Revocable Proxies

Proxy.revocable and its behavior

We can also create revocable proxies using `Proxy.revocable`. This is useful when we pass proxies to other objects, but we want to keep a centralized control of when we want to shut down our proxy.

```
let payload = {
    website: 'zsoltnagy.eu',
    article: 'Proxies in Practice',
    viewCount: 15496
}

let revocable = Proxy.revocable( payload, {
   get: function( ...args ) {
        console.log( 'Proxy' );
        return Reflect.get( ...args );
   }
});

let proxy = revocable.proxy;

console.log(proxy.website);
//> Proxy
//> "zsoltnagy.eu"

revocable.revoke();

proxy.website;
//> Uncaught TypeError: Cannot perform 'get' on a proxy that
//> has been revoked
//>    at <anonymous>:3:6
```

Once we revoke the proxy, it throws an error when we try using it.

As both the `revoke` method and `proxy` are accessible inside `revocable`, we can use the ES6 shorthand notation for objects to shorten our code:

```
// ...
// Create a revocable proxy
let {proxy, revoke} = Proxy.revocable( payload, {
   get: function( ...args ) {

        console.log( 'Proxy' );
        return Reflect.get( ...args );
   }
});

// Revoke the proxy
revoke();
```

Now, let’s talk about the various use cases of proxies in the next lesson.