes. It’s how views are created. You can collate a collection of components together to form a view component. For example:

class SomeComponent extends React.Component{   
 render(){   
 <div>   
 <nav />   
 <blog />   
 <footer />  
 </div>   
 }   
}

Alternatively, you can use <React.Fragment> to return a collection of components without needing to wrap it around a parent HTML element. For example:

class SomeComponent extends React.Component{   
 render(){   
 <React.Fragment>   
 <nav />   
 <blog />   
 <footer />   
 </React.Fragment>   
 }   
}

**Can we pass in multiple components into ReactDOM.render()?**

No. ReactDOM.render() only accepts one value in the first argument. However, if you want to create more than one component at a time, you can always wrap it with a parent element.

Here’s an example:

ReactDOM.render(   
 <div>   
 <nav />   
 <blog />   
 <footer />   
 </div>,   
 document.getElementById('app') );

**What happens if my React container already has things in it?**

Let’s pretend this is the container code you have:

<div id="app">   
 <h1>   
 Whoops! You don't have JavaScript support on your browser. Akward.   
 </h1>   
</div>

When we run ReactDOM.render() , all the contents inside the target container will be wiped and get replaced by our component instead. To the user, the <h1> in the code sample above would be as if it never existed.