Handling Deeply Nested Props

In this lesson, we'll discuss how to handle a component when nested props are used.

`React.memo` does a shallow comparison of props. By implication, if you have nested props objects, the comparison will fail.

To handle such cases, `React.memo` takes in a second argument, an `equalityCheck` function.

Here's a basic example:

```
import React, { memo } from 'react'
export default  memo (function MyComponent (props) {
    return ( <div>
        Hello World from {props.name.surname.short}
     </div>
    )
}, equalityCheck)
function equalityCheck(prevProps, nextProps) {
  // return perform equality check & return true || false
}
```

If the `equalityCheck` function returns `true`, no re-render will happen. This would mean that the current props and previous props are the same. If it returns `false`, then a re-render will occur.

If you're concerned about incurring extra performance hits from doing a deep comparison, you may use the **lodash** isEqual utility method.

```
import { isEqual } from 'lodash'
function equalityCheck(prevProps, nextProps) {
        return isEqual(prevProps, nextProps)
}
```

Let's conclude this in the next lesson.