

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

\*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.

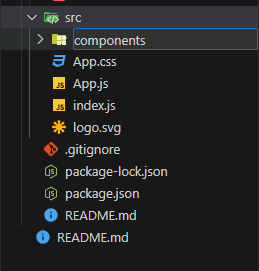
\*and the reason why we need to work with functions is very simple, functions are re-usable, and this means that we can create a function and then re-use it by calling it.

\*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:**

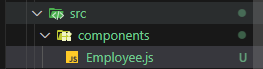
Step 1: Inside of our src folder, we are going to create a new folder called components.



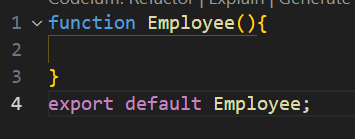
**Step 2**: Inside of this components folder, which is just a folder that will contain our components, we are going to create a component/java script file.

\*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

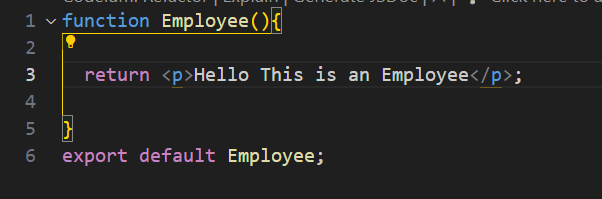


\*in order to create a function in java script we have to use the keyword “function” and then we have to give the name of the function, in this case the name of the function is “Employee” which matches with the name of the component file that we created.

\*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 JSX paragraph text.

\*So all I have here is a Java Script function that returns JSX

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.

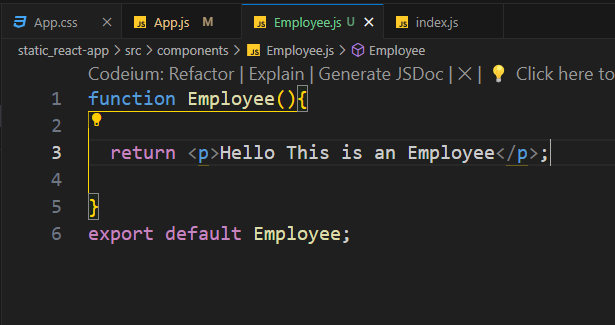


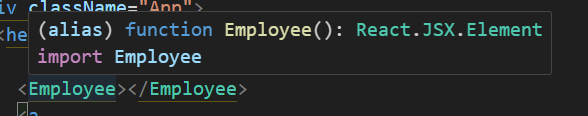
\*to adjust this all we want is the folder name and the file name.



\*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.

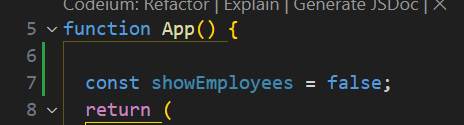
------------------------------------------------------------------------------------------------

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>

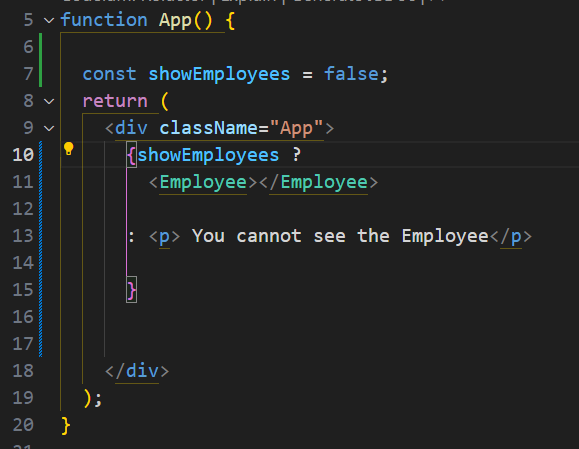
{

//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

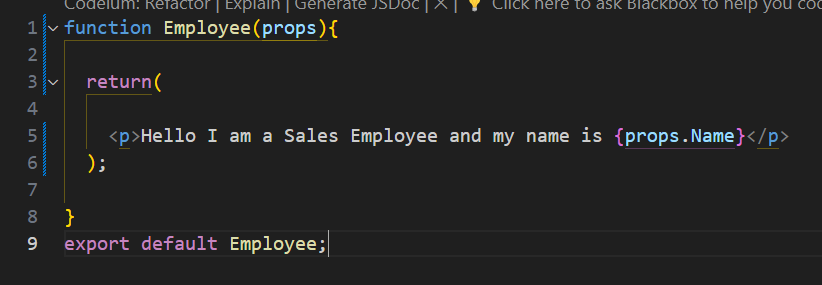
1) Props allow us to create a component / function that takes an arbitrary parameter.

\*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.

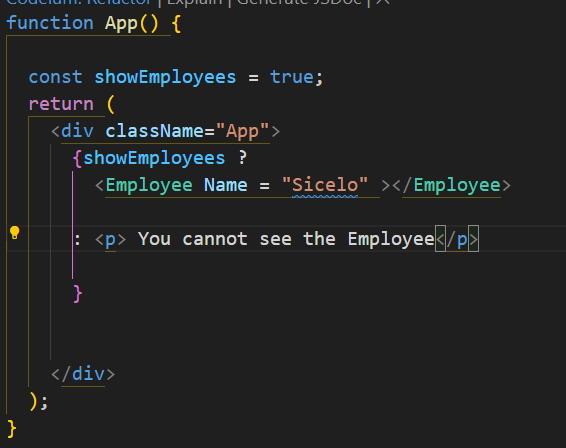
\*Once I have created my props, all I now have to do is to use the props inside of my JSX. And I use it as an Object, where I am able to give it a unique name, that identifies what the prop actually is.

\*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.

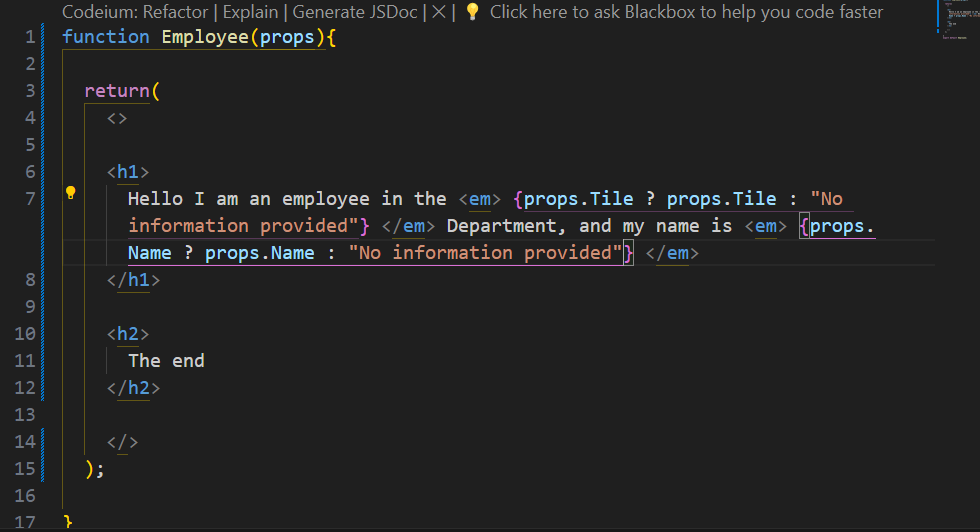
\*In this case I know I know that the function <Employee/> has a props parameter called props.Name, so when I call this <Employee> function inside of my main function, I need to provide an argument for this props parameter.



\*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:

**UseState**

***Scenario:***

\*we have been asked to design an interactive website for HP store, so HP store is a company that sells laptops. When a user lands on the hopme page for HP store they are introduced to a wide range of laptops that they can buy, and once a user has seen a laptop that they are interested in they have the option to add the laptop to their cart, and from the cart they have the option to either continue browsing or to checkout.

\*But the moment they add an item to their cart, what needs to happen is the following, we need to have buttons, a “+” and a “-“ button with a value in the middle, and because they have already selected an item and added that specific item to their cart the default value for these button is 1

e.g. (-) 1 (+)

\*and what needs to happen is the following, the user needs to be able to perform 2 key functionalities which are to increment(++) or decrement(--).

\*The key idea here is that we need to keep track of this value every time the user performs an operation, whether that operation is to increment(+) or decrement(-)

\*and we must also be able to control the value that is shown, we cannot allow the value to be less than 0. So the value can never be negative. So you cannot decrease this value below 0.

Coding:

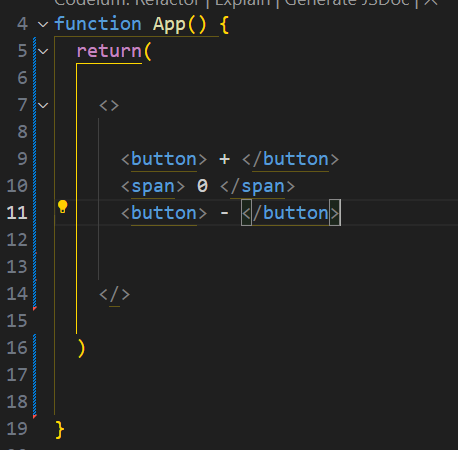
Step 1: We have to code the user interface

\*so the first step is that we have to create these buttons that vale a default value in the middle: (-) 0 (+)

<botton> -

<span> 0

<button> +



\*so whats happening here is that at this particular point we have just set up the user interface, we have just set up the front-end. We have not coded any functionality, this means that we have not coded the functionality that requires us to track the state of this value which has been defaulted to zero (0)

\*this means that when the user clicks on the (+) button, the value of will increment to a 1, and this means that we have a state change, from 0 to 1, and we need to be able to keep track of this new state.

Step2: we need to import the hook that we are going to use

\*The hook that we have to import and use here, depends on the functionality that we want to implement, we want to change and track the state of this value that is zero.

\*So the hook that we want to use is the “State” hook. So we have to import the “useState” hook

|  |
| --- |
| Import {useState} from React; |

How to use the {useState} hook

\*to use this hook, we simply have to call it as a function

useState(x)

\*Because of the functionality that we want to implement, using this hook, where we want to have a default state, and we want to be able to change this default state(increment / decrement) and we want to be able to track the changes that are made to this state when the user click on either of these buttons, it means that we this function call for {useState} = useState(), must have a certain number of arguments.

|  |
| --- |
| const [count, setCount] = UseState(x) |

X: is the default state

count = current\_value : is a variable that is used to represent the current state, in every iteration

\*this is important, because we have this default value which in this case is zero (x = 0) and we want to either increment (x+1) or decrement(x-1) this default value, and when we increment this value ( x + 1 =1) the 1 becomes the current value. And we need to a way to store this current value and keep track of this current value in each iteration.

setCount: is a variable that is used to change(increment / decrement) the default state

\*we know that we need to have a default state, and we need to be able to track the changes that we make to this default state, so it makes sense that the first argument that we have is the default state

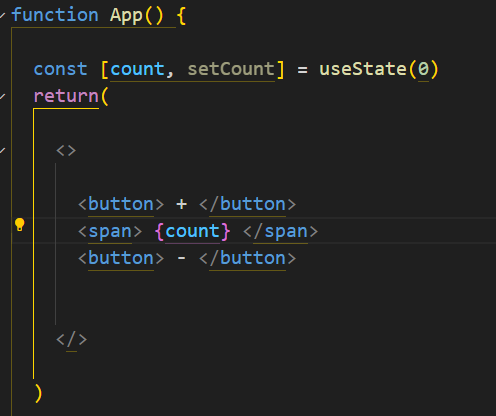
\*the second important thing that we need to understand about useState is that we need to set it up an an array [] that has 2 values inside. It has a variable that represents the initial state, as well as a function that allows us to change the initial state.

\*so what we can now do is that we can go into the default user-interface that we created and we can replace each of the values that we created in the default user-interface with the variables that represent them.



0 = count

\*even though zero(0) is the default state, it is the current state. And everytime we make a change this value of zero(0) will always represent our current state.

 A white square with black and white lines

Description automatically generated

\*immediately as we can see here the program recognizes that the current value for count is the defualt value of zero.

Step 3: we need to code the function for setCount which will be used to make the appropriate changes to the current value when the user interacts with the interace

\*we know from the scenario that was given to us, that in order for the user to change the current state which is represented by the variable “count” they have to interact with the front-end, and how they interact with the front-end is by clicking on the buttons which are either (+) = increment(+1) , or (-) = decrement(-1)

\*so what is the action that is performed by the user in order to change the current state ? they have to **“click” on the button**

\*so we need to understasnd the action that the user has to take, and the change in state that has to result from the action that user takes. So we have a action-reaction sitation, an input – output situation.

\*so because the user has to click on a button, the type of action we have is an “onClick” action

Action = “onClick”

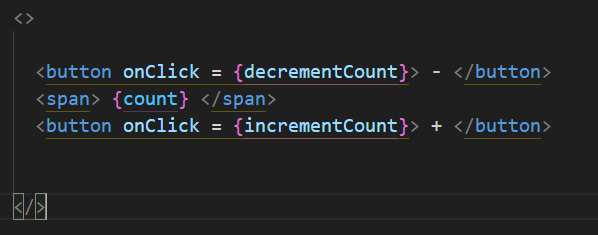
Reaction = “decrementCount”



\*so what this says is that when the user clicks on the button with a (-) sign the re-action that I want, the change that I want to see happening is that I want the count variable which represents the current state, I want the current state to decrement(-1)



\*I do the same thing for the (+) button, when the user click on the (+) button which is the action, the re-action that I want is that I want to decrement the current state (-1)



\*but the important thing that we need to understand here is that incrementCount and decrementCount are the re-actions that I want, they are the state changes that I want to see happening. It means that they are functions, that I need to go and define in terms of the variable that I created which is setCount

\*and I need to keep in mind here that whatever changes I perform here, will always be performed in relation to the current state.

Step 4:

A screen shot of a computer program

Description automatically generated

\*and these are my functions, they will both call the function that is called setCount and they will make the appropriate changes to the variable called count which is used to represent the current state of my variable.

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

Implement functional programming:

\*If I was to implement this logic in functional programming, how would I do it

\*how do I take a previous value and then then increment that previous value to create a new value in functional programming

|  |
| --- |
| fun prev\_value -> prev\_value + 1  where fun = means that I am creating a lambda function, a function without a name |

\*now react allows me to create lamba functions. All I have to do is to remove the keyword “fun” and then wrap everything around a bracket

|  |
| --- |
| prev\_value = prev\_value + 1 |

