useEffect

Our project already has all of the useEffects that we need. But just so you can get a feel for how it works, do the following exercises to get some hands-on practice with useEffect.

Analyzing the code

1. Edit Menu.tsx. Examine the useEffect that is there. Note that it has an empty dependency array as its second argument. If you were pressed to do so, could you explain what this useEffect is doing? _____ Could you explain why we have an empty dependency array? _____

2. Now look at Order.tsx. It has a useEffect there also. What's happening in it? _____
   _____

3. Why does it NOT have an empty dependency array? What's that about?

componentDidUpdate

If we evolve our app far enough we'll encounter situations where useEffect is truly called for. Since that need isn't apparent yet, let's play around with useEffect so you can get an idea when it is needed and when not.

4. Edit App.tsx. Add this:

```
useEffect(() => console.log(`A ${Math.random()}`))
```

5. Run and test. Add something to the cart. Can you explain why you see your useEffect every time you hit the Add button? _____
   _____

6. Navigate to the Cart (Check out). Did you see the useEffect run? Why or why not? _____
   _____

7. In Cart, type something into a special request box. Whoa! The useEffect is running on every keystroke? Why? _____
   _____

componentDidMount

8. Edit App.tsx again. Add this:

```
useEffect(() => console.log(`B ${Math.random()}`), [])
```

9. Go through the steps from before. What do you see about this useEffect? _____
   _____

componentDidUnmount

10. Still in App.tsx, add this:

```
useEffect(() => () => console.log(`C ${Math.random()}`), [])
```

11. Run and test again. See if shows up. Did it? _____ Why or why not? _____
    _____

12. Alright, fine then! Put this in Menu.tsx:

```
useEffect(() => () => console.log(`D ${Math.random()}`), [])
```

13. Run and test again. Are you seeing it now? _____ Again, why or why not? _____
    _____

Another look at cleaning up

14. Edit Login.tsx. Add this:

```
const timer = setInterval(() => console.log(`E ${Math.random()}`), 2000);
```

This will start a process that will print a random number every two seconds.

15. Go ahead and run your site starting on the main page. Do you see the console.log yet? You shouldn't.

16. Now navigate to Login.

Now do you see it? You should. And every two seconds, you should see another random number. Cool, right?

17. Navigate away to Home. Look in the console. What the ...? You're still seeing the function run every two seconds!

18. Even better, navigate back to Login and away to any other menu choice. Now there are two showing up every two seconds. Keep navigating to Login and away. You'll see that each time you're creating another timer that never stops!

What can we do?

The answer is to register something that says "When we leave, stop the timer". We need a cleanup function.

19. Put this in Login:

```
useEffect(() => () => clearInterval(timer))
```

20. Run and test again, navigating to and away from Login.

This says that when the component is disposed of, run the returned function which stops the timer.

21. Last step ... Go back through and clean up everything we added in this lab. We won't need them in future labs.