How to deal with asynchronous code in LWC?

You might have seen something like this before:

```
async connectedCallback() {
    await getAccounts({ country: this.country }).then(async accounts => {
        await findPartners({ accounts: accounts }).then(partners => {
            this.partners = partners;
        });
    });
}
```

This approach results in a callback nest that is difficult to understand and debug. On the next page, you will learn how to do it properly!

Use **async/await** when you want asynchronous code to behave like a synchronous one. JavaScript will pause function execution until the promise settles:

```
async connectedCallback() {
    try {
        let accounts = await getAccounts({ country: this.country });
        this.partners = await findPartners({ accounts: accounts });
    } catch (error) {
        console.error(error);
    }
}
```

If the data can be displayed later, or if you wish to perform another action only after the promise has been settled, utilize the **then/catch** block:

```
connectedCallback() {
    getAccounts({ country: this.country })
        .then(accounts => {
            this.accounts = accounts;
            this.hideSpinner();
        })
        .catch(error => {
            console.error(error);
        });
}
```