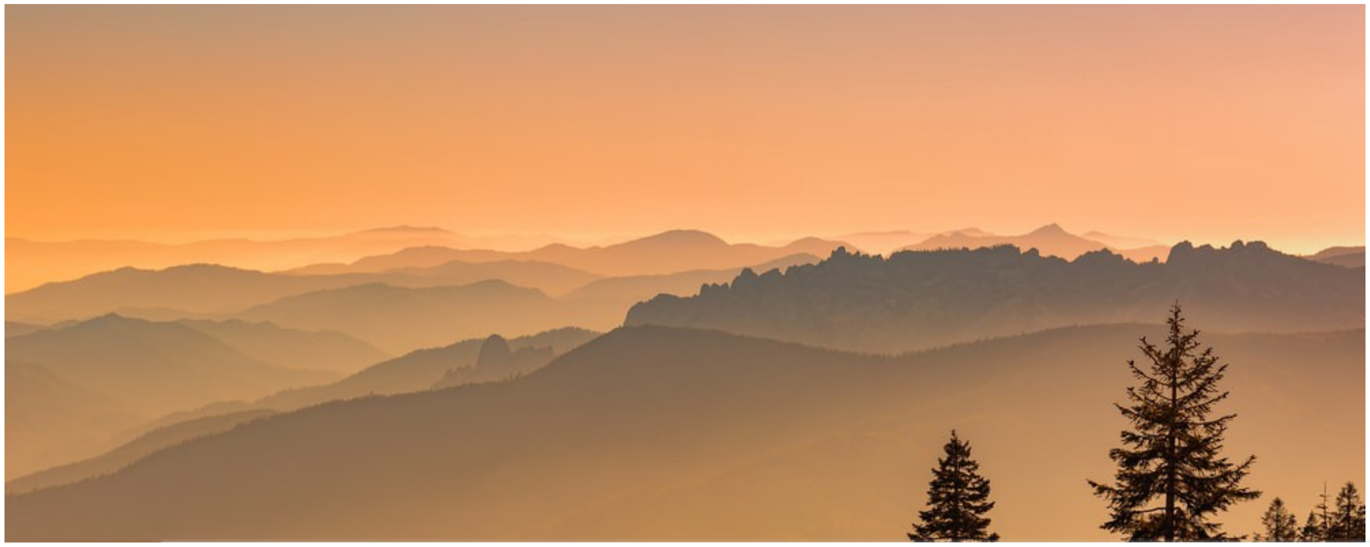
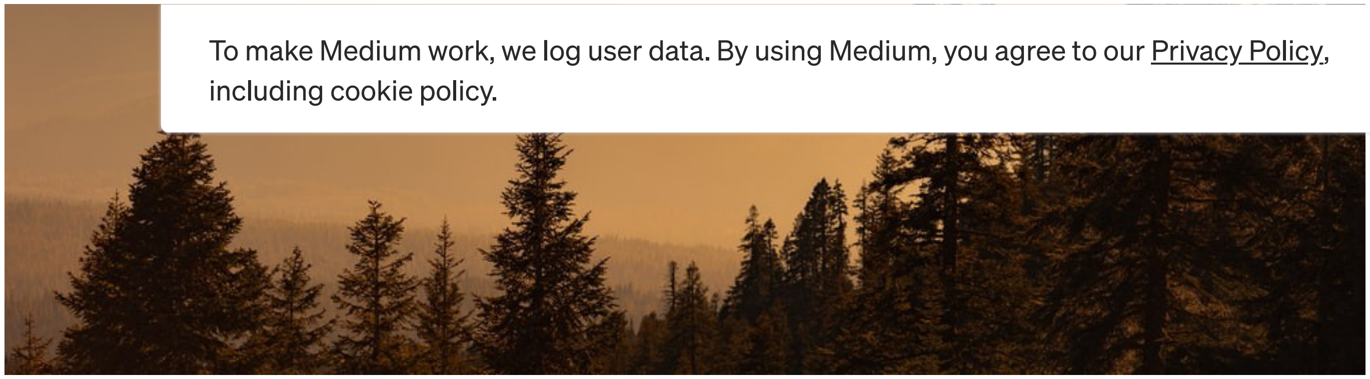
React's Context API Explained

All you need to know about React's Context API

Hussain Arif Follow
Jul 21 · 9 min read ★





Source: Dave Hoeffler On Unsplash.com

Nowadays, writing web applications can get extremely tedious and complex. If you're building a React app, you will sometimes send data from one component to another component. However, due to the complexity of your project, there might be times where you've had to send `states` and `props` from component to component, thus further increasing the complexity. This is where React's Context API can prove to be incredibly useful.

Let's first discover why we need the Context API.

What Is Context and Why Is It Useful?

As an exa
movies a

Let's first create a file called `Movies.js` and define states in it. These states contain values about the movies.

Movies.js

```
import React,{useState} from "react";
import MovieList from "./MovieList"
function Movies(props) {
   const [movies,setMovies]= useState([
      {
         name : 'Harry Potter',
         Price : 10
      },
      {
         name : 'Cheese Puffs: The Movie',
         Price : 55
      },
      {
         name: 'PlayStation vs Xbox: The documentary',
         Price: 10
      }
   ]);

   return (
   <div>
      {movies.map(movie=>(
    </
    )
}
export default Movies
```

MoviesNoContext.js hosted with ♡ by GitHub view raw

- `Line 21-24` : Use the `map` method to display all of the array items in the `movies` state through the `MovieList` component. The `MovieList` component simply displays the `name` property as a header and the `price` property as a paragraph element.

We have rendered the `MovieList` component in this file. The `MovieList.js` file contains the following code:

```
import React from "react"
function MovieList(props) {
return (
    <div>
        <h3>{props.movie.name}</h3>
        <p>{props.movie.Price}</p>
    </div>
)
}
export default MovieList
```

MovieListNoContext.js hosted with ♡ by GitHub view raw

This uses

refresher

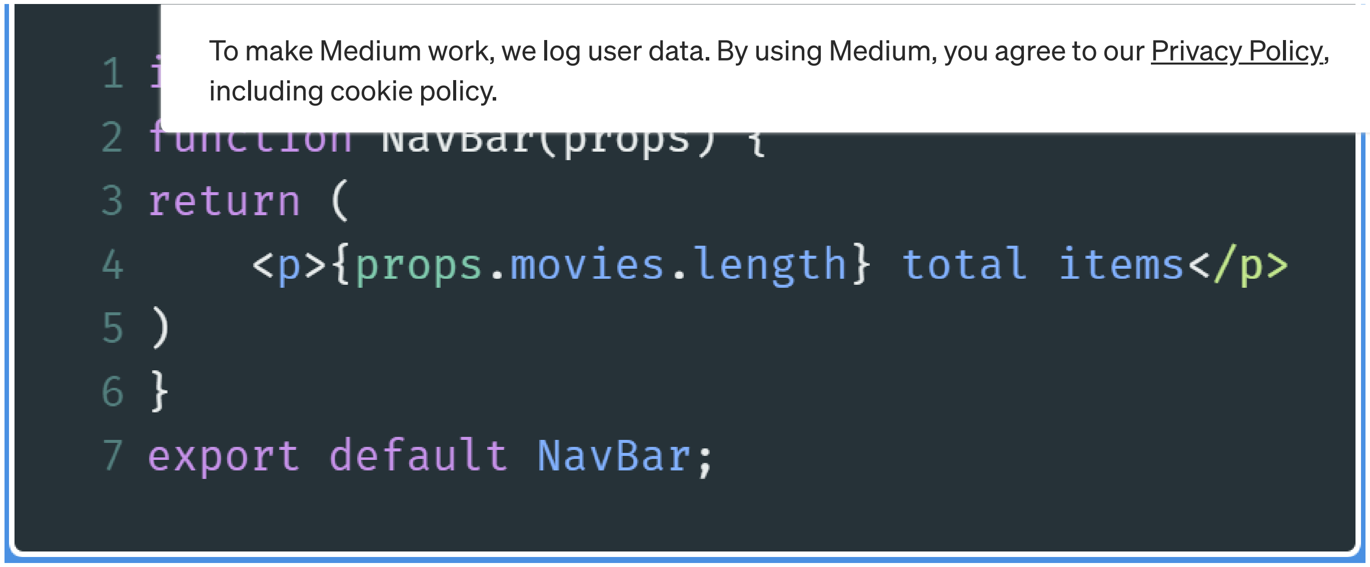
In `App.js`, all we need to do is render these components:

```
import React,{useState,useContext} from 'react';
import MovieContext from "./MoveContext"
import './App.css';

function App() {
  return (
   <div>
     <MovieContext/>
   </div>
    );
}

export default App;
```

AppNoContext.js hosted with ♡ by GitHub view raw

However, I now want to build a navbar, which will show me the number of movies:

`NavBar.js`



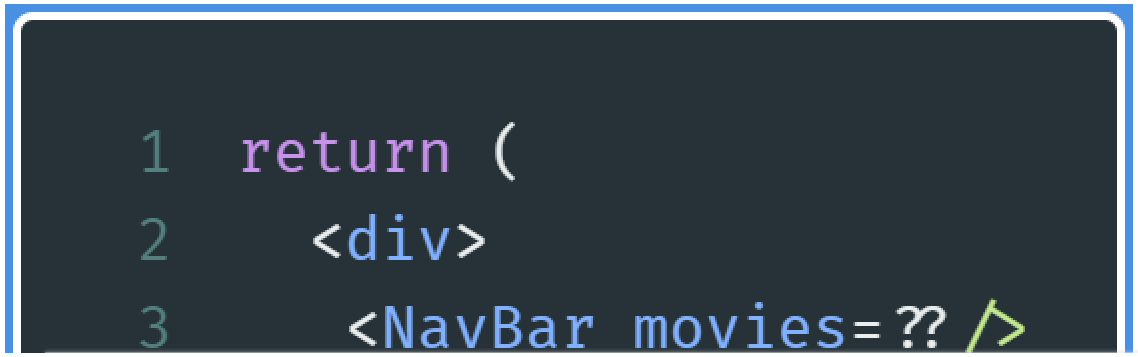
To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

```
i
function NavBar(props) {
return (
     <p>{props.movies.length} total items</p>
)
}
export default NavBar;
```

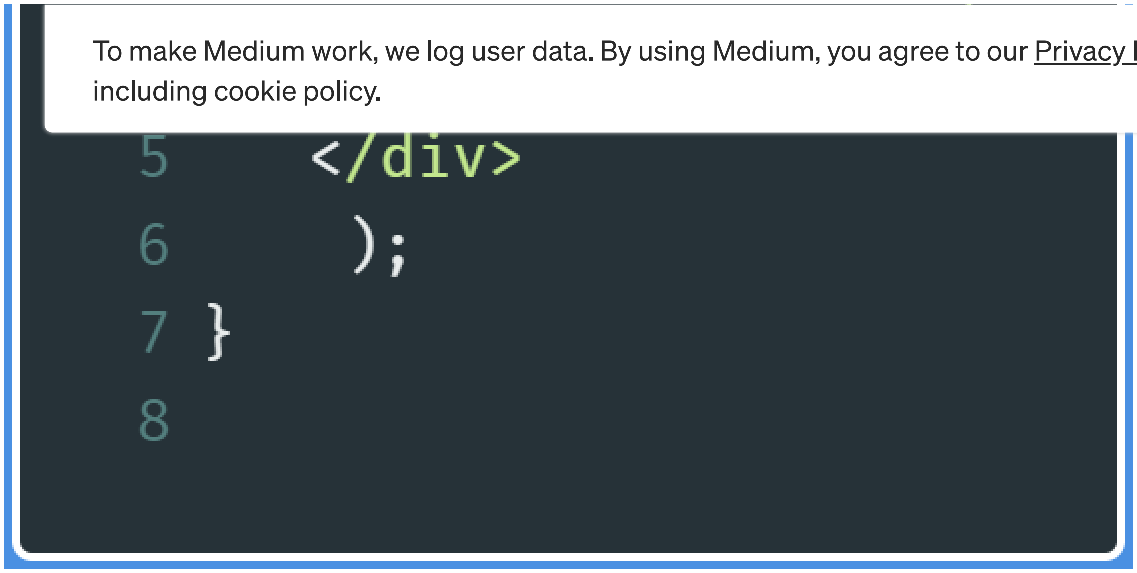
NavBar.js

We simply have to pass in `props` that will then return the number of items in the database. Let's just put it in `App.js`. This is where our problem occurs.

In `App.js`:



```
return (
   <div>
    <NavBar movies=?? />
```



```
    </div>
  );
}
```

A minor problem we have faced in App.js

What should we put in the `movies` prop in `NavBar` so that we can output a result? There is one solution: We can define our `movies` state in `App.js` and then pass down those into our `NavBar` component. We need to manually pass down props at every level. This point is important.

However, this workaround still contradicts our objective. It still makes our program appear complex.

Let's use Context to solve this problem. The React Context API provides the programmer a way to pass data from component to component without having to pass down props manually from each component to another.

Thus, thi
applicati

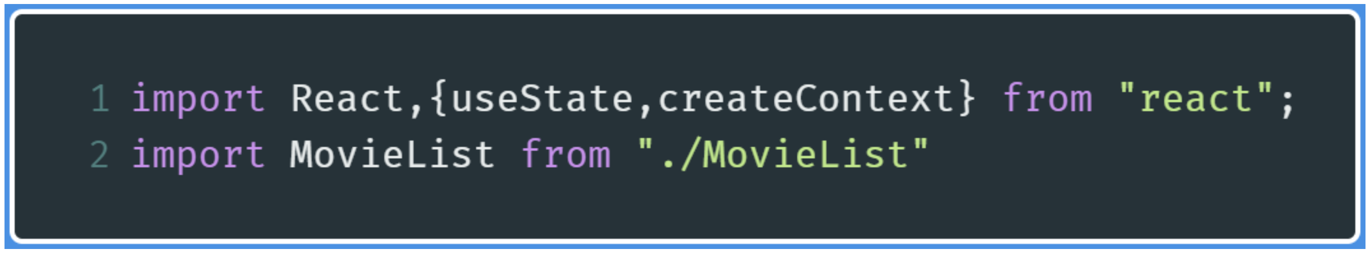
Let's first start by altering this application's code.

Usage of Context API

MovieContext.js

Let's start by renaming our `Movie.js` file to `MovieContext.js`. This file will be responsible for passing data to other components.

Here we will start by importing `createContext` from the `react` module. Thus, alter your `import` statement as follows:



```
import React,{useState,createContext} from "react";
import MovieList from "./MovieList"
```

Imports in MovieContext.js

This `crea`

will aid i

Next, let's `export` our `Context` instance like so:

```
export const MyContext= createContext();
```

Exporting the MyContext context instance

Now, we will export a function called `MovieContext` and then define it like so:

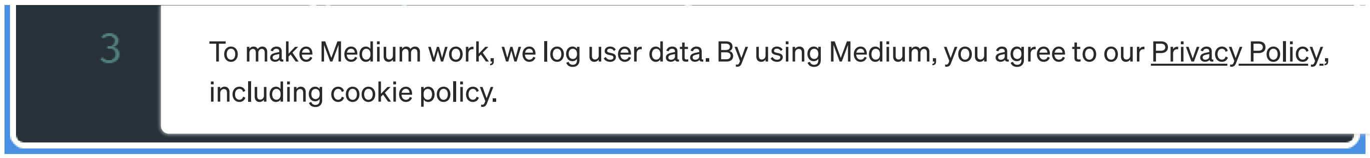
```
export function MovieContext(props) {
   const [movies,setMovies]= useState([
      {
         name : 'Harry Potter',
         Price : 10
      },
      {
         name : 'Cheese Puffs: The Movie',
         Price : 55
10
11
12             name: 'PlayStation vs Xbox: The documentary',
13             Price: 10
14         }
15
16     ]);
17
18     return (
19     //code goes here
20     )
21 }
22
```

Further code in MovieContext.js

- `Lines 2-16` : Standard definition of our `movies` state

In our `return` block, write the following code (where `code goes here` is written on `line 19`):

```
<MyContext.Provider value="This is a value">
      {props.children}
```



```
3
```

Further code in MovieContext.js

These lines of code indicate that we are now fully capable of sharing data between components without passing down props manually.

The `value` attribute in the `MyContext.Provider` tag in `line 1` is the data that we will share to various components.

`props.children` on `line 2` means that the components that will be rendered between the `MovieContext` tags will have access to the data located in `MovieContext`.

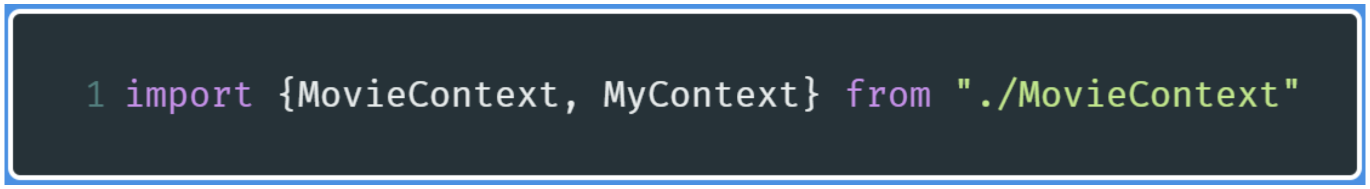
If this is unclear, don't worry. It will be explained through code later in this post.

MovieList.js

If you recall, this component was used to display the `movies` array.

Let's modify this file.

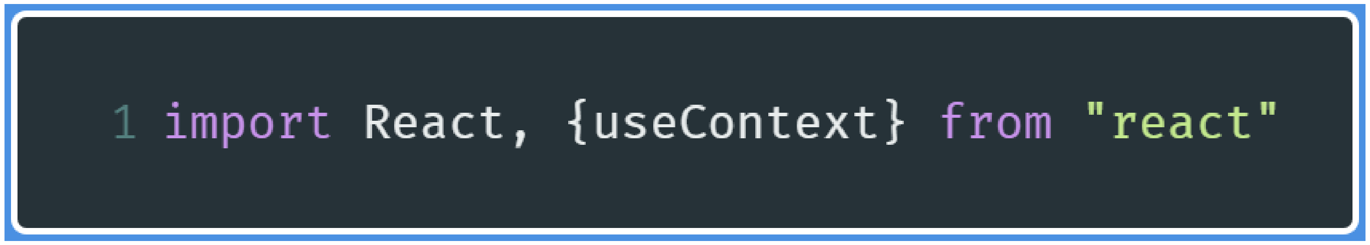
First, imp



```
import {MovieContext, MyContext} from "./MovieContext"
```

Imports in MovieList.js

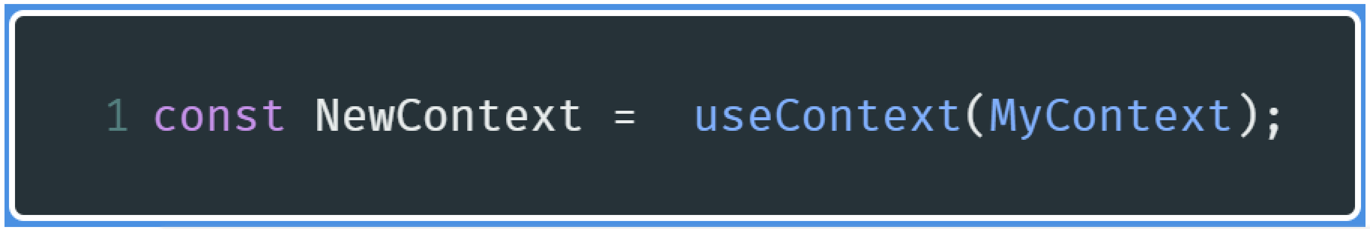
Other than that, import `useContext` like so:



```
import React, {useContext} from "react"
```

Further imports in MovieList.js

Within the `MovieList` function definition, start by writing the following line of code:

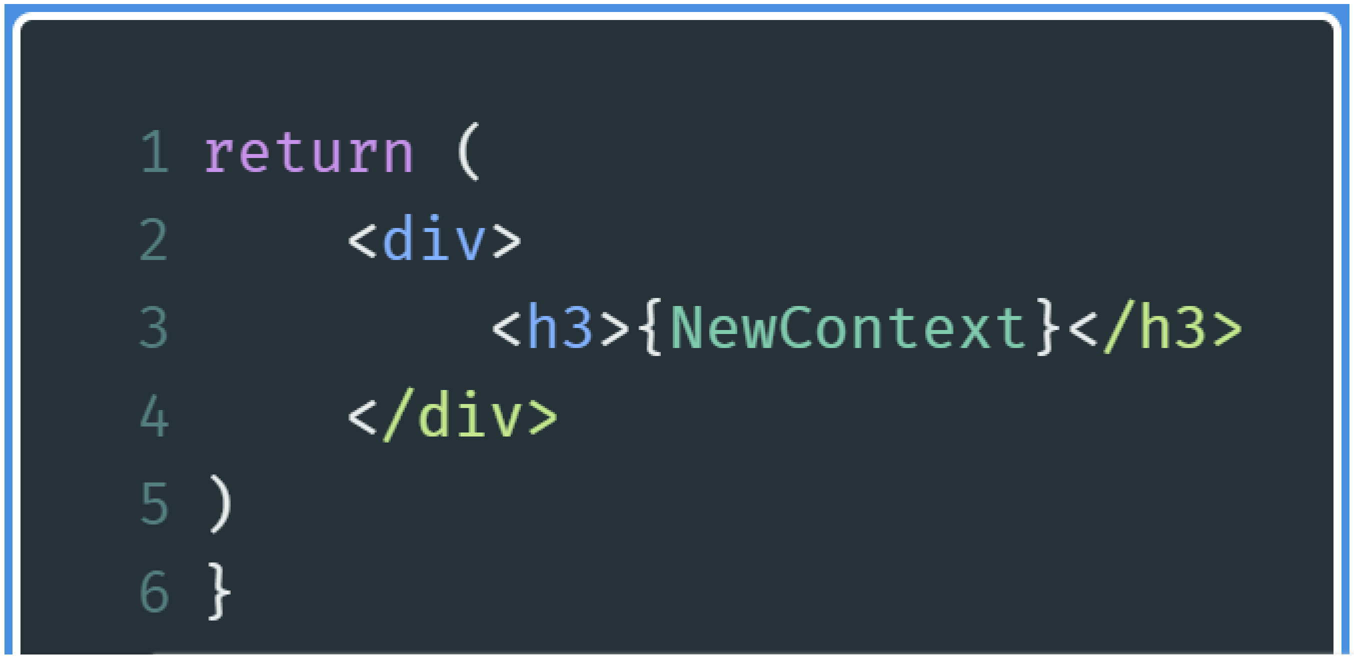


```
const NewContext =  useContext(MyContext);
```

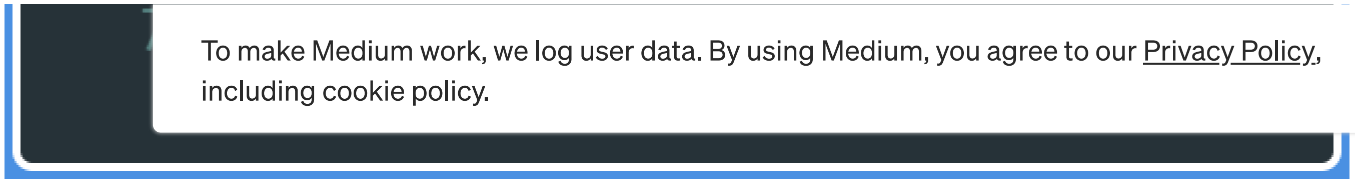
This line basically declares a `context` hook called `NewContext` . This `useContext` function takes in an argument that asks for what context object it should use. As we want to use the `MyContext` instance, we pass in `MyContext` as the argument.

One question though: What is the value of the `MyContext` variable? We'll find out its value shortly. Before that, we'll have to change our code further.

Now, write the following code after the `NewContext` declaration:



```
return (
    <div>
        <h3>{NewContext}</h3>
    </div>
)
}
```



Further code in MovieList

In this code, we are simply outputting the value of `NewContext` and then using the `export` statement so that the `MovieList` function can be used in other files.

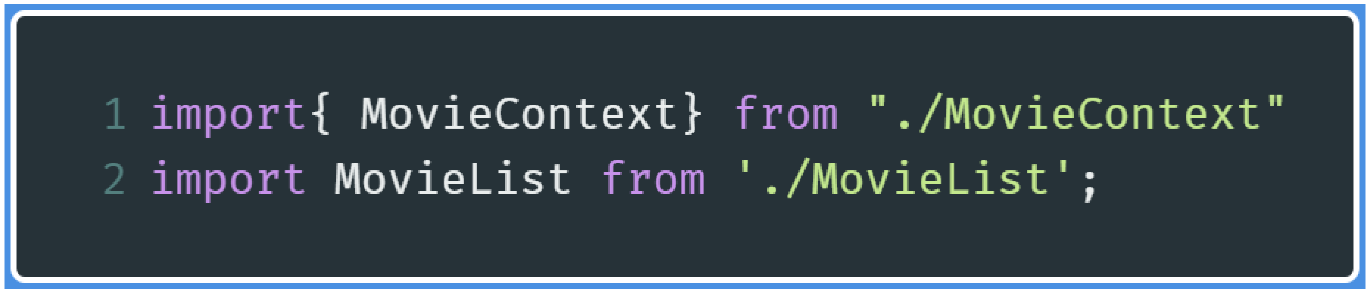
In the end, our file will look like this:

```
import React,{useContext} from "react"
import {MyContext} from "./MoveContext"
function MovieList() {
 const NewContext =  useContext(MyContext);
return (
    <div>
        <h3>{NewContext}</h3>
    </div>
)
}
export default MovieList
```

MovieListWithContext.js hosted with ♡ by GitHub view raw

App.js

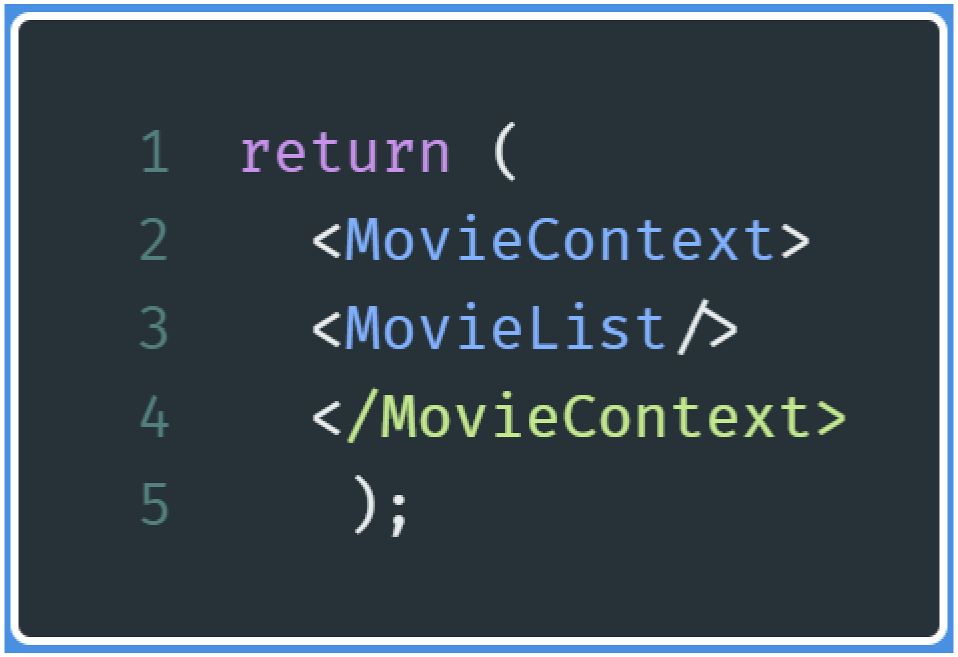
First, add



```
import{ MovieContext} from "./MovieContext"
import MovieList from './MovieList';
```

Imports in App.js

Now, in the `return` block, write the following code:



```
return (
   <MovieContext>
   <MovieList/>
   </MovieContext>
    );
```

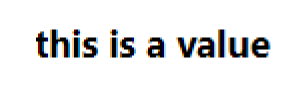
Further code in App.js

This cod

shared b

In the end, `App.js` looks like this:

Run the code. This will be the output:



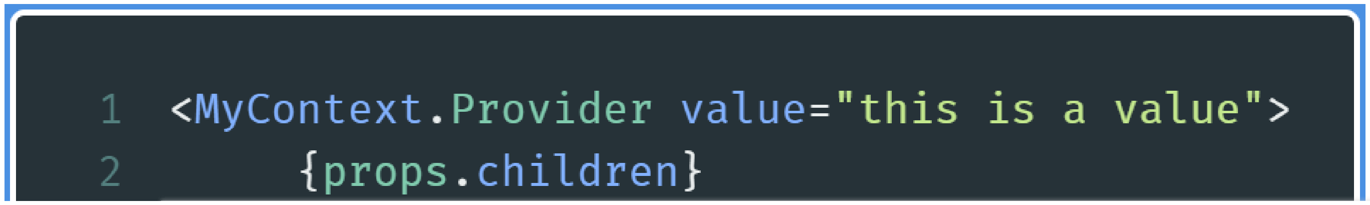
```
this is a value
```

The output of the code

So where did this string come from?

Let's backtrack to `MovieContext.js` and find the following piece of code:

In `MovieContext.js` :



```
<MyContext.Provider value="this is a value">
     {props.children}
```

Code to find in MovieContext.js

This means that the data we write in the `value` attribute will be shared with the components.

We have successfully shared a string as data using the Context API. Let's now move on to sharing the `movies` state.

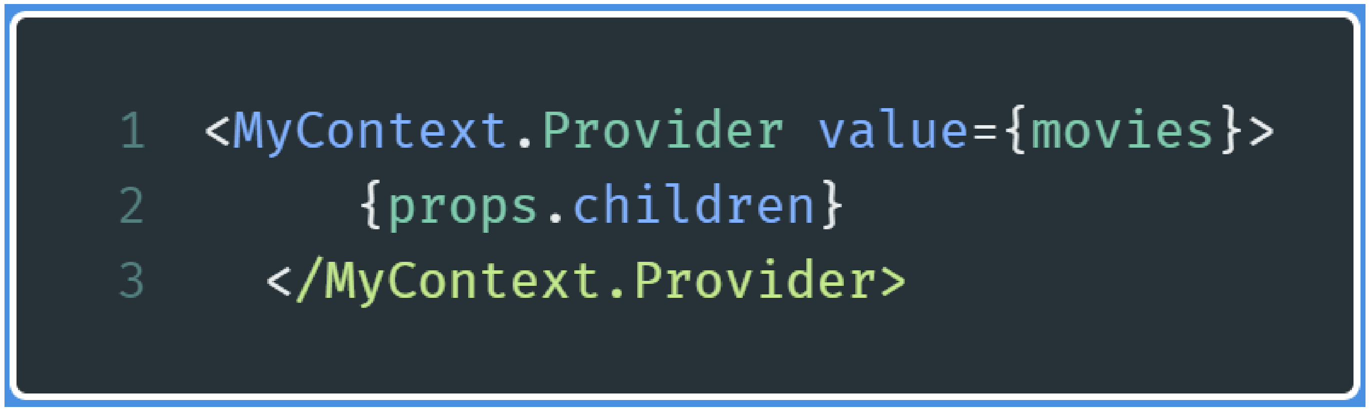
Sharing States With Context

MovieContext.js

In `MovieContext.js` , in the `return` block, find the following line of code:

```
<MyContext.Provider value="this is a value">
     {props.children}
  </MyContext.Provider>
```

Since we want to share the `movies` state, go to the `value` attribute and replace `this is a value` with the `movies` state like so:



```
<MyContext.Provider value={movies}>
     {props.children}
  </MyContext.Provider>
```

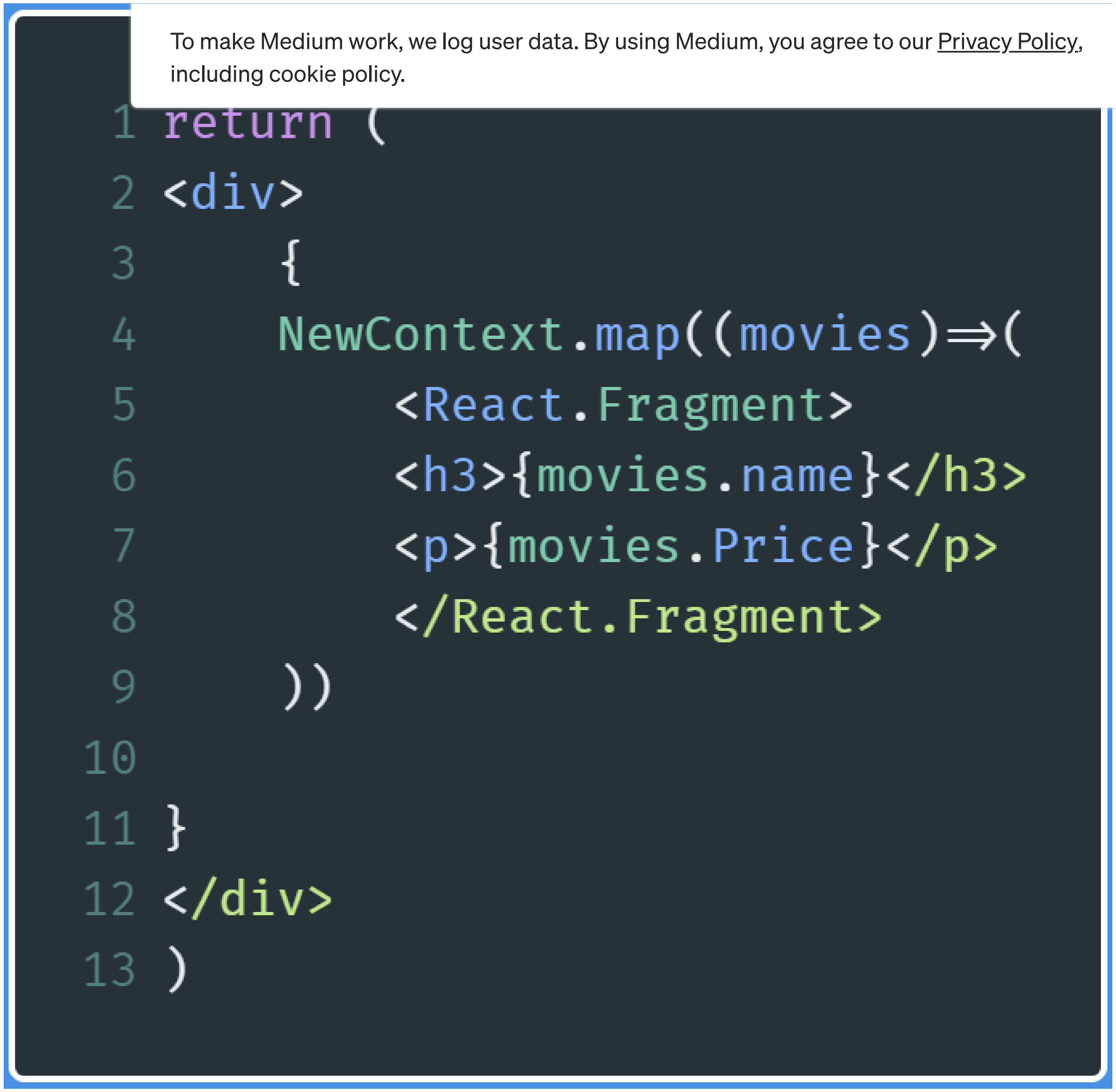
Code to replace in MovieContext.js

We have now passed the `movies` array as accessible data.

Since `movies` is an array, let's use the `map` method to display the array elements to the browser.

MovieList.js

Go to the `MovieList.js` file and modify the `return` block like so:



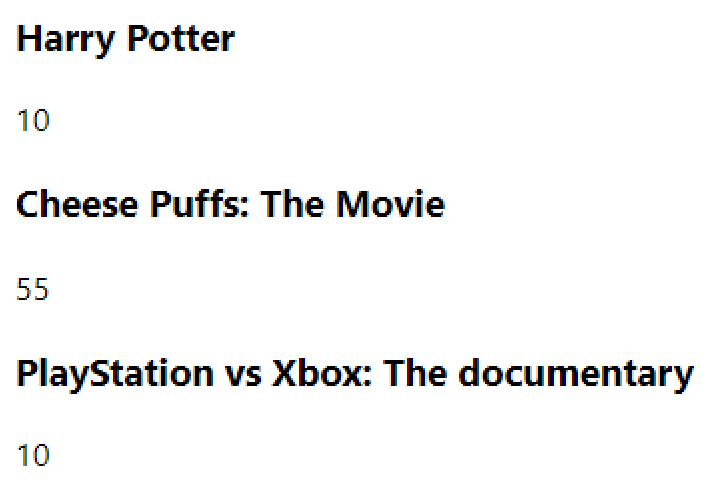
```
return (
<div>
    {
    NewContext.map((movies)=>(
        <React.Fragment>
        <h3>{movies.name}</h3>
        <p>{movies.Price}</p>
        </React.Fragment>
    ))

}
</div>
)
```

Code in MovieList.js

- `Lines`
  to dis

  to render multiple parent elements, we use `React.Fragment` as a parent element. `React.Fragment` represents an empty element in React.

Run the code, and this will be the output:



```
Harry Potter

10

Cheese Puffs: The Movie

55

PlayStation vs Xbox: The documentary

10
```

Code output

We have finally displayed our `movies` state without passing it down as props.

Let's say

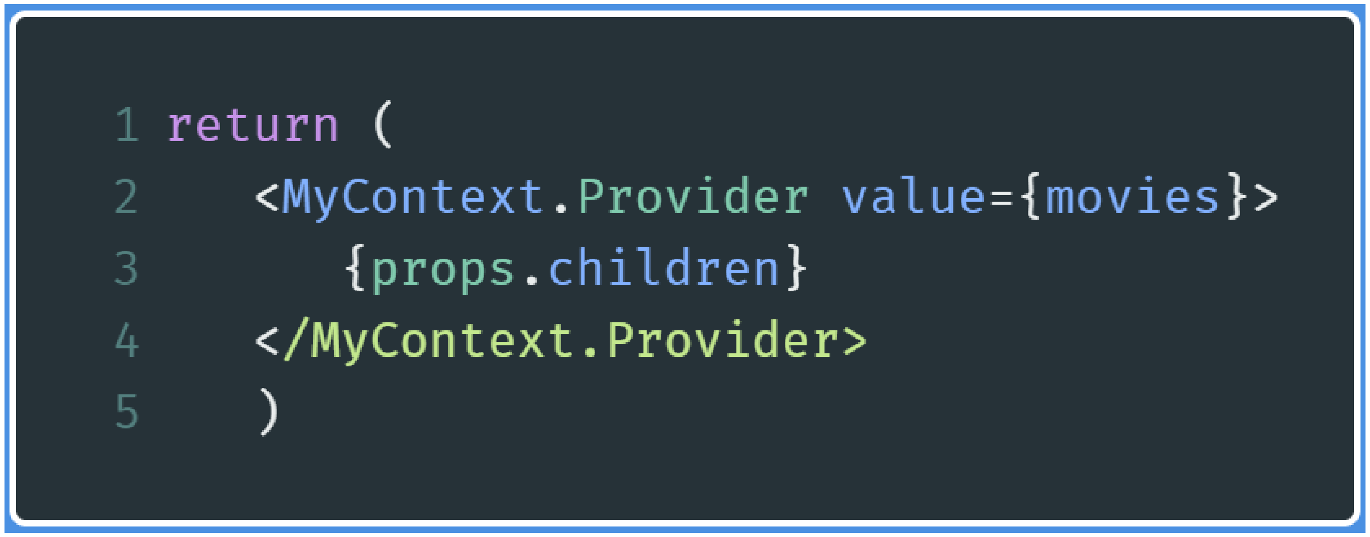
want to r

the world of React, this is easily possible. Let's move on to the next section.

Sharing Data and Functions With Context

MovieContext.js

In `MovieContext.js`, find the following `return` block:

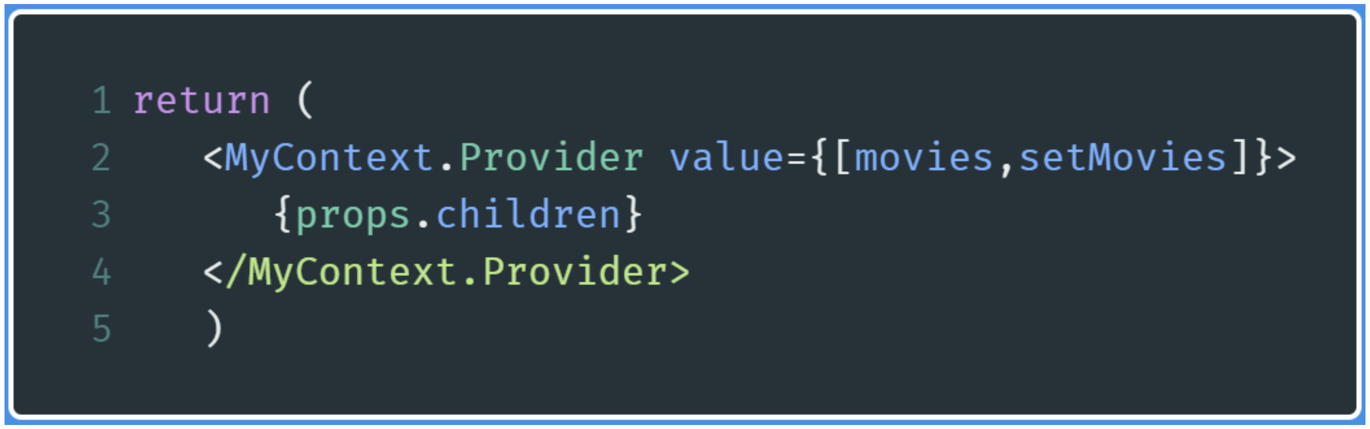


```
return (
   <MyContext.Provider value={movies}>
      {props.children}
   </MyContext.Provider>
   )
```

Code in MovieContext.js

Since we

`value` at



```
return (
    <MyContext.Provider value={[movies,setMovies]}>
       {props.children}
    </MyContext.Provider>
    )
```

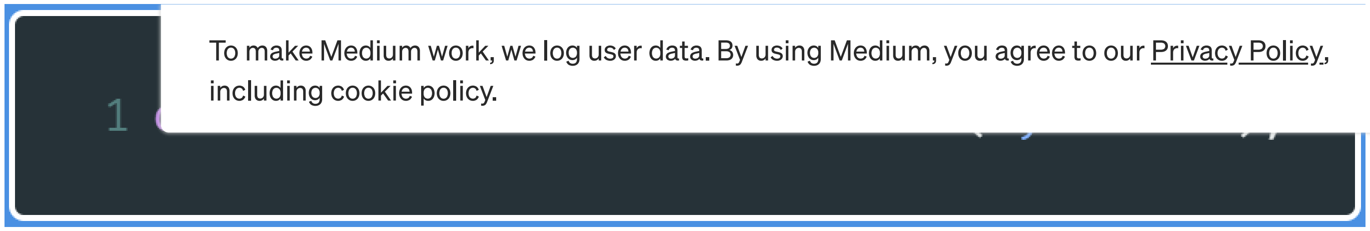
Code in MovieContext.js

This means that we have sent an array as accessible data. The first element is the `movies` array and the second element is the `setMovies` function, which will be used to set the values of the `movies` array. This step now means that we have shared data as well as the function using Context.

MovieList.js

We will make a minor amendment to this file.

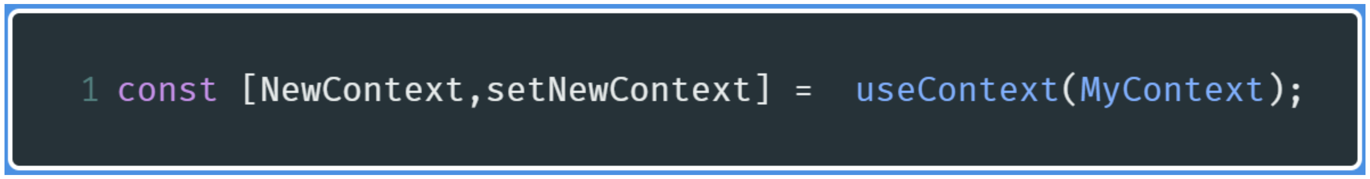
In the `MovieList` function, find this line of code:



To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

Code in MovieList.js

Then replace it with the following:



```
const [NewContext,setNewContext] =  useContext(MyContext);
```

Code in MovieList.js

As we sent an array as accessible data, we use array destructuring to process this array so that it can be used within `MovieList` .

If we run the code, we will get the same output as before.

Let's now move on to using the shared function so that we can add elements to the `movies` array.

ChangeContext.js

In this fil

This will

Create a new file called `ChangeContext.js` and write the following code in it:

```
import React,{useContext,useState} from  'react'
import {MyContext} from "./MovieContext";

function ChangeContext() {
    const [NewContext,setNewContext] =  useContext(MyContext);
    const [name,setName] = useState('');
    const [price,setPrice] = useState(0);
  const changeName=(event)=>{
    setName(event.target.value);
  }
  const changePrice= (event)=>{
      setPrice(event.target.value);
  }
    const changeValue = (event)=>{
        event.preventDefault();
        setNewContext(prevValue=> [...prevValue , {name : name, Price : price}]);
    }
    return (
        <div>
            <form>
                <input type='text' name='name' onChange={changeName}/>
                <input type='numer' name='price' onChange={changePrice}/>
                <button onClick={changeValue} />
            </form>
        </div>
26        )
27    }
28    expor
```

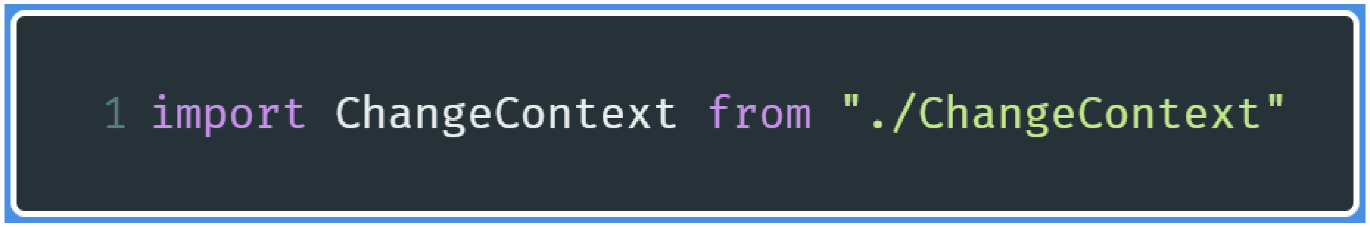
ChangeContext.js hosted with ♡ by GitHub view raw

- `Line 5` : Bring in the respective Context objects from the `MovieContext.js` file.
- `Lines 6-7` : We initialize the `name` and `price` hooks. These hooks will be assigned the respective values of their text fields.
- `Lines 8-13` : These functions will change the values of the `name` and `price` hooks. They will be executed whenever the user writes data in the text field(an `onChange` event has occurred). `event.target.value` is the value present in the text field.
- `Line 14` : Whenever the `Submit` button is clicked, then the `changeValue` function is invoked.
- `Line 15-17` : `event.preventDefault` prevents the page from reloading whenever the form is submitted. Since we cannot push elements to hooks, we use this method to add elements to the `movies` array.
- `Lines 18-27` : Here we are rendering a standard form component.

App.js

We will r

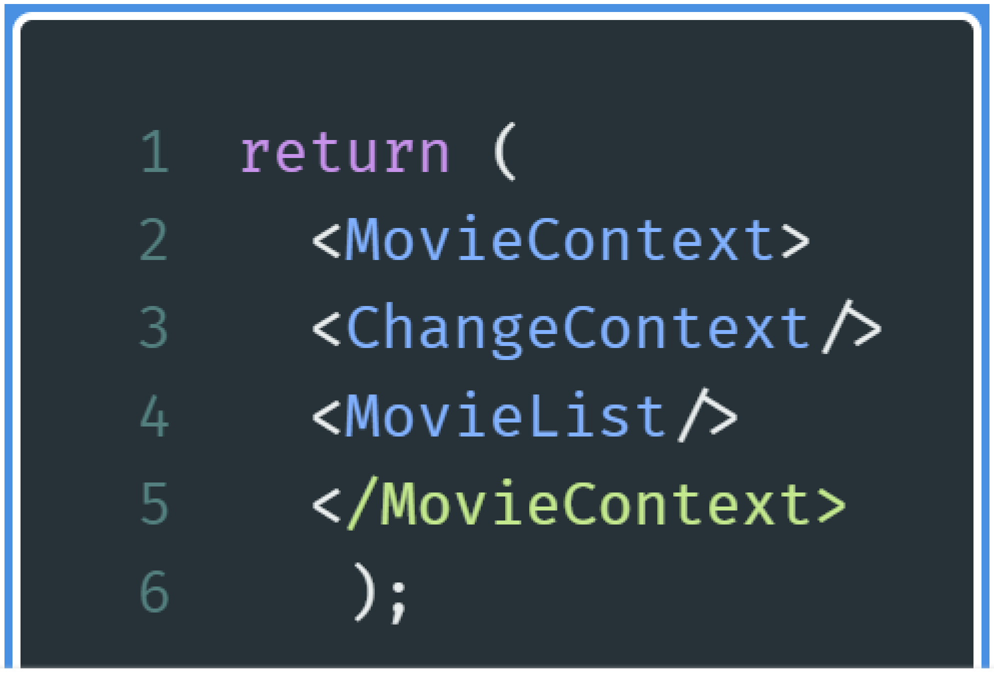
Import `ChangeContext` like so:



```
import ChangeContext from "./ChangeContext"
```

Code in ChangeContext.js

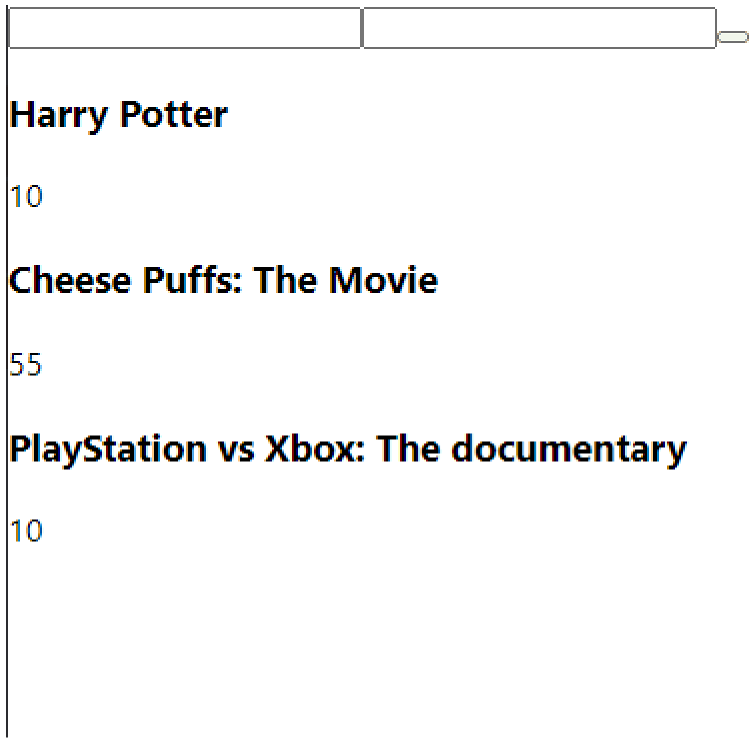
Now replace the `return` block with the following lines of code:



```
return (
    <MovieContext>
    <ChangeContext/>
    <MovieList/>
    </MovieContext>
     );
```

This means that now `ChangeContext` has access to the data shared by the `MovieContext` component.

Running the code will give you this output:



```
Harry Potter

10

Cheese Puffs: The Movie

55

PlayStation vs Xbox: The documentary

10
```

Code output

Notice th
the page

The navbar component

During the Introduction section, we wanted to display the number of elements in the `movies` array. This is now possible using Context.

Create a new file called `Navbar.js` and write the following code:

```
1  import React,{useContext} from "react";
2  import {MovieContext, MyContext} from "./MovieContext"
3  function NavBar(){
4      const [NewContext,] = useContext(MyContext)
5      return (
6          <p>Length : {NewContext.length} </p>
7      )
8  }
9  export default NavBar;
```

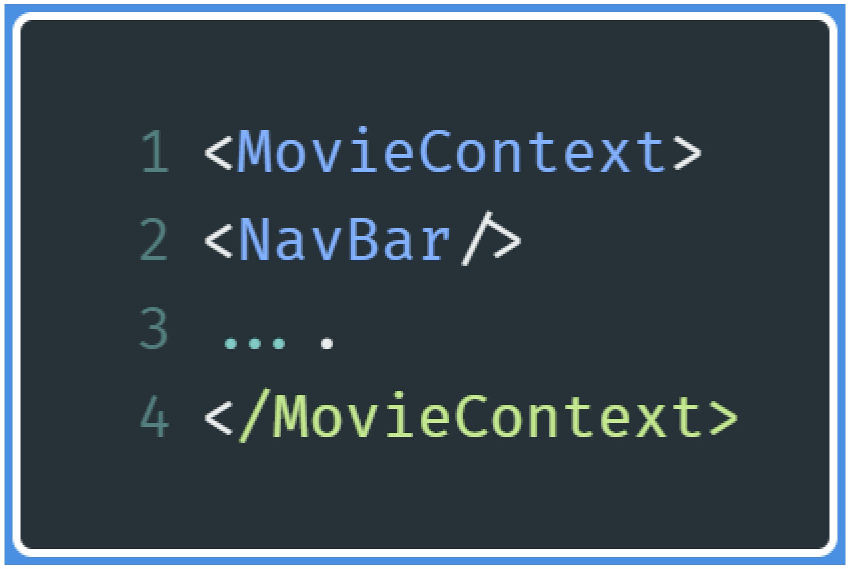
NavBarFull.js hosted with ♡ by GitHub view raw

- `Lines 1-2`: Our standard imports
- `Line 4` : Using the `MyContext` Context object
- `Line 6`: Display the number of elements in the `movies` array

Now go t

MovieCon

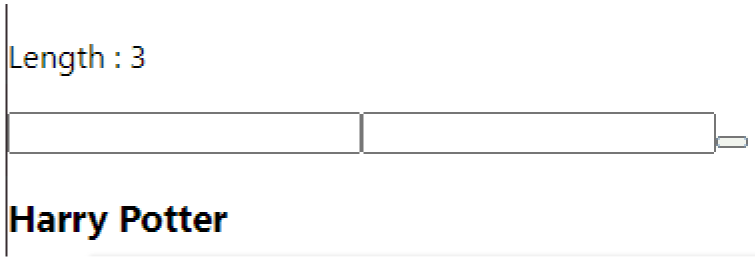
In `App.js` :



```
<MovieContext>
<NavBar/>
....
</MovieContext>
```

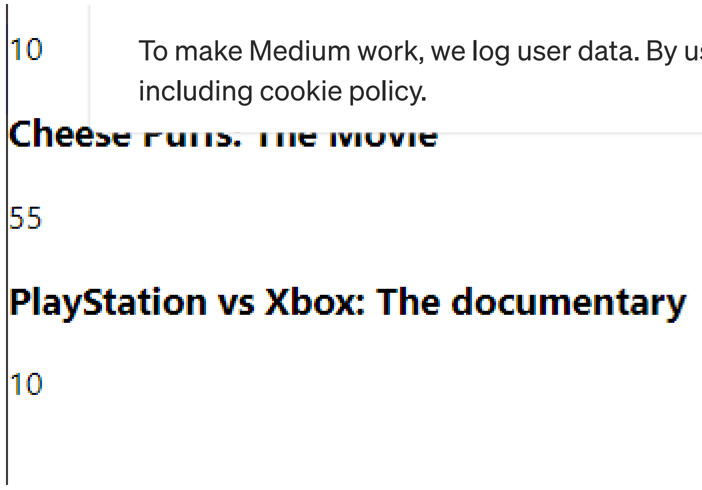
Code in App.js

This will be the output:



```
Length : 3
```

Harry Potter



```
10

Cheese Puffs: The Movie

55

PlayStation vs Xbox: The documentary

10
```

Code output

As you can see, whenever we add a new entry to this list, the element updates and the corresponding length of the `movies` array is shown.

Recap

Code

At the end of this tutorial, these are the files:

- `App.js`

```
impor
impor
impor
import MovieList from './MovieList';
import ChangeContext from "./ChangeContext"
function App() {
  return (
   <MovieContext>
   <ChangeContext/>
   <MovieList/>
   </MovieContext>
    );
}

export default App;
```

AppWithContextFull.js hosted with ♡ by GitHub view raw

- `MovieContext.js`

```
import React,{useState,createContext} from "react";
import MovieList from "./MovieList"

export const MyContext= createContext();
export function MovieContext(props) {
   const [movies,setMovies]= useState([
      {
         name : 'Harry Potter',
         Price : 10
      },
      {
         name : 'Cheese Puffs: The Movie'
```

```
        {
          name: 'PlayStation vs Xbox: The documentary',
          Price: 10
        }

    ]);

    return (
    <MyContext.Provider value={[movies,setMovies]}>
       {props.children}
    </MyContext.Provider>
    )
}
export default MovieContext
```

MovieContextFull.js hosted with ♡ by GitHub view raw

- `MoviesList.js`

```
import React,{useContext} from "react"
import {MyContext} from "./MovieContext"

function MovieList() {
 const [NewContext,setNewContext] =  useContext(MyContext);
return (
<div>
    {
    NewContext.map((movies)=>(
        <React.Fragment>
    ))

}
</div>
)
}
export default MovieList
```

MovieListFull.js hosted with ♡ by GitHub view raw

- `ChangeContext.js`

```
import React,{useContext,useState} from  'react'
import {MyContext} from "./MovieContext";

function ChangeContext() {
    const [NewContext,setNewContext] =  useContext(MyContext);
    const [name,setName] = useState('');
    const [price,setPrice] = useState(0);
  const changeName=(event)=>{
    setName(event.target.value);
  }
  const changePrice= (event)=>{
      setPrice(event.target.value);
  }
    const changeValue = (event)=>{
        event.preventDefault();
        setNewContext(prevValue=> [...prevValue , {name : name, Price : price}]);
    }
```

```
      r
```

To make Medium work, we log user data. By using Medium, you agree to our Privacy Policy, including cookie policy.

```
                <input type='text' name='name' onChange={changeName}/>
                <input type='numer' name='price' onChange={changePrice}/>
                <button onClick={changeValue} />
            </form>
        </div>
    )
}
export default ChangeContext;
```

ChangeContext.js hosted with ♡ by GitHub view raw

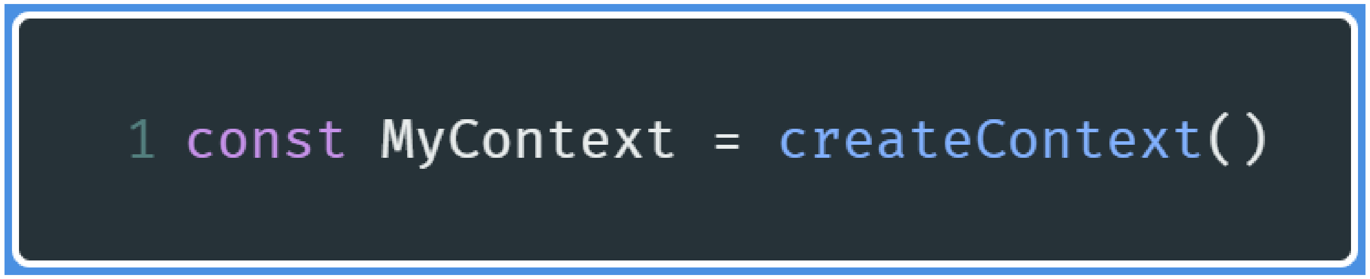
- `NavBar.js`

```
import React,{useContext} from "react";
import {MovieContext, MyContext} from "./MovieContext"
function NavBar(){
    const [NewContext,] = useContext(MyContext)
    return (
        <p>Length : {NewContext.length} </p>
    )
}
export default NavBar;
```

NavBarFull.js hosted with ♡ by GitHub view raw

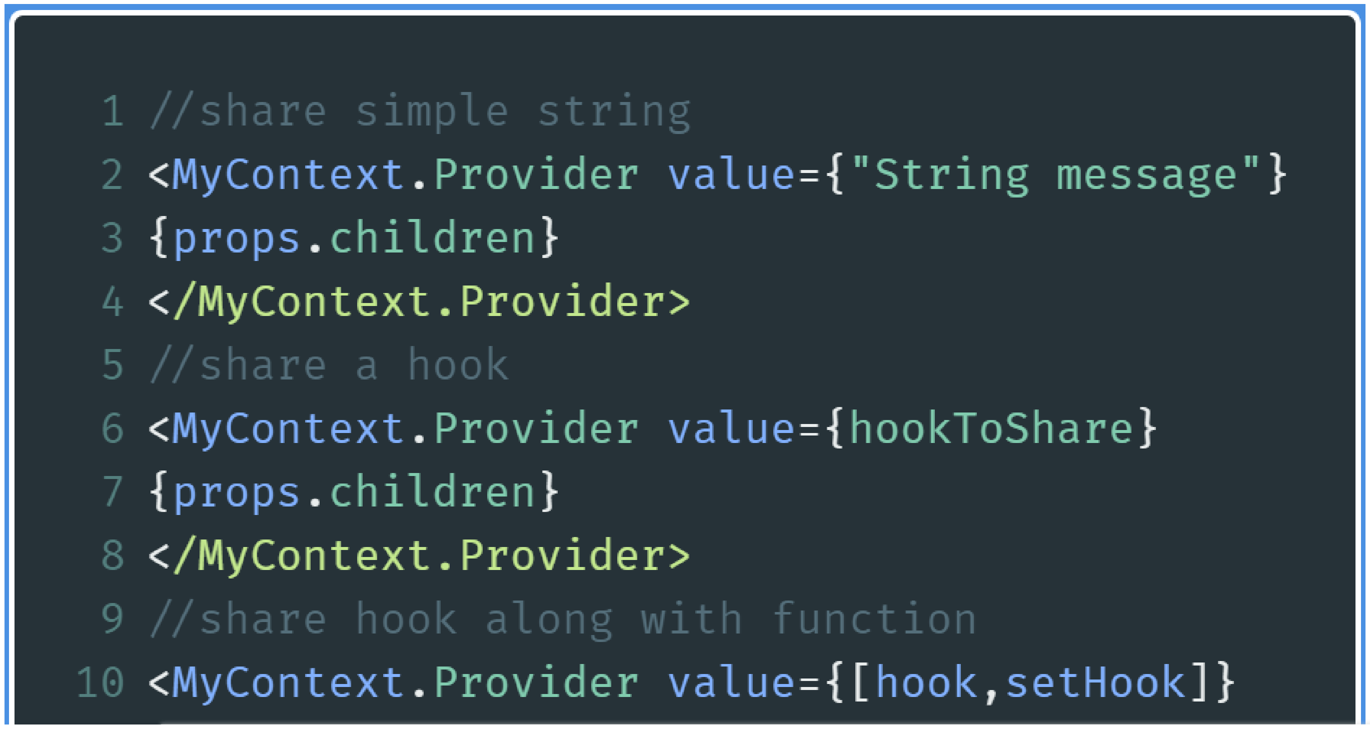
Additi

- To create Context:



```
const MyContext = createContext()
```

- To share data:



```
//share simple string
<MyContext.Provider value={"String message"}
{props.children}
</MyContext.Provider>
//share a hook
<MyContext.Provider value={hookToShare}
{props.children}
</MyContext.Provider>
//share hook along with function
<MyContext.Provider value={[hook,setHook]}
```

```
{
<
```

- To use Context:

```
//use hook and function
const [context,setContext]= useContext(MyContext)
```

Further Reading

- Here's How the New Context API Works — Wes Bos
- React Context — Dev Ed

Conclusion

Even tho

extremel

tutorial.

In case any of this is confusing, you are advised to play with the code and de-structure the above sample programs. Additionally, you can also use Google and further YouTube resources to go in-depth about Context.

Thank you so much for reading! Have a great day!

Next Post: Simple Web Scraping Project in Node.js
Previous Post: Login System In Node.Js

Thanks to Zack Shapiro.

Programming React Reactjs JavaScript Nodejs