# Changing Class based NewsMonkey components to Functional based | Complete React Course in Hindi #39

In the previous article, we have a short introduction to hooks. In today's video, we will be changing the class-based Components to functional-based components in the NewsMonkey application. Remember, You can even have a part of your application as a functional component and the other part as a class-based component.

## **Getting Started**

We have a total of five components which are Navbar, Spinner, "App.js", "News.js", "Newsltem.js". So let's begin changing these class-based components to the functional component:

#### Changing Navbar.js:

Firstly, we will begin refactoring the Navbar component of the NewsMonkey Application. We can easily change the class component of "Navbar.js" to a functional component, by removing the render function and returning the component in an arrow function as shown below:

#### Changing Spinner.js:

Similarly, we can make the spinner.js a function-based component. To do so we have to firstly remove the render function and after that return the spinner in an arrow function as shown below:

## Figure 1.2: Changing the Class component of Spinner.js Changing NewsItem.js:

Now, we are going to change NewsItem to a function-based component as it won't have any side effects in the NewsMonkey application. To do so, We have to first remove the render function and after that return, the component in an arrow function and remember to change 'this.props' with the props keyword.

Figure 1.3: Changing the Class component of NewsItem.js

Hence, we have successfully performed a basic refactor of the three components of the application.

The next two components, "News.js" and "App.js", are quite challenging to change to a function-based component, so without further ado let's dive straight into it:

### **Changing News.js**

We will first begin changing the 'proptypes' and 'default Props' to the function-based component. If you remember, we can use the prop types in the functional component as shown below:

```
News.defaultProps = {
    country: 'in',
    pageSize: 8,
    category: 'general',
}
News.propTypes = {
    country: PropTypes.string,
    pageSize: Proptypes.number,
    category: PropTypes.string,
}
```

Moreover, we will change the Export class component in an arrow function. After that, we will be using the 'usestate' hook in News.js to update the state and to set an initial value of the state. To do so firstly, Import the 'useState' hook in News.js. As a result, we can now utilize the 'use state' hook inside the function as shown below:

```
const News = (props) => {
   const [articles, setarticles] = useState([])
   const [loading, setloading] = useState(true)
   const [page, setPage] = useState(1)
   const [totalresults, setTotalResults] = useState(0)
```

**Explanation:** Since we have set the state with the help of the 'useState' hook, therefore we can remove the constructor from the News.js as it is not required anymore. Now, we can define a state by using the above state functions, and hence we will remove 'this' keyword, before declaring any state or variable. Make sure to define updateNews, fetch more and other functions by converting them into an arrow function.

Figure 1.4: Changing the News.js

#### Changing componentDidMount

To perform the task of componentDidMount, we will be using the useEffect function as shown below:

```
useEffect(() => {
    updateNews();
}, [])
```

**Result:** Hence, we have successfully refactored the application from a class-based component to a function-based component. Here's a look at the NewsMonkey application.

## **NewsMonkey - Top Entertainment Headlines**

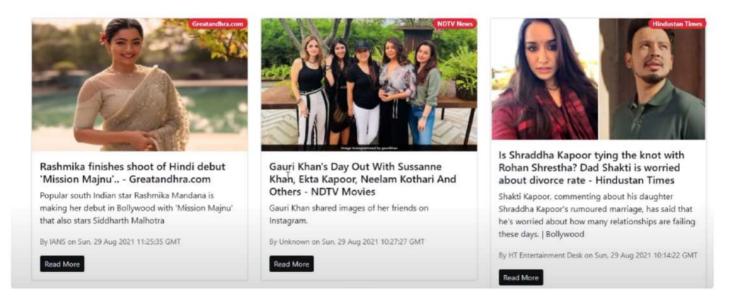


Figure 1.5: NewsMonkey Application