**Learn React: Variables, Conditionals and Event Listeners Inside A Component –**

[Aphinya Dechalert](https://medium.com/@PurpleGreenLemon?source=post_page-----9acc61750fcd--------------------------------) Nov 6, 2021

Part 4 of Learn React from Scratch Series

Previously on ***Learn React from Scratch Series*:**

React Components can appear complex but it’s not that hard once you’ve got the general structure of it. Here’s a code snippet of what a component in React looks like:

import React from 'react';  
import ReactDOM from 'react-dom';class SomeComponent extends React.Component{  
 //some code here  
   
 render(){  
   
 //some more code here  
   
 return (  
 //return HTML block here  
 );  
 }  
}

For single-line HTML blocks in a React component, you can just write it like this:

class SomeComponent extends React.Component{  
 render(){  
 return <h1>Hello!</h1>;  
 }  
}

For multi-line HTML blocks in a React component, you need to surround your return block with a pair of () brackets. Here is an example of the syntax:

class SomeComponent extends React.Component{  
 render(){  
 return(  
 <h1>  
 <span>multi-line yeeeha</span>  
 </h1>  
 )  
 }  
}

To add JavaScript variables to your render(), use {} braces. Here is an example:

import React from 'react';  
import ReactDOM from 'react-dom';const squirrels = {  
 alt: 'dotted squirrel',  
 src: 'https://dottedsquirrel.com  
}class PrintSquirrel extends React.Component{  
 render(){  
 return(  
 <h1>  
 <a href={squirrels.src}>{squirrels.alt}</a>  
 </h1>  
 )  
 }  
}ReactDOM.render(<PrintSquirrel />, document.getElementById('app'));

So that’s the basics/revision covered. Let’s move into putting a logical condition inside your React component class.

**How to Acess Variables Inside A React Component class**

Sometimes you just need a little logic or variable to help display the right things. In a React class component, you can do this inside the render() . Just add your code before the return. Here is an example:

import React from 'react';  
import ReactDOM from 'react-dom';const squirrels = {  
 alt: 'dotted squirrel',  
 src: 'https://dottedsquirrel.com  
}class PrintSquirrel extends React.Component{  
 render(){  
 const n = Math.floor(Math.random() \* 10 + 1);   
 return(  
 <h1>  
 <a href={squirrels.src}>There are {n} squirrel/s.</a>  
 </h1>  
 )  
 }  
}ReactDOM.render(<PrintSquirrel />, document.getElementById('app'));

Why do we have our logic inside render() and not as a separate block outside? It's to do with scoping and variable accessibility. If you put your variable outside of the render() function, you will need to use the keyword this to access it.

In short, this lets you access variables the closest block scope. Here is a quick sketch of the accessibility:

Diagram

Description automatically generated

*The above diagram is from another article I wrote. You can check out the entire infographic and explanations about*[*this, let, and var relationship here.*](https://www.dottedsquirrel.com/var-let-this-javascript/)

So, if you had your variable outside of render() but inside the component class, you code would look something like this:

import React from 'react';  
import ReactDOM from 'react-dom';const squirrels = {  
 alt: 'dotted squirrel',  
 src: 'https://dottedsquirrel.com  
}class PrintSquirrel extends React.Component{  
   
 const n = Math.floor(Math.random() \* 10 + 1); render(){  
 return(  
 <h1>  
 <a href={squirrels.src}>There are {this.n} squirrel/s.</a>  
 </h1>  
 )  
 }  
}ReactDOM.render(<PrintSquirrel />, document.getElementById('app'));

Both methods presented in the code snippet above are both valid — it’s just a matter of if you want to use this or not.

**Writing an if statement inside a React Component**

You can run an if statement inside a React component. Here is a code sample of what it looks like:

import React from 'react';  
import ReactDOM from 'react-dom';class PrintSquirrel extends React.Component {  
 render(){  
 let text;  
 if(squirrel == 1){  
 text = "A squirrel jumped over the big grey fox";   
 }else{  
 text = "There are bazillion squirrels in the forest";  
 }  
   
 return(  
 <div>  
 <h1>Introduction to gibbrish</h1>  
 <p>{text}</p>  
 </div>  
 )  
 }  
}ReactDOM.render(<PrintSquirrel />, document.getElementById('app'));

If you have your conditional if block and text outside of render(), you'd need to use this in order to access it.

**Attaching a function on an event listener in React**

To render a function on an event listener, all you have to do is call it with the {} as per usual. Remember to call it as a variable {example} rather than as a function {example()} - or else the function will run when it first loads rather than wait for it to get triggered by the event listener.

Here is the code sample snippet on how to write attach a function on an event listener in React:

import React from 'react';  
import ReactDOM from 'react-dom';class SomeClass extends React.Component {  
   
 let text = "pending...."; runHoover(){  
 text = "woosh!";  
 }  
   
 render(){  
 return(  
 <h1 onHover={this.runHoover}>{this.text}</h1>  
 )  
 }  
}ReactDOM.render(<SomeClass />, document.getElementById('app'));

**React FAQ:**

**Do you need to use variables to set a component’s attributes?**

The quick answer is — it depends. If you know that your attributes are not going to change, then you don’t need to use variables. However, if you need them to change over time based on context, then you’d need to use variables.

**When should I put my logic/code inside or outside of**render()**in a React component?**

Where you put your ‘other’ and additional code depends on its length, complexity, and reusability. Here are the scenarios:

* If you know that you’re going to use the function somewhere else, you can lift the state and put it outside the class so that it is accessible by other classes. However, you’ll need to be aware of global scope pollution as your project grows. What is global scope pollution? When you have a lot of stand-alone functions and variables that sit in the global state, which can lead to overlaps and namespace conflicts.
* If your code is short and simple, it can sit inside render()
* If your code is longer and has complex logic involved, it is better to put it outside of render()

Ultimately, it is up to you where you put your logic. Just remember that if you put your logic outside of render(), you'll need to use the this keyword to access it.

**Can we use multiple event handlers in a single event?**

Yes. Just put your event handlers into an anonymous function that can be triggered when the event occurs. Here’s the syntax/code snippet on how you can do so.

import React from 'react';  
import ReactDOM from 'react-dom';class SomeClass extends React.Component {  
   
 let text = "pending...."; runHoover(){  
 text = "woosh!";  
 } runBleepBlop(){  
 console.log('Hoover goes bleep blop');  
 }  
   
 render(){  
 return(  
 <h1 onHover={() => {  
 this.runHoover();  
 this.runBleepBlop();  
 }}>{this.text}</h1>  
 )  
 }  
}ReactDOM.render(<SomeClass />, document.getElementById('app'));