

------------------------------------------------------------------------------------------------

\*The whole idea with React is this simple idea that we are going to make use of Components / Functions. This is why when you have a look at how we write our React code we have functions that have a return statement.

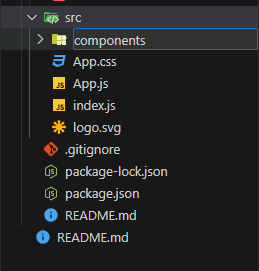
\*and the reason why we need to work with functions is very simple, functions are re-usable, and this means that we can create a function and then re-use it by calling it.

\*The basic idea with React is that we are writing Java script code, we are writing a Java script function that returns HTML.

------------------------------------------------------------------------------------------------

**How to Create our first Component/ function:**

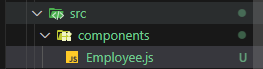
Step 1: Inside of our src folder, we are going to create a new folder called components.



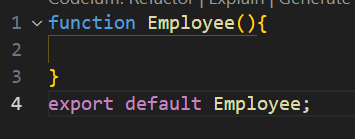
**Step 2**: Inside of this components folder, which is just a folder that will contain our components, we are going to create a component/java script file.

\*what we want to do is that we want to create a java script function that is called Employee, and this means that we need to name the file as “Employee.js”

\*this is very similar to working with classes in Java, because when we create a class in java, we have to name the file with the exact same name as the class name.



**Step 3**: we need to create this function called Employee

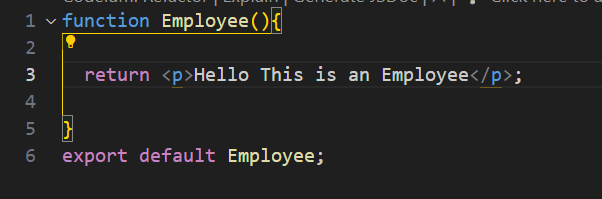


\*in order to create a function in java script we have to use the keyword “function” and then we have to give the name of the function, in this case the name of the function is “Employee” which matches with the name of the component file that we created.

\*it is a function so it must have the 2 brackets() that allow the function to take a parameter.

\*you also need that last piece of code at the bottom, the one that says export default Employee;

**Step 4**: we need to code the body of the function



\*I know that every Java script function must have a “return” keyword. Which is basically what I want the function to return / output. And in this case I want this function to simply output HTML paragraph text.

\*So all I have here is a Java Script function that returns HTML Paragraph text.

\*and I know that every Java script statement needs to end in a semi-colon.

**Step 5**: I need to use this component / Call the function:

1. The first thing that we need to take note of here is that a Java script function, is called inside of another Java script function.

\*The java script component that I want to call this Java script component, is called App.js

\*what you need to take note of here is simply the following, if we go back to the basic of Object Orientated programming, we have a main() function that becomes the entry point to our program. Meaning that this is the first function that is executed. This means that App.js is the main function. It is the function that we will use to call our component function.

\*the very first thing that we need to do inside of our Main function / App.js function is that we need to import our component file so that we can have access to the functions that are inside of the file.

\*So what we do is that we copy the relative path, and then we adjust it.

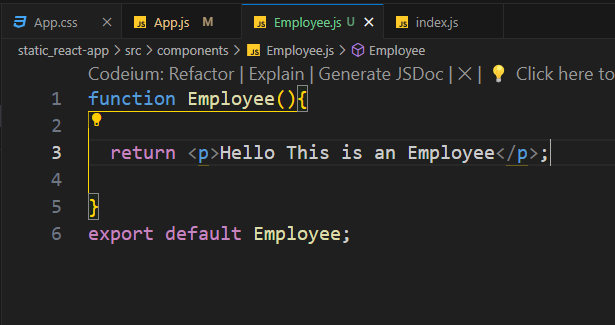


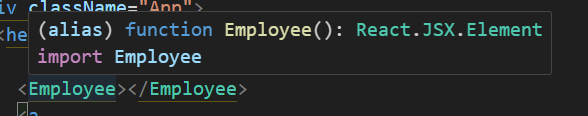
\*to adjust this all we want is the folder name and the file name.



\*we don’t use back-slash we use forward-slash. And then in front of the first forward-slash we have to put a dot.

**JSX**

****



\*If we hover above the function call for the function called Employee() this is what the intellisense is able to pick up.

\*what the intellisense picks up is that Employee() is a custom function, it is a component it is an alias. That’s what the words alias basically mean, they are there to tell us that we have a custom function, we have an alias.

\*Not only is the intellisense able to tell us that we have a custom function that we needed to import, but its able to tell us what the function returns. And in this case the function returns a React.JSX.Element, its not returning a html element, its returning a React.JSX.Element which in our case is equivalent to HTML.

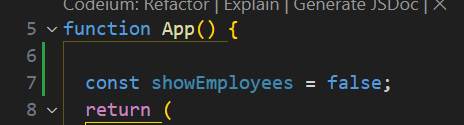
------------------------------------------------------------------------------------------------

Ternary Operators in Java Script:

\*what Java script allows us to do is to create a Boolean variable outside of our return statement, we are then able to code logic inside of the return statement that depends on the evaluation of the Boolean variable that we declared outside of the return statement.

How to create and use a Ternary operator in Java script:

1. The very first thing that we have to is that we need to go and create a constant variable outside of the return statement, and this constant variable has to be a Boolean.



\*that’s exactly what we have done here, we have created this constant variable called “showEmployees”, we have used the keyword “const” in order to show that it’s a constant variable, it’s a variable whose value cannot be changed later on. And we have assigned it the value false.

return(

<div>

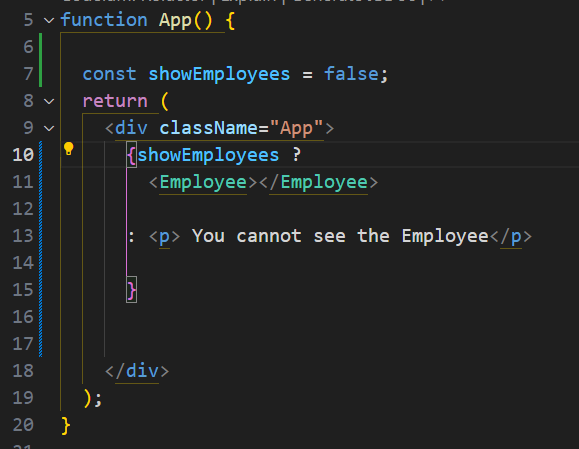
{

//java script code that uses a variable declared outside the return

}

</div>

);



------------------------------------------------------------------------------------------------

PROPS in React:

Props allow us to pass data from a parent component to a child component.

\*Props are very important, because they allow us to do 2 important things that we know from programming

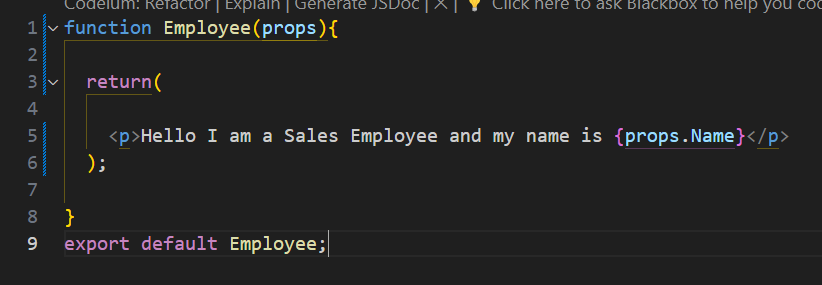
1) Props allow us to create a component / function that takes an arbitrary parameter.

\*this means that when we create the Employee() function, we can now create this Employee function with an arbitrary/prop parameter.

\*what we need to keep in mind here is that java script has what we call automatic type inference. This means that when we create a data type in java script, we don’t have to explicitly state the data type of the variable that we create, java script will look at what value we have assigned to the given variable and then based on the value that we have assigned to the given variable, java script will determine the data type of that variable.

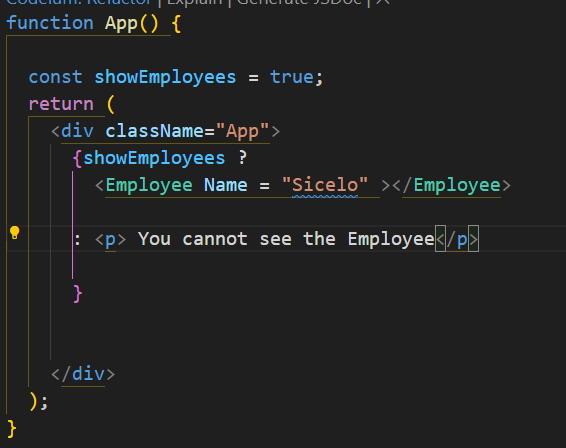
\*Once I have created my props, all I now have to do is to use the props inside of my JSX. And I use it as an Object, where I am able to give it a unique name, that identifies what the prop actually is.

\*the word props just means generic.



\*because I have a function that takes a props/generic parameter. What I need to do is that when I call this function, I need to provide an argument for the parameter.

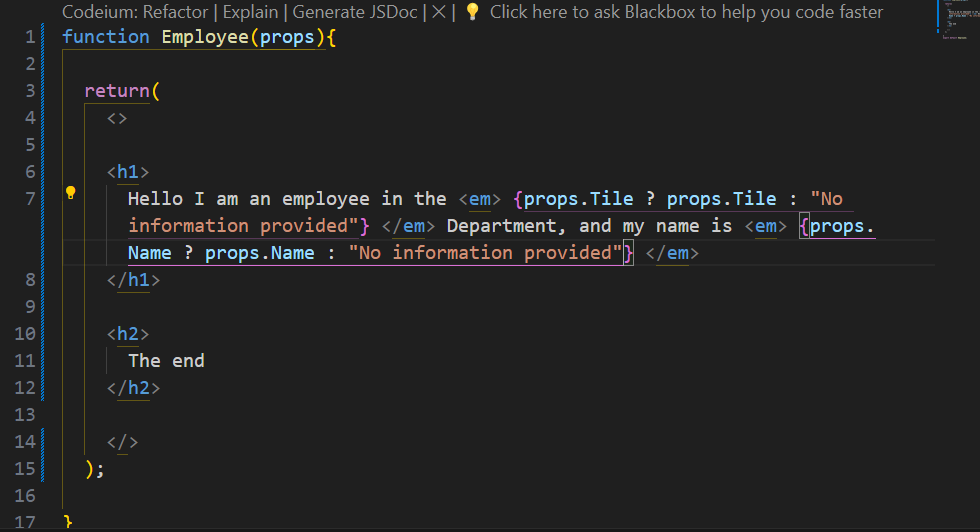
\*In this case I know I know that the function <Employee/> has a props parameter called props.Name, so when I call this <Employee> function inside of my main function, I need to provide an argument for this props parameter.



\*you can create as many props parameters as you need to, you just need to ensure that you when you call the function, you are able to provide arguments for all of the props parameters that you created In the order you create them on.

\*So you have basically created a props function, and this means that you have created a component/function that is a template, and its only supplied with information/data when it is called.

\*The word props simply means that we are creating a template that is only passed information when it is called, and again here we see the overall theme when it comes to React which is re-usability.



-----------------------------------------------------------------------------------------------

Introduction to Hooks in React:

\*Okay so here is the basic idea of whats happening and what we want to change.

\*So far we have learnt about props, and that the basic idea that props allow us to create what we call “props” function / components. So the basic idea here is that when we create our function, we can pass “props” as a parameter in the function, we can then create as many Objects using this parameter.

\*what we then had to do was to provide arguments for these props parameters when we called the function inside of what we call the main function / App.js

\*so when we called the function we needed to provide arguments for the props parameters.

\*so what we want to right now is that we no longer want to provide the arguments for the props parameters when we make the function call, we want to create an input field, and we want top pass the input from the field into the props Objects.

\*So the basics of what we want to do right now using Hooks is that we want to take user input and pass it as arguments for the props parameters.

**Pure HTML vs Hooks:**

\*the first thing that we need to highlight is why Hooks in React are a game changer. In classical HTML 5, we use an input field in order to get user input, but every input field has to be accompanied by a “submit” button, and only once we press the submit button is the page able to refresh and display the input that we gave.

\*The limitation is that HTML 5 doesn’t give you real-time keystrokes, what this means is the following:

1) If we were to create an App that requires someone to enter their banking pin = 12345 , what classical HTML 5 will do is that it will require the person to enter their whole pin(12345) and then they will be required to press the enter / submit button, and only once they have entered the submit button will their pin be verified.

2) what React does using hooks is that it will allow us to track the individual keystrokes that a person makes. This means that when a person hits the first keystroke which is “1” we are able to take this keystroke as a function argument and we are able to pass it to the function parameter and this means that we are able to verify this keystroke to see if it is correct. And this happens in real time as soon as the person types the keystroke.

\*and when the person enters the ‘2” we do the same. What React does via hooks is that it allows us to take keystrokes as real time input via a input field, we no longer need to have a submit button.

\*so the basic idea here with hooks is very simple, we no longer need to wait for a person to hit the submit button and then we verify whether their pin is wrong or right, the pin is verified per keystroke and this means that by the timer the person hits the submit button, we have already verified the pin.

\*What you also need to make sure that you understand when it comes to React is the Jargon that we use, there is clear difference between a Child component and a Parent component.

**Parent component:**

\*The parent component is the Main function, in this case it is the functional called App.js. It is the function in which we call the child function/ component. If you remember very well its in App.js that we are able to do the function call. Hence App.js is refereed to as the parent component.

\*It is also in the parent component that we have to import the child component.

\*so when we call the child component in the parent component we have to provide arguments for the props parameters. Where are these props parameters they are in the child component.

\*when we provide data input / arguments in React, we will always pass the data/arguments from the parents component to the child component.

**Child component:**

\*The child component is the component / function that has to be imported into the parent component

\*It is in the child component that we will have props parameters, it is the child component that is the template

|  |
| --- |
| So in here in essence this is what we are doing.   1. When we create props parameters in the child component/function, we are infact creating these Object containers that we will later fill with data. The reason why we have to use Objects container is because we want to create a generic container that we can later fill with data, any data. So in essence we are creating these Object container / placeholders and we will fill them with data when we provide arguments for them in the parent component/function 2. Now that we have these Object containers that we created in our Child component and we are supposed to fill them with data / arguments, its in the parent component, where we do the function call where we will have to provide the arguments/data that will fill these object containers. 3. But we have 2 options for providing this data/arguments, we can either hard code the arguments / data. Which is what we do in the function call when we specify the arguments as Name = “Sicelo”. The second option is to get the arguments via an input field that is able to record and process individual key stroke characters as they are typed into the input field. |

How to implement Hooks into React:

1. The first thing that we are supposed to understand here is that we parametrize the child component using props. And then we create props Objects inside of our semantic tags.
2. Once we have parametrized the child function/component, we then go ahead and we provide the arguments for these parameters in the parent function / component.

\*another important concept that we are supposed to understand when it comes to React is this idea of a state. A state is able to keep track of changes made to the components and then update the user interface with these changes without requiring a page refresh. This means that the state allow us to perform live real-time key stroke updates, this means that we are able to see the changes on the interface as we type them.

\*so the state allows us to process key-stroke user input via an input field using the interface it allows the parent component to process this input as a argument for the prop parameters and then pass this to the child function/component where it can be used as the data foe the Object place holder that was created using a prop, and then update the user interface with this placeholder data in real-time.

How to implement the coding:

1. We need to go to the parent component(App.js) and we need to create an input field.

\*we know that we want to create an input field that will appear on the user-interface. And we know that we want any data that is typed into the input field to be processed by the parent component as through it was an argument for the prop Object parameter placeholders that we created in the child component.

\*remember that when we hard-coded the arguments, we hard-coded them in the parent component, when we were doing the function calls, inside of the return statement. So that’s exactly where we will create this input filed in the parent component, just above the function call.

1. The first thing that we need to do is that we need to go and create an html input field. Where we simply specify the input type.

<>

<input type = “text”/>

</>

1. Inside of this HTML element tag, I need to have an attribute that is called onChange. Onchange is what we call an event. It is the event in java script that will allow us to record changes as they happen in real-time. It will allow us to record and track the changes that are happening to this input file in real time.

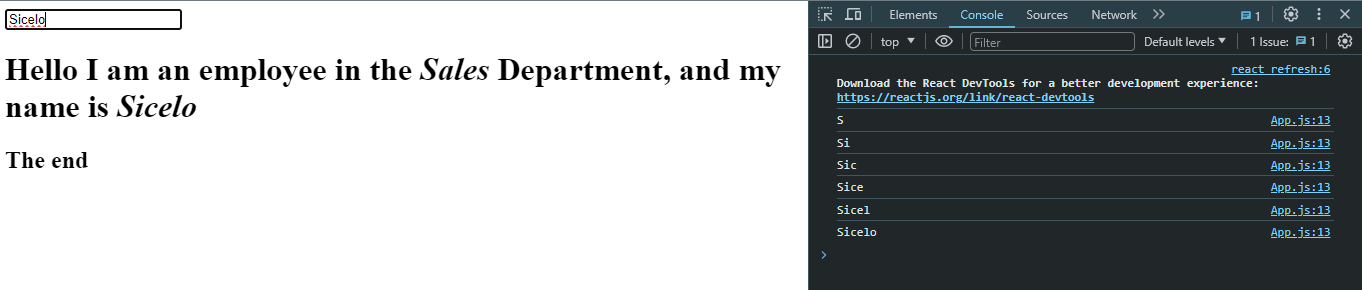


\*Okay so whats happening here is that we have an input text field, which is written in html, and for every html code that we write, we must have these opening and closing empty snippets <> </>

\*So what’s basically happening here is that I have this input field that has changed into a stack. And what this means is that I can only add values to the top of the stack and I can only remove values from the top of the stack.

\*But what I have done is that I have taken this stack and I am now able to index this stack from 0….n, and this means that I can add values where this n is. And I can also remove values where this n is.

\*and that’s what I have basically done by using this onChange = event. And because I have decided to consoile.log() this e.target.value I can actually see how this input field behaves like a stack



A screenshot of a computer

Description automatically generated

1. Thew first thing that we have to do is that we have to import {useState} from ‘react’