How to read route parameters

We've got our routes defined and can visit them. That's great. Did you notice that one of the routes has a parameter? The FilmDetails route which is visited by the user hitting "/film/1" or "/film/5" or "/film/100". Obviously the number in that URL has some significance; it is the film id.

The idea would be for the FilmDetails component to somehow grab that number from the URL and then use it locally.

1. Give that URL a try. Type in "/film/1". What film do you see? Now try "/film/2". What film now?

My guess is that you're going to see the same film details no matter what is in the URL because we've hardcoded the filmId.

2. Edit FilmDetails and find where we've hardcoded it. Remove that hardcode.
3. import the useParams hook from react-router-dom

Remember that when you call useParams, it returns an object of every key-value pair of route parameters. So it returns an object.

4. Call useParams and use any JavaScript technique to pull the "filmId" parameter from that object. This is assuming you called it "filmId" in the route definition back in App.js. You might want to double-check that if you can't see the parameter. They have to match.
5. Do this to get the film:

```
const state = store.getState();
if (state.films && state.films.length) {
  film = state.films.find(film => film.id === +filmId);
}
```

Note: we have to jump through a couple of hoops because the page begins rendering before the Redux store is loaded. In the code above, we're checking that state.films exists and that it is an array with content before we try to search through it for the film with that id.

6. Run and test. Use several filmIds. When you can see a different film for each filmId you provide in the URL, you got it right.

7. Bonus! See if you can read the films from props instead of state. If you can get that out, you will be able to remove the dependency on the store. This is a great demo of why that {…state} idiom can be so helpful.

Getting the showingId parameter for PickSeats

We have a similar situation with PickSeats. No matter what showing is in the URL, we show the user the same map because we're hardcoding a showingId.

8. Go through the few but necessary steps to read showingId from the URL.
   Hints:
   - Remember to import useParams
   - Calling useParams returns an object with the showingId you'll need
9. Run and test. When you can change the number in the URL and see different films, theaters, and showing times, you've got it right.