ToiletPaper #133

JavaScript Proxy Object

Autor: Santo Pfingsten / Software Engineer / Standort Leipzig

✗ Problem

Ihr wollt ein Objekt bereitstellen, bei dem ihr vorher nicht wisst, welche Operationen darauf angewandt werden? Oder ihr wisst vielleicht welche Operationen, aber jede einzelne Operation manuell zu schreiben wäre viel zu zeitintensiv? Mit Operationen meine ich z. B. Lesen, Schreiben, Löschen, Erzeugen (new), Aufrufen, Enumerieren, Untersuchen, etc.

✓ Lösung

Mit dem JavaScript (ES6) Proxy Objekt kann man für alle Operationen, die man auf einem Objekt ausführen kann, sogenannte Handler definieren (gelegentlich als Traps bezeichnet). In diesen Handlern kann man den eigentlichen Abruf abfangen und eigene Logik implementieren. Ein JavaScript Proxy-Objekt hat also nichts mit einem Netzwerk-Proxy zu tun.

➔ Beispiel

Dem Proxy Konstruktor übergibt man ein Target- und ein Handler-Objekt. Die Funktionen im Handler-Objekt erhalten das Target als ersten Parameter und ggf. weitere Parameter je nach Typ. Es herrscht also eine strikte Trennung zwischen Daten und Logik. Jede der folgenden Funktionen ist optional:

It's a Trap!

```
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);
```

Mögliche Anwendungsfälle:

- Erstellen von Mocks im Bereich von Unit-Testing
- Unkomplizierter Zugriff auf Datenbanken, bzw. das Übersetzen von einer API auf eine andere.
- Eine Operation auf mehreren Objekten ausführen (Verteiler-Prinzip)
- Logging, Caching, Datenvalidierung, read-only Zugriff, Seiteneffekte, ...

+ Weiterführende Aspekte

- 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
- Ein privates Projekt, bei dem ich Proxys eingesetzt habe: https://github.com/Lusito/mockzilla
- Weitere Beispiele: LDFlex, Immer, Alpine.js, exploringjs.com
- Abseits von IE, Opera Mini und Baidu unterstützen alle Browser das Proxy-Objekt
- Alternativ gibt es ein Polyfill, welches allerdings nicht alle Traps unterstützt: https://www.npmjs.com/package/proxy-polyfill