

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

\*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.

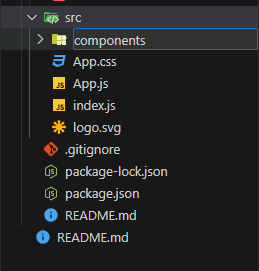
\*and the reason why we need to work with functions is very simple, functions are re-usable, and this means that we can create a function and then re-use it by calling it.

\*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:**

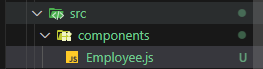
Step 1: Inside of our src folder, we are going to create a new folder called components.



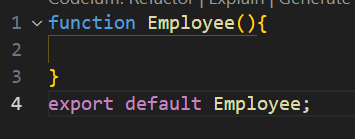
**Step 2**: Inside of this components folder, which is just a folder that will contain our components, we are going to create a component/java script file.

\*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

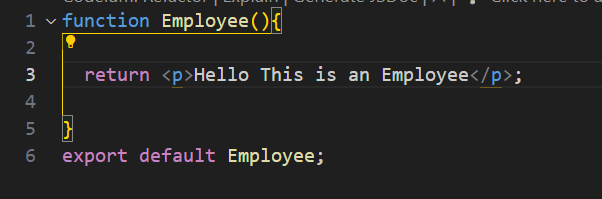


\*in order to create a function in java script we have to use the keyword “function” and then we have to give the name of the function, in this case the name of the function is “Employee” which matches with the name of the component file that we created.

\*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.

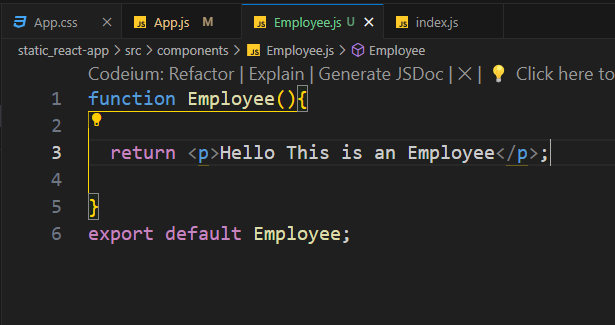


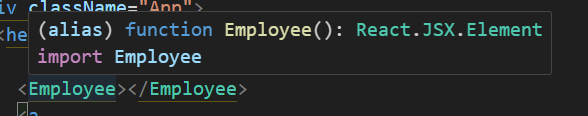
\*to adjust this all we want is the folder name and the file name.



\*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.

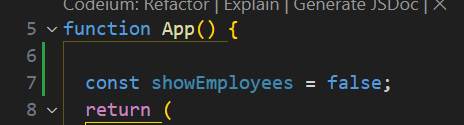
------------------------------------------------------------------------------------------------

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>

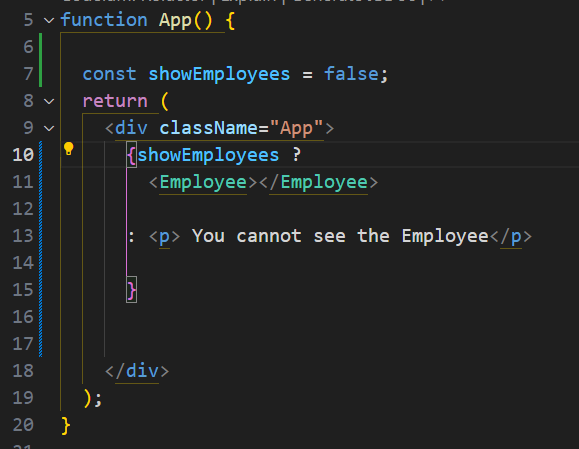
{

//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

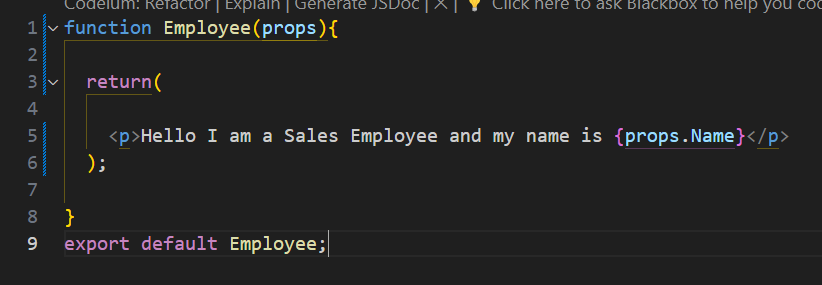
1) Props allow us to create a component / function that takes an arbitrary parameter.

\*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.

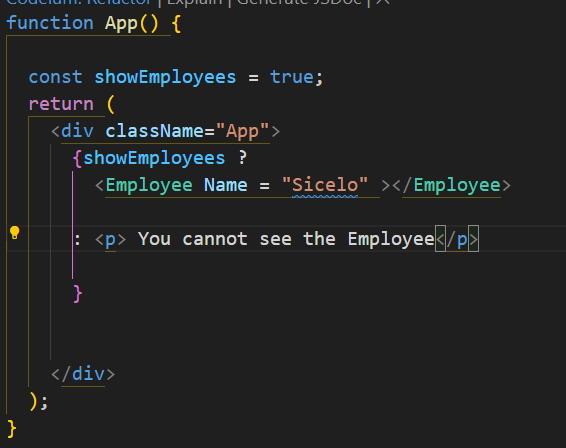
\*Once I have created my props, all I now have to do is to use the props inside of my JSX. And I use it as an Object, where I am able to give it a unique name, that identifies what the prop actually is.

\*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.

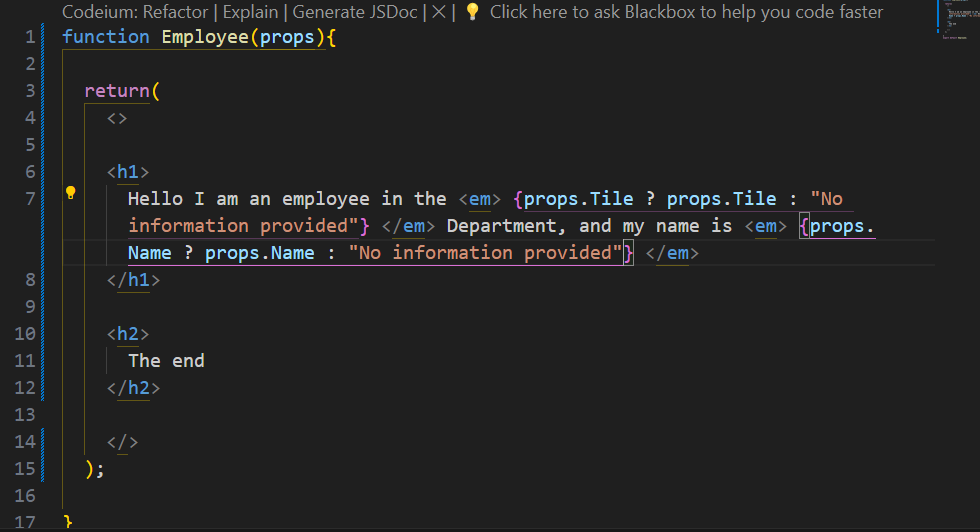
\*In this case I know I know that the function <Employee/> has a props parameter called props.Name, so when I call this <Employee> function inside of my main function, I need to provide an argument for this props parameter.



\*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:

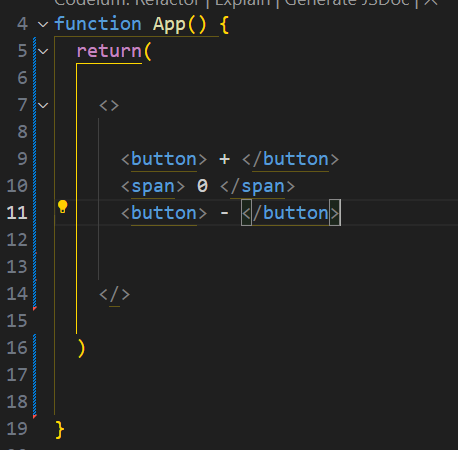
Step 1: We have to code the user interface

\*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

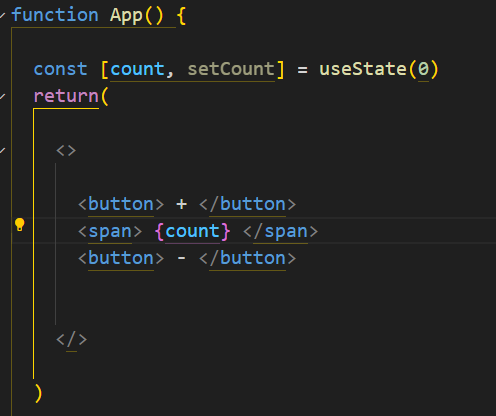
\*the second important thing that we need to understand about useState is that we need to set it up an an array [] that has 2 values inside. It has a variable that represents the initial state, as well as a function that allows us to change the initial state.

\*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”

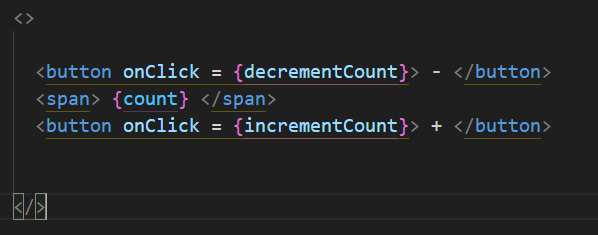
Reaction = “decrementCount”



\*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:

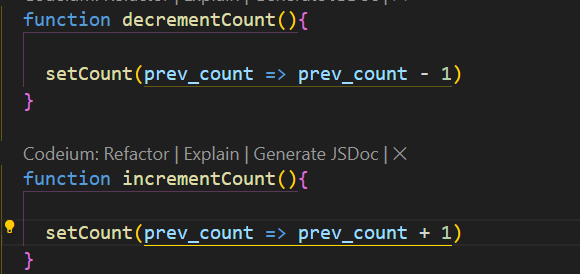
\*If I was to implement this logic in functional programming, how would I do it

\*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 |



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**useEffect Hook**

**Scenario \ UseCase**

\*We want to create a radio button, and what that means is that we want to create buttons that give a user choices, in this case the choices are the “Posts”, “Users” and “Comments” and how a radio button works is that a user can only select 1 option at a time, and if they select a new option that old option is overrideen with new data.

\*so we are still working with “States”, where we have a default state which represents the current state, and the user is able to change the current state, and create a new state by clicking on buttons.

\*So what is happening here is that I have buttons the user can click on. So the action that is take by the user is that the user can “Click” on a given button. But when the user clicks on a button there is functionality that I want to happen, there is a certain re-action that I want to see happening.

\*so when the user clicks on a button, which is used to represent the selection that they made or rather the choice that they have made, what needs to happen is that the text at the bottom needs to change to represent this new selection. So we will have a default state, and we will have to use these buttons to change the default state.



Step 1: I need to code the user Interface

\*The user-interface is what the user has to click on in order to interact with the system, and for this simple case the user has to click on 3 buttons which become the options that they have to select.

\*I need to code the buttons, and this is very simple, because I have buttons that contain text inside of them. This is pure html.

A screen shot of a computer

Description automatically generatedA close-up of a sign

Description automatically generated

Step 2: I need to code the functionality of the button

\*now that I have the buttons, that the user has to click on, in order to interact with the system, now I need to code the functionality, I need to code what I want to happen when the user clicks on these buttons, and I know that when the user clicks on a given button, I want the a state change.

Step 2.1: I need to go and create my useState Object function.

\*because I want a state change to happen when the user clicks on a button, it means that I need to have a {useState} object function. And this makes sense because I want to have a default state, which is the text “post” and I want the state to change when the user clicks on a given button, I will need to code a {UseState} object function.



\*so I need to create the LHS of my UseState function I need to pass it the default value which in this case is the text “post”

\*what I now need to work on is the LHS, where I have to create my const array which I know takes 2 values, a variable to represent the default state, and a function that allows me to change the default state.

\*Because you have radio buttons that are used to represent the different options that a user can select and they can only select 1 option at a time. It means that these buttons represent a resourceType. In React the moment you have a radio button it means that you have a resourceType.

**Lambda function:**

1. => B

A = input (previous\_state / default state / current\_state)

B = output, which is always the function call for the set function, which is the function that will be used to change the current state into the new state which becomes the default state and it represents the new state which becomes the current state

\*the first thing that we need to understand here is the relationship between the input state(A) and the output state(B). when we call the setFunction which is the function that is used to transform the current state into the new state. What we need to understand here is whether or not the output state depends on us knowing the input state.

\*Heres a simple example, when we were creating the counter, the output state depended on us knowing the input state. Because we were always incrementing the previous state(input) in order to create a new state(output). This is why in the counter lambda function we had to create a variable to represent the previous state, so that we could create the new state.

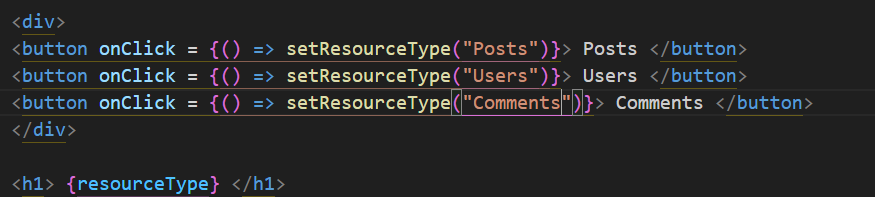
Increment\_count(){

(previous\_state) => previous\_state + 1

}

\*But because this is a radio button that is used to represent different resource types, and because I know that every time I select a new button the previous data is always overridden, it means that I don’t need to keep track of the previous\_state. What this means is that the output doesn’t depend on the input. So to create the output, we don’t need to know what the input(previous\_value) is. And this means that we don’t need to have a variable that represents the “previous\_value” so we can leave that empty and do a straight forward functon call that sets a new value (current\_value).





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

\*When we create a pure function: f(x) = y in functional programming, we create a function that takes an input (x) and produces an output (y). and this functional performs no side effects. That’s one of the conditions for creating a pure function.

\*Side effect: a side effect is basically anything that the function does besides producing an output. So an example of a side effect is printing output to the console or reading input from the user.

\*when we use the {useState} hook, like we have been doing, we were using the useState hook as thought it was a pure function, because all the lambda function that we coded, did 1 job and they did it well, they took input and they produced an output and they performed no side effects.

\*But what we want to do right now is to perform a side effect. And that’s what the “useEffect” hook allows us to do, it allows us to perform side effects.

Scenario / UseCase:

1. We used the {useState} hook, which is a pure function and it performs no side effects in order to a resource type button. Basically we just created a radio button, and what that means is that when the user selects a resource type(radio button) we are able to track the resource type that the user selected, and this means that we are able to track the selection that the user made.
2. So what is happening now is that we have an API, which is the interface between the front-end and the back-end, so basically we have a database somewhere that contains data that we need. And the only way that we can interact with this database and get this data is via an API.

\*so we have these radio buttons / resourceType, that the user can select in order to make a choice. So what we have is an API that contains data for each of the options that the user can select. So what we want to do is to fetch the data, based on the options that the user selects

3) But we only want to fetch the data provided that there is a change in state. So what this means is that if the user is given 3 options to chose from : A, B,C

If the select the option A, then the data that is stored under the resourceType = A should be retrieved and displayed, but they if they select A again, meaning that they are selecting the same resource consecutively then it means that we have no state change, so we don’t fetch the data from the API. So we need to have a hook that only produces a side effect when we have a change in state.

\*so in this case A,B,C are the 3 states that we have, and we only want to produce a side effect when there is a change in state.

|  |
| --- |
| https://jsonplaceholder.typicode.com/ |



A screenshot of a computer

Description automatically generated

1. How your json file is formatted is that you have the resource type and the resource. An example of this is the following. When you want to scrape data for a Book, a Book will the following the headings

|  |  |  |  |
| --- | --- | --- | --- |
| Book title | Author | Description | Price |

\*The headings/table rows are what we call the “resourceType” and then the data that is filled in each heading/row is called the resource.

\*what this means is that “posts”, “comments”, “albums”, “photos” ect are the resource types. When we click on the links for the resource types we get the resources

\*But notice how the resourceTypes are contained in a URL, call this URL the “end points”

|  |  |
| --- | --- |
| resourceType = posts | resource |

|  |  |
| --- | --- |
| resourceType = comments | resource |

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

Step 1: we need to import the {useEffect} hook

\*we know that we need to implement the functionality that allow our lambda functions to produce a side effect when there is a state change. And we know that we need to use the {useEffect} hook and the very first thing that we need to do here is to import the {useEffect} hook

Step 2: we need to create and use the {UseEffect} object function

\*The structure of the {useEffect} hook is very similar to the structure of the {useState} hook, in terms of how we set it up, and when we have to set it up, we have to set it up from the RHS(function call) to the LHS.

\*so this means that we always start with the function call, however the function call for the useEffect hook takes 2 arguments, a lambda function and a array. And this array takes a the name of the variable that we want to see a state change in, in order to see a side effect.

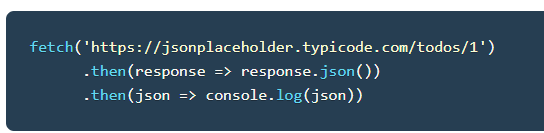
\*again its important that we understand what the useEffect hook does, it basically allow us to create a side effect. This means that whenever we have a state change, the useEffect hook will allow us to have a side effect , which is something that the useState hook doesn’t do.

\*when it comes to React and the {useEffect} hook, the most basic side effect that we can have is to interact with an API.

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

How to Query the API

1. To query the API we have to do a simple copy and paste of the code that allows us to fetch the data from the server.



\*This is just a simple fetch(‘Start\_URL’) command with a simple start url, similar to what we had in python scrapy .

\*and then we have lambda functions that take an input and produce an output.

\*what happens here is that the browser will issue a http get request(fetch) request to the server, and the server will respond to the get request, and return the data as a “response” hence the input becomes the response. The input is the response and the output is the response in json format. We are just changing the format of the data. And in the second function, we are taking the output from the first operation which is the json format, and we are producing a new output which is the console.log()