Proxies vs. ES5 Getter/Setter

Initially, it can be a bit unclear as to why proxies are all that beneficial when there are already getter and setter methods provided in ES5. With ES5's getter and setter methods, you need to know *before hand* the properties that are going to be get/set:

```
var obj = {
    _age: 5,
    _height: 4,
    get age() {
        console.log(`getting the "age" property`);
        console.log(this._age);
    },
    get height() {
        console.log(`getting the "height" property`);
        console.log(this._height);
    }
};
```

With the code above, notice that we have to set `get age()` and `get height()` when initializing the object. So when we call the code below, we'll get the following results:

```
obj.age; // logs 'getting the "age" property' & 5
obj.height; // logs 'getting the "height" property' & 4
```

But look what happens when we now *add a new property* to the object:

```
obj.weight = 120; // set a new property on the object
obj.weight; // logs just 120
```

Notice that a `getting the "weight" property` message wasn't displayed like the `age` and `height` properties produced.

With ES6 Proxies, we *do not need to know the properties beforehand*:

```
const proxyObj = new Proxy({age: 5, height: 4}, {
    get(targetObj, property) {
        console.log(`getting the ${property} property`);
        console.log(targetObj[property]);
    }
});

proxyObj.age; // logs 'getting the age property' & 5
proxyObj.height; // logs 'getting the height property' & 4
```

All well and good, just like the ES5 code, but look what happens when we add a new property:

```
proxyObj.weight = 120; // set a new property on the object
proxyObj.weight; // logs 'getting the weight property' & 120
```

See that?!? A `weight` property was added to the proxy object, and when it was later retrieved, it displayed a log message!

So some functionality of proxy objects may seem similar to existing ES5 getter/setter methods. But with proxies, you do not need to initialize the object with getters/setters for each property when the object is initialized.

NEXT