Once the promise is resolved, the success callback (here simply logging the user on the console) will be called.

The cool part is that it flattens the code. For example, if your resolve callback is also returning a promise, you can write:

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

but more beautifully:

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

Another interesting thing is the error handling, as you can use one handler per promise, or one for all the chain.

One per promise:

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

One for the chain: