Use Cases

automated testing, building fake servers for development, making memorization proxies, establishing a logging layer, controlling client-side validation and handling access rights

When studying `Proxy.revocable`, we concluded that we could centralize control of data access via revocable proxies. All we need to do is pass the revocable proxy object to other consumer objects, and revoke their access once we want to make data inaccessible.

In **Exercise 1**, we will build a proxy that counts the number of times a method was called. Proxies can be used in automated testing. If you read the SinonJs documentation, you can see multiple use cases for proxies.

SinonJs spies, stubs, etc. were initially implemented and used in ES5. The reference is for the purpose that proxies are used in automated testing.

You can also build a fake server for development, which intercepts some API calls and answers them using static JSON files. For the API calls that are not being developed, the proxy simply passes the request through, and lets the client interact with the real server.

In **Exercise 2**, we will build a memorization proxy on the famous Fibonacci problem, and we will check out how many function calls we save with the lookup. If you extend this idea, you can also use a proxy to memorize the response of expensive API calls, and serve these calls without recalculating the results.

Exercise 3 highlights three use cases.

First, we may need to deal with JSON data that has a non-restricted structure. If we cannot make assumptions on the structure of a document, proxies come in handy.

Second, we can establish a logging layer including logs, warning messages, and errors using proxies.

Third, if we have control over when we log errors, we can also use proxies to control client-side validation.

If we combine the ideas of **Exercise 2** and **Exercise 3**, we can use proxies to restrict access to endpoints based on user credentials. This method does not save the server side implementation, but proxies give you a convenient way to handle access rights.

Now, let’s solve some exercises.