A Drip of JavaScript

Object Equality in JavaScript

Originally published in the A Drip of JavaScript newsletter.

Equality is one of the most initially confusing aspects of JavaScript. The behavior of `==` versus `===`, the order of type coercions, etc. all serve to complicate the subject. Today we'll be looking at another facet: how object equality works.

You might suppose that if two objects have the same properties and all of their properties have the same value, they would be considered equal. Let's take a look and see what happens.

```
var jangoFett = {
    occupation: "Bounty Hunter",
    genetics: "superb"
};

var bobaFett = {
    occupation: "Bounty Hunter",
    genetics: "superb"
};

// Outputs: false
console.log(bobaFett === jangoFett);
```

The properties of `bobaFett` and `jangoFett` are identical, yet the objects themselves aren't considered equal. Perhaps it's because we're using triple equals? Let's test that theory.

```
// Outputs: false
console.log(bobaFett == jangoFett);
```

The reason for this is that internally JavaScript actually has two different approaches for testing equality. Primitives like strings and numbers are compared by their value, while objects like arrays, dates, and plain objects are compared by their reference. That comparison by reference basically checks to see if the objects given refer to the same location in memory. Here is an example of how that works.

```
var jangoFett = {
    occupation: "Bounty Hunter",
    genetics: "superb"
};

var bobaFett = {
    occupation: "Bounty Hunter",
    genetics: "superb"
};

var callMeJango = jangoFett;

// Outputs: false
console.log(bobaFett === jangoFett);

// Outputs: true
console.log(callMeJango === jangoFett);
```

On the one hand, the variables `jangoFett` and `bobaFett` refer to two objects with identical properties, but they are each distinct instances. On the other hand `jangoFett` and `callMeJango` both refer to the same instance.

Because of this, when you are trying to check for object equality you need to have a clear idea about what sort of equality you are interested in. Do you want to check that these two things are the exact same instance? Then you can use JavaScript's built-in equality operators. Or do you want to check that these two objects are the "same value?" If that's the case, then you'll need to do a bit more work.

Here is a very basic approach to checking an object's "value equality".

```
function isEquivalent(a, b) {
    // Create arrays of property names
    var aProps = Object.getOwnPropertyNames(a);
    var bProps = Object.getOwnPropertyNames(b);

    // If number of properties is different,
    // objects are not equivalent
    if (aProps.length != bProps.length) {
        return false;
    }

    for (var i = 0; i < aProps.length; i++) {
        var propName = aProps[i];

        // If values of same property are not equal,
        // objects are not equivalent
        if (a[propName] !== b[propName]) {
            return false;
        }
    }

    // If we made it this far, objects
    // are considered equivalent
    return true;
}

// Outputs: true
console.log(isEquivalent(bobaFett, jangoFett));
```

As you can see, to check the objects' "value equality" we essentially have to iterate over every property in the objects to see whether they are equal. And while this simple implementation works for our example, there are a lot of cases that it doesn't handle. For instance:

- What if one of the property values is itself an object?
- What if one of the property values is `NaN` (the only value in JavaScript that is not equal to itself?)
- What if `a` has a property with value `undefined`, while `b` doesn't have this property (which thus evaluates to `undefined`?)

For a robust method of checking objects' "value equality" it is better to rely on a well-tested library that covers the various edge cases. Both Underscore and Lo-Dash have implementations named `_.isEqual` which handle deep object comparisons well. You can use them like this:

```
// Outputs: true
console.log(_.isEqual(bobaFett, jangoFett));
```

I hope that this drip of JavaScript has helped you get a better grasp of how object equality works.