

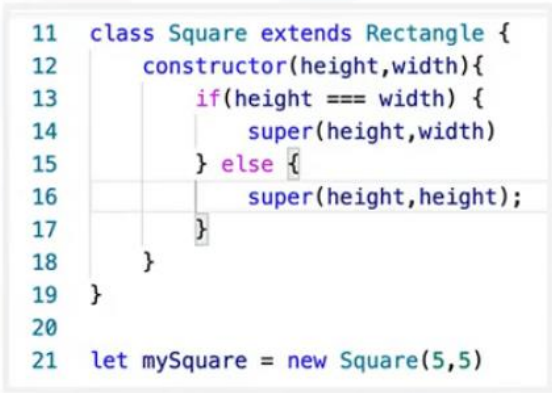
## Developing Cloud Applications with Node.js and React

### Module 4 Cheat Sheet: Building a Rich Front-End Application using React & ES6

## ES6

Package/Method	Description	Code Example
<b>let</b>	This keyword allows you to declare a variable with block scope, meaning that the variable can be used only within the block where it is created. You can change the value of a let variable within this block.	<pre>//last line throws an error because num is out //of scope {   let num = 5;   console.log(num);   num = 6; } console.log(num);</pre>
<b>const</b>	Allows you to declare constants whose values cannot be changed.	<pre>//last line throws an error because num is defined as //a constant with a value of 5 const num = 5; console.log(num); num = 6; console.log(num);</pre>
<b>Arrow function examples</b>	<p>The sayHello() function doesn't take any parameters and has only one statement. Notice that there are no curly brackets.</p> <p>Arrow functions can return values. The oneParamArrowFunc() has one parameter. The function brackets are not mandatory if it does not have a return value. However, because this one does return a value, the declaration must be in curly brackets.</p> <p>The twoParamArrowFuncWithReturn() function has brackets around the parameters because it has two</p>	<pre>const sayHello = ()=&gt; console.log("ES6 function Hello World!");  const oneParamArrowFunc = name =&gt; {return "hello " + name};  const twoParamArrowFuncWithoutReturn =   console.log(first, last) =&gt; { "hello " + first +   " " + last};  const twoParamArrowFuncWithReturn = (first, last) =&gt;   {return "hello " + first + " " + last};</pre>

	parameters. It has curly brackets around the return statement.	
<b>this</b>	Here, "this" refers to the current object. "this" is used in the same context as in most object oriented languages.	<pre> function Person(name, age){   this.name = name;   this.age = age;   return this; } let person1 = Person("Jason", 50);  //prints the entire prototype of the person1 //object. console.log(person1);  //prints the person1 name console.log(person1.name);  //prints the person1 age console.log(person1.age); </pre>
<b>constructors</b>	Constructors are special methods within the class, which are automatically called when an object of the class is created. In this example, when we create a Rectangle of height 10 and width 5, it passes 10 and 5 as parameters to the constructor and constructs the Rectangle.	<pre> class Rectangle {   constructor (height, width) {     this.height = height;     this.width = width;   } }; let myRectangle = new Rectangle(10, 5); </pre>

<p><b>super()</b></p>	<p>A subclass has a special privilege to call the superclass constructor with the super() method call.</p> <p>A square is a rectangle with the same value for width and height. Here, if the height is not the same as the width specified, the width will become equal to the height.</p>	 <pre> 11 class Square extends Rectangle { 12     constructor(height,width){ 13         if(height === width) { 14             super(height,width) 15         } else { 16             super(height,height); 17         } 18     } 19 } 20 21 let mySquare = new Square(5,5) </pre> <pre> class Square extends Rectangle {   constructor(height, width){     if (height == width){       super(height, width)     } else {       super(height, height);     }   } } let mySquare = new Square(5,5); </pre>
-----------------------	--	---

## React

<p><b>React, ReactDOM, Babel</b></p>	<p>Three important packages that you use to build React applications are the React package, the ReactDOM package, and the Babel package. The React package holds the React source for components and their states and properties. The ReactDOM package is the glue between React and the DOM. And Babel is the module which is available in most modern browsers. It is used to compile and interpret the JSX.</p>	<pre> &lt;html&gt; &lt;!--Load React API--&gt; &lt;script src= "https://unpkg.com/react@16/umd.production.min .js"&gt;&lt;/script&gt; &lt;!--Load React DOM--&gt; &lt;script src="https://unpkg.com/react- dom@16/umd/react-dom.production.min.js"&gt;&lt;/script&gt; &lt;!--Load Babel Compiler--&gt; &lt;script src= "https://unpkg.com/- standalone@6.15.0/babel.min.js"&gt;&lt;/script&gt; &lt;body&gt; &lt;div id= "comp1"&gt;&lt;/div&gt; &lt;script type= "text/bable"&gt; </pre>
--------------------------------------	--	--

	<p>You must include <code>react.production.min.js</code>, <code>react-dom.production.min.js</code>, and <code>babel.min.js</code> in the HTML.</p> <p>All React components must implement the <code>render()</code> method.</p> <p>You can then render the component using the <code>ReactDOM.render</code> method specifying the component name like a HTML tag and any attribute that you want to set, in this case the name attribute.</p>	<pre>class Mycomp extends React.Component{   render() {     return &lt;h1&gt;This is my own component named     {this.props.name}&lt;/h1&gt;;   } } ReactDOM.render(&lt;Mycomp name= "myBrandNewComp"/&gt;,document.getElementById("comp1")) ; &lt;/script&gt; &lt;/body&gt; &lt;/html&gt;</pre>
<b>Functional Component event handlers:</b> <b>onClick,</b> <b>clickEvent,</b>	<p>Event handlers for functional components can be set through properties. The <code>onClick</code> handler is the most used for functional HTML components. In this example, you can see which properties are being set to the App component.</p> <p>Here you are the property <code>clickEvent</code>, to an anonymous function.</p> <p>The App component will use the anonymous function set to the <code>clickEvent</code> property to handle <code>button_onClick</code> event by passing that property to the <code>onClick</code> event of the button inside the App component.</p>	<pre>function App(props){   return (     &lt;div&gt;       &lt;button onClick={props.clickEvent}&gt;Click Me!&lt;/button&gt;     &lt;/div&gt;   ); } export default App;  import React from "react"; import ReactDOM from "react-dom"; import App from "./App"; ReactDOM.render(   &lt;React.StrictMode&gt;     &lt;App color="blue" size="25" clickEvent={       ()=&gt;{alert("You clicked me!")}}     &lt;/&gt;   &lt;/React.StrictMode&gt;,   document.getElementById("root") );</pre>

<b>User-defined React Components render()</b>	<p>User-defined React components inherit <code>React.Component</code> and must override the <code>render()</code> method. User-defined components can have states and props. These components have a lifecycle that can be managed and maintained. Create the <code>App</code> class which extends or inherits from <code>React.Component</code>.</p> <p>To set properties to the component, pass the props to the constructor. Call the <code>super</code>, in this case the constructor of the <code>React.Component</code> class, with the props. In the case of the React component, the constructor in the derived class must call the super constructor. Override the render method which is the method that actually renders the component. The render method can only return one component. However, that component can have multiple child components.</p>	<pre>import React from "react"; class App extends React.Component {   constructor(props){     super(props)   }   render() {     return &lt;button       onClick={this.props.clickEvent}&gt;Click       Me!&lt;/button&gt;;   } } export default App;</pre>
---	---	--