Hack 1, Part 2: The Making of Objects

In which we use a wizard and goblins to explain objects, properties and references

Stolen Braces

Within the Great Console, lived an wizard who lived in the shadows and envied `;`.

One day, the wizard noticed that `;` had left his blueprints for `{` and `}` and his variable boxes out in the garden.

Wizard crept into the garden and copied the blueprint for the curley braces and the variable boxes, with the intent to *remake them*.

The wizard took the braces and the *remade them* along with the variable nets, writing upon the Great Console and then waving his way he yelled "Objecto Literallous":

```
> var goblin = {
    name: 'ralph',
    color: 'blue'
  };
  goblin;

Object {name: "goblin", color: "blue"}
```

And from the console sprung an object goblin. Object goblins had access to their own room of vaults, each vault with a name that matched a key upon the key chain carried by the goblin.

By giving the goblin a name of the key, the goblin would unlock the corresponding vault and copy the value within to a piece of paper which would be handed back to the asker:

```
> var name = goblin.name;
  var color = goblin.color;
  console.log(name, color);
"goblin" "blue"
```

The goblin could also be asked to replace the value in a vault with a new value:

```
> goblin.color = 'red';
  console.log(goblin.color);
"red"
```

One day the wizard, in a foul mood, assigned one goblin to the property of another:

```
> var fullName = {
    first: 'ralph',
    last: 'shortbottom'
  };

  var goblin = {
    name: fullName,
    color: 'green'
  };
```

To the wizard's suprise, both the `fullName` variable *and* the `name` property returned the same exact goblin:

```
> fullName;
Object {first: "ralph", last: "shortbottom"}
> goblin.name;
Object {first: "ralph", last: "shortbottom"}
> fullName === goblin.name;
true
```

How could this be? Why was the first goblin not locked into the second goblin's vault?

To the wizard's repeated the operation and watched closely.

```
> var one = { name: 'one' };
  var two = { name: 'two', other: one };
```

This time the wizard noticed that when goblin `one` was sent to the `other` vault, goblin two wrote down the goblin's name on a special pink slip of paper and put it into the vault.

When the wizard asked for the value:

```
> two.other;
Object {name: "one"}
```

Goblin `two` copied the value from the vault as had been done for the other values, but then the goblin went to goblin break room and located the other goblin.

The wizard realized it was even possible to chain requests and the goblins would eventually return the right value:

```
> two.other.name;
"one"
```

The wizard was not particularly nice to the goblins. Often the wizard would just ask for keys that didn't exist.

```
> goblin.haha;
undefined
```

One day the wizard, on a whim, made up the word `constructor` and asked the goblin for that key:

```
> goblin.constructor;
function Object() { [native code] }
```

WAT?????

The wizard demanded to see the goblin's key chain. While there was no `constructor` key, there was an odd key with the name `__proto__` (pronounced dunder-proto). He demanded the goblin return the value for `__proto__`.

The goblin mumbled something about his "prototype" and wandered off to fetch the value, first copying the name from the pink slip, then coming back from the break room with a ancient goblin.

```
> goblin.__proto__;
Object {}
```

The wizard quickly asked several other goblins for their `__proto__` property, and each one return the same exact goblin!

```
> var one = { name: 'gobs' };
  var two = { name: 'bobblin' };
  one.__proto__ === two.__proto;
true
```

The wizard asked the old goblin if we was always the goblin returned by `__proto__`. The old goblin said that using the "Objecto Literallous" spell that would always be true.

But then the old goblin showed the wizard how to use `Object.create` to control what goblin would the `__proto__` of another goblin:

```
> var one = { foo: 'foo' };
  var two = Object.create(one);
  two.bar = "bar";
  var three = Object.create(two);
  three.qux = 'qux';
  console.log(three.foo, three.bar, three.qux);
foo bar qux
```

Asking a goblin for a key that was in their vaults, or any of their prototype's vaults, would eventually find the correct value!

The wizard also learned that a goblin could be created who had no prototype:

```
> var goblin = Object.create(null);
  goblin.__proto__;
undefined
```

Review

- Object are sets of properties
- Objects are not stored as values, but as a reference to be able to find the object
- Property Layering
  - Look in child, if not found look in parent (prototype)
  - Also applies to nested function scopes

END PART 2

Exercises

Write out the following object literals:

1. A person that has a name of "Sally" a birthday of May 1, 1989 and a favorite color of blue
2. An address that has street, city, state and zip
3. A store named "My Store" that has the address from #2, and owner who is #1