Introduction

introduction to proxy and some traps while handling the access of the target

A **proxy** is an object that wraps an object or a function and monitors access to the wrapped item, a.k.a. the **target**. We use proxies for the intention of blocking direct access to the target function or object.

The proxy object has some *traps* which handle access to the target. The traps are the same as the methods used in the Reflect API. Therefore, this section assumes that you are familiar with the Reflect API, as you will use the same calls. The following traps are available:

- `apply`
- `construct`
- `defineProperty`
- `deleteProperty`
- `get`
- `getOwnPropertyDescriptor`
- `getPrototypeOf`
- `has`
- `isExtensible`
- `ownKeys`
- `preventExtensions`
- `set`
- `setPrototypeOf`

Once a trap of a proxy object is executed, we can run any code, even without accessing the target. The proxy decides if it wants to provide you access to the target, or handle the request on its own.

The available traps allow you to monitor almost any use case. Some behavior cannot be trapped though.

cannot be trapped though.

We cannot tell if our object is compared to another value, or wrapped by another object. We cannot tell if our object is an operand of an operator. In reality, the absence of these use cases is hardly problematic.

Now, let's learn how to define these proxies.