ToiletPaper #133

JavaScript Proxy Object

Author: Santo Pfingsten / Software Engineer / Office Leipzig

Problem

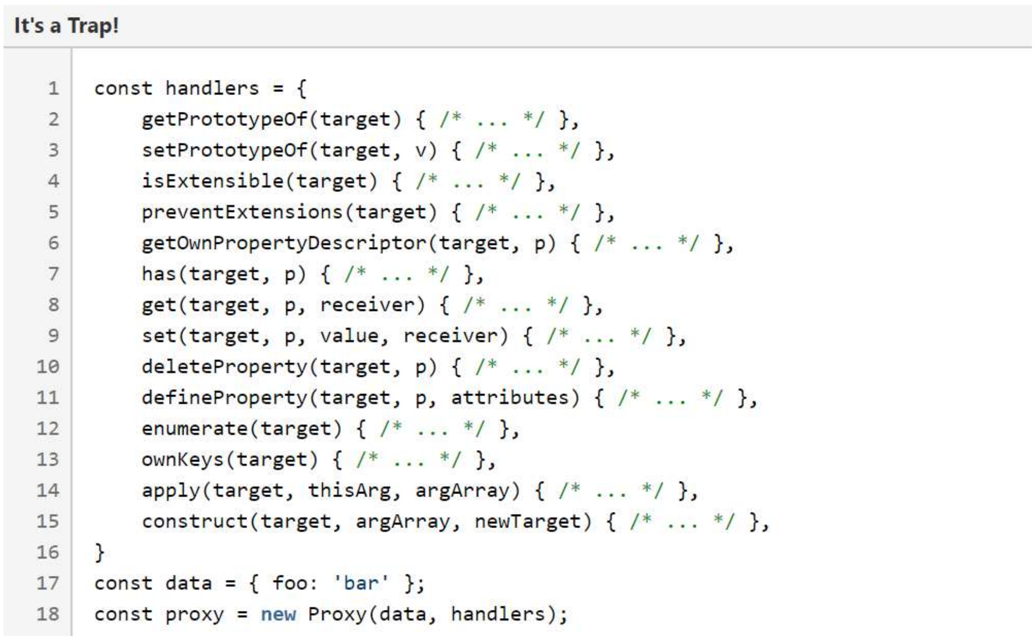
You want to provide an object and you don't know beforehand which operations will be applied on it? Or maybe you know which operations, but to write every single operation manually would be way too time consuming? By operations I mean e.g. reading, writing, deleting, instantiating (new), calling, enumerating, examining, etc.

Solution

With the JavaScript (ES6) Proxy object, you can define so-called handlers (sometimes called traps) for all operations that can be performed on an object. In these handlers, you can intercept the actual request and implement your own logic. As you can see, a JavaScript proxy object has nothing to do with a network proxy.

Example

A target and a handler object are passed to the proxy constructor. The functions in the handler object receive the target as the first parameter and possibly further parameters depending on the type. So, we have a strict separation between data and logic. Each of the following functions is optional:


It's a Trap!

```javascript
const handlers = {
    getPrototypeOf(target) { /* ... */ },
    setPrototypeOf(target, v) { /* ... */ },
    isExtensible(target) { /* ... */ },
    preventExtensions(target) { /* ... */ },
    getOwnPropertyDescriptor(target, p) { /* ... */ },
    has(target, p) { /* ... */ },
    get(target, p, receiver) { /* ... */ },
    set(target, p, value, receiver) { /* ... */ },
    deleteProperty(target, p) { /* ... */ },
    defineProperty(target, p, attributes) { /* ... */ },
    enumerate(target) { /* ... */ },
    ownKeys(target) { /* ... */ },
    apply(target, thisArg, argArray) { /* ... */ },
    construct(target, argArray, newTarget) { /* ... */ },
}
const data = { foo: 'bar' };
const proxy = new Proxy(data, handlers);
```

Possible applications:

- Creating mocks in the field of unit testing
- Uncomplicated access to databases, or translating from one API to another
- Performing an operation on multiple objects (distribution principle)
- Logging, caching, data validation, read-only access, side effects, ...

Further Aspects

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Proxy
- https://www.digitalocean.com/community/tutorials/js-proxy-traps
- https://ponyfoo.com/articles/es6-proxy-traps-in-depth
- A private project where I used proxies: https://github.com/Lusito/mockzilla
- More examples: LDFlex, Immer, Alpine.js, exploringjs.com
- Apart from IE, Opera Mini, and Baidu all browsers support the proxy object
- Alternatively, there is a polyfill, which does not support all traps: https://www.npmjs.com/package/proxy-polyfill