Promise States & "finally"

You learned about the different promise states:

- **PENDING** => Promise is doing work, neither `then()` nor `catch()` executes at this moment
- **RESOLVED** => Promise is resolved => `then()` executes
- **REJECTED** => Promise was rejected => `catch()` executes

When you have another `then()` block after a `catch()` or `then()` block, the promise re-enters **PENDING** mode (keep in mind: `then()` and `catch()` always return a new promise - either not resolving to anything or resolving to what you `return` inside of `then()`). Only if there are no more `then()` blocks left, it enters a new, final mode: **SETTLED**.

Once **SETTLED**, you can use a special block - **`finally()`** - to do final cleanup work. `finally()` is reached no matter if you resolved or rejected before.

Here's an example:

```
somePromiseCreatingCode()
    .then(firstResult => {
        return 'done with first promise';
    })
    .catch(err => {
        // would handle any errors thrown before
        // implicitly returns a new promise - just like then()
    })
    .finally(() => {
        // the promise is settled now - finally() will NOT return a
new promise!
        // you can do final cleanup work here
    });
```

You don't have to add a `finally()` block (indeed we haven't in the lectures).