How to navigate imperatively

Here's how our site should flow. The user should be able to visit the LandingPage. They change the days as desired and tap/click on a FilmBrief. That sends them to the FilmDetails where they can tap/click on a showing time. That tap/click sends them to PickSeats for that showingId.

That would be slick, but none of that navigation works yet so let's wire them together.

From LandingPage to FilmDetails

1. Edit FilmBrief in your IDE.
2. import useNavigate from react-router-com at the top.
3. Call useNavigate:

```
const navigate = useNavigate();
```

4. Find the outermost <section> and add an onClick event to it.

The function called by onClick should navigate() to /film/filmId where the filmId is set to the current film.id. In other words, if you're looking at film 754, you should navigate("/film/754") or if you're looking at film 17, you should navigate("/film/17").

5. Go ahead and make that happen.
6. Run and test. You should be able to click on any FilmBrief and immediately navigate to FilmDetail for that film.

FilmDetail to PickSeats

Now let's navigate from ShowingTimes when the user clicks/taps on a showing time. They should go to PickSteats so they can choose which seat(s) they want to buy.

7. The steps you just went through apply for this next navigation also except that you'll go navigate("pickseats/showingId").
8. Go for it. Make the navigation happen. You'll know you've got it right when you can click on any showing in FilmDetail and be able to pick seats in the right theater for the right movie at the right showing time.

PickSeats to Checkout

For the PickSeats-to-Checkout linkage you have a choice. Since there are no route parameters to set, you can either convert the button to a <Link> and use a bunch of styling to make it look like a button, or you can useNavigate() to push the user there.

9. Make your choice and make that happen. There are no wrong answers.
10. Once you can navigate all the way from LandingPage to Checkout properly, you can be finished with this section.

Logging out

You might have noticed that we haven't done anything with the logging out function yet. Let's let the user log out. As of now the user can navigate to /logout and visit the <Logout> component, a perfect place to put the logic to un-authenticate!

11. Edit Logout.js and add this to the component:

```
import { useDispatch } from 'react-redux';
```

12. Then put this inside the function:

```
dispatch(actions.logout());
```

This is great. They'll certainly be logged out but they're seeing this confusing page. Let's send them off to another location once they're logged out. A <Navigate> component is a terrific candidate to do that.

13. In the rendered JSX, Add a <Navigate> component, sending the user to "/".
14. Run and test with the console open. Navigate to Login. Sign in. Look at the console. You should see an empty user object in state. Navigate anywhere you like, watching the user object. Then choose to visit /logout whether by the navigation menu or just entering the URL.
15. If you're pushed to LandingPage and the user becomes undefined, you're successful!