Adding Nested Properties As Dependencies To useEffect

In the previous lecture, we used object destructuring to add object properties as dependencies to `useEffect()`.

```
const { someProperty } = someObject;
useEffect(() => {
  // code that only uses someProperty ...
}, [someProperty]);
```

This is a **very common pattern and approach**, which is why I typically use it and why I show it here (I will keep on using it throughout the course).

I just want to point out, that they **key thing is NOT that we use destructuring** but that we **pass specific properties instead of the entire object** as a dependency.

We could also write this code and it would **work in the same way**.

```
useEffect(() => {
  // code that only uses someProperty ...
}, [someObject.someProperty]);
```

This works just fine as well!

But you should **avoid** this code:

```
useEffect(() => {
  // code that only uses someProperty ...
}, [someObject]);
```

Why?

Because now the **effect function would re-run whenever ANY property** of `someObject` changes - not just the one property (`someProperty` in the above example) our effect might depend on.