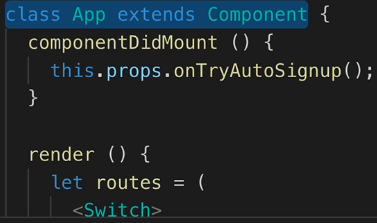
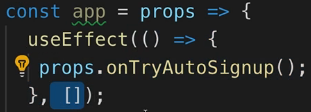
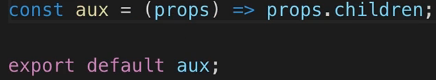
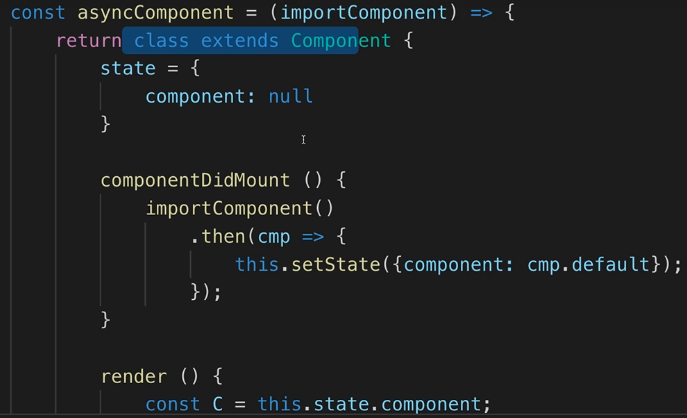
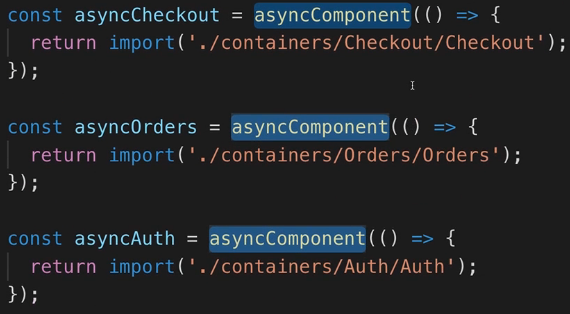
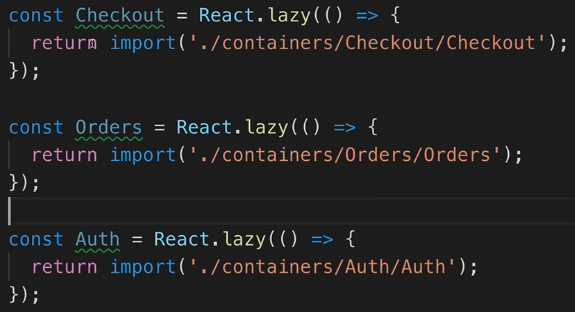
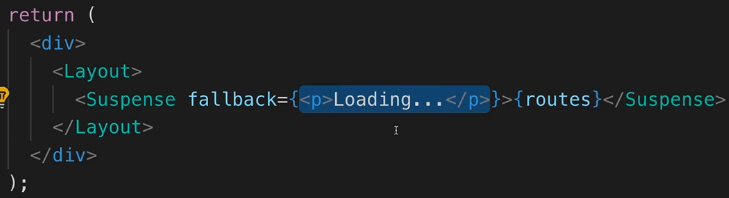
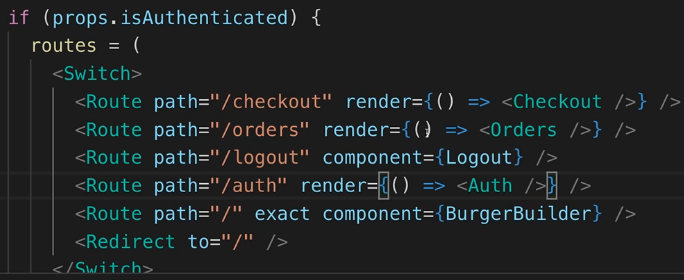
**Introduction**  
\* Let’s convert our project to use React Hooks and therefore only use Functional components.  
\* That’s a great practice for using Hooks.

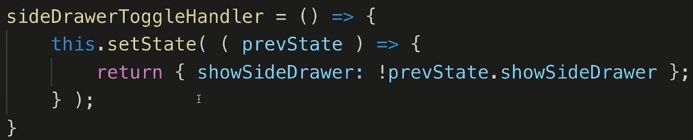
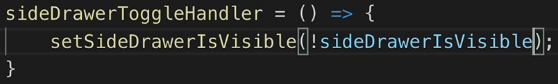
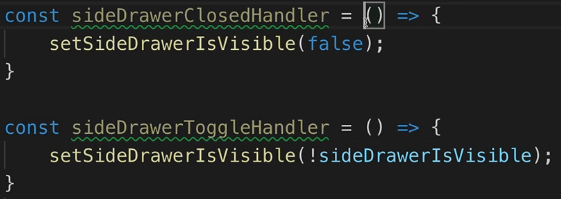
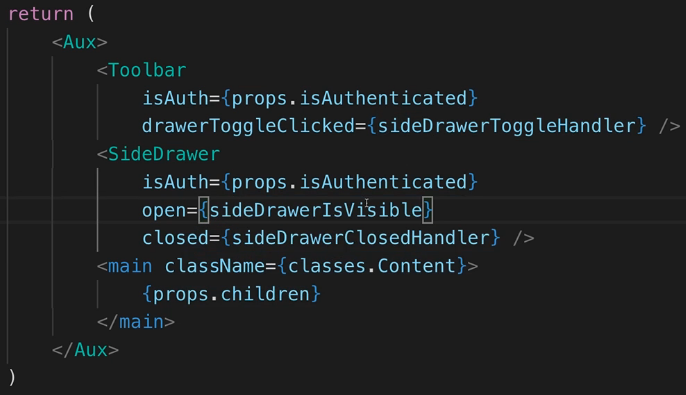
**Converting <App>**  
\* Our goal will be to convert all class-based components into Function components.  
\* **In** **index.js sets up the root of our application and in there we have no class-based component**.  
\* In App.js we want to convert it  
  
\* Let’s remove the `class extends Component`.  
\* Let’s remove the render() {}  
\* Let’s remove the componentDidMount()  
=> **We’re going to use useEffect() instead and we’ll pass it a function which will execute whenever this component re-renders**.  
\* Let’s use props instead of this.props.  
  
\* Now mapStateToProps() and mapDispatchToProps() can still be used, there’s nothing wrong with that, the only thing we need to change here is the name because I changed it from `App` to `app`.  
  
=> **So we can still connect Redux to our Functional component, nothing wrong with that**.  
\* Now everything works as before.

**Routing with React.lazy()**  
\* **In our `reducers` and `actions` files - so in the store file in general, we won’t have to do anything because all that Redux related stuff can stay as it is, it’s not related to components**.  
\* The `hoc` folder, that changes - we got 4 components there   
\* `**aux**` is a Functional component:  


\* `**asyncComponent**` which we use for Lazy Loading when we route is actually a Functional component that returns a Class-based component:  
  
=> We could try to convert this but actually it’s not worth the effort - instead we can delete it and instead use a feature we saw in the React Routing module:  
**React.lazy()**  
=> We do that in App.js, it’s a **native React feature** for **Lazy Loading**.  
\* So in App.js, we can replace:  
  
\* with:  
  
\* The React.lazy() is provided by React and essentially it does what we before had to do manually with asyncComponent - withor HOC (Higher Order Component).  
\* It was added with Reac 16.6.  
\* **We have to import** **Suspense** **component to make it work**.  
=> **This has to wrap any place where we want to use these lazy loaded components**.  
=> It takes the **fallback** property which defines **what to render while we’re waiting for one of these lazy loaded components to load**.  


\* So now we can use our lazily loaded components in the routes and for that, we just have to replace component:  
  
\* with render, there we get a function that executes which should simply return our JSX component:  
  
  
\* The BurgetBuilder isn’t lazily loaded so we don’t need to change it.  
\* Now let’s adjust the routes in the IF check, if the user is authenticated:  
  
\* Now with that, we’re using React.lazy() and Suspense and it all still works.  
\* **We’re using a modern React feature which is never a bad idea**.

**Converting the Layout Component**  
\* We have the Layout component and it’s a Class-based component.  


  
\* Here we set it based on the previous state, this is now super simple:  
  
  
\* Now remove the render() {}.  
\* Convert the methods to constants that hold functions:  
  
\* Change this.props to props.  
\* Change this.ourFunction to just ourFunction.  
\* Change state.something to just something.  
  
\* And lastly, export the function with lowercase name:  


**Converting withErrorHandler HOC**  
\*  
\*  
\*

00:00 / 00:00

Today: 0

18.8. => 10  
19.8. => 11  
20.8. => 12  
21.8. => 14 11 Alg-Java 3 React Hooks  
22.8. => 5  
23.8. => 16  
24.8. => 9  
25.8. => 13  
26.8. =>  
27.8. =>

**Resources**  
\*  
npm Package (Global State Hook)  
<https://www.npmjs.com/package/use-global-hook>  
\*