index

September 20, 2024

1 Spread+getter=:finnadie:

1.1 Abstract

A common pattern to "extends" object and or clone them is to use the spread operator, this can lead to this kind of bizarre behavior when used with getter:

[7]:
```
let counter = 0;
const original = {
    get counter() {
        return counter++;
    }
};
const copy = {...original};
console.log('original.counter, expecting 0, but found:', original.counter);
// Okay, if original.counter is 1, if I call copy.counter, it should be 2,␣
 ↪right?
console.log('copy.counter, expecting 2, but found:', copy.counter);
```

```
original.counter, expecting 0, but found: 1
copy.counter, expecting 2, but found: 0
```

1.2 Problem

While `getter` looks like function, they are a Property without `value` and instead with a `get` implementation. Most read operation will use the `[[Get]]` internal method which will execute the `get` and use its return value as `value`. This means that, when copy an object containing `getter` properties, those are evaluated, and the result of the evaluation is then assigned in the spread object.

1.3 Solution

Instead of copying value, we need to copy property descriptors. Property Descriptor can be considered a "subset" of Property, minus the `name` ECMAScript comes with various methods to manipulate those: * `Object.getOwnPropertyDescriptor`: Returns the Property Descriptor of a given key * `Object.getOwnPropertyDescriptors`: Returns an object akin to a map between the `name` of all Properties and their respective Property Descriptor of a given object. * `Object.defineProperty`: Set a Property Descriptor on a given key of a given object. See it as the 'set' counterpart of `Object.getOwnPropertyDescriptor` * `Object.defineProperties`: Assign all Property Descriptor of a given object, on another object, using the same keys.

With the 2nd & 4th methods, we can craft a “spread for properties” like so:

[1]:
```
let counter = 0;
const original = {
    get counter() {
        return counter++;
    }
};
const copy = Object.defineProperties({}, Object.
 ↪getOwnPropertyDescriptors(original));
console.log('original.counter, expecting 0, found:', original.counter);
console.log('copy.counter, expecting 1, found:', copy.counter);
console.log('original.counter, expecting 2, found:', original.counter);
```

```
original.counter, expecting 0, found: 0
copy.counter, expecting 1, found: 1
original.counter, expecting 2, found: 2
```

[ ]:
```

```