```
getUser(login)
  .then(function (user) {
    return getRights(user);
  })
  .then(function (rights) {
    return updateMenu(rights);
  })
  .catch(function (error) {
    console.log(error); // will be called if getUser or getRights fails
  })
```

You should seriously look into Promises, because they are going to be the new way to write APIs, and every library will use them. Even the standard ones: the new Fetch API does for example.

3.10. Arrow functions

One thing I like a lot in ES6 is the new arrow function syntax, using the 'fat arrow' operator (⇒). It is SO useful for callbacks and anonymous functions!

Let's take our previous example with promises:

```
getUser(login)
  .then(function (user) {
    return getRights(user); // getRights is returning a promise
  })
  .then(function (rights) {
    return updateMenu(rights);
  })
```

can be written with arrow functions like this:

```
getUser(login)
  .then(user => getRights(user))
  .then(rights => updateMenu(rights))
```

How cool is it? THAT cool!

Note that the return is also implicit if there is no block: no need to write `user ⇒ return getRights(user)`. But if we did have a block, we would need the explicit return: