Chapter 74: Proxy

Parameter	Details
target	The target object, actions on this object (getting, setting, etc...) will be routed through the handler
handler	An object that can define "traps" for intercepting actions on the target object (getting, setting, etc...)

A Proxy in JavaScript can be used to modify fundamental operations on objects. Proxies were introduced in ES6. A Proxy on an object is itself an object, that has *traps*. Traps may be triggered when operations are performed on the Proxy. This includes property lookup, function calling, modifying properties, adding properties, et cetera. When no applicable trap is defined, the operation is performed on the proxied object as if there was no Proxy.

Section 74.1: Proxying property lookup

To influence property lookup, the `get` handler must be used.

In this example, we modify property lookup so that not only the value, but also the type of that value is returned. We use Reflect to ease this.

```
let handler = {
    get(target, property) {
        if (!Reflect.has(target, property)) {
            return {
                value: undefined,
                type: 'undefined'
            };
        }
        let value = Reflect.get(target, property);
        return {
            value: value,
            type: typeof value
        };
    }
};

let proxied = new Proxy({foo: 'bar'}, handler);
console.log(proxied.foo); // logs `Object {value: "bar", type: "string"}`
```

Section 74.2: Very simple proxy (using the set trap)

This proxy simply appends the string `" went through proxy"` to every string property set on the target `object`.

```
let object  = {};

let handler = {
    set(target, prop, value){ // Note that ES6 object syntax is used
        if('string' === typeof value){
            target[prop] = value + " went through proxy";
        }
    }
};

let proxied = new Proxy(object, handler);

proxied.example = "ExampleValue";

console.log(object);
// logs: { example: "ExampleValue went through proxy" }
```