A JavaScript Taco

Objectives

- Learn about objects in JavaScript
- Learn about properties and methods
- Learn what `this` is

Objects - storing information in named properties

Oftentimes when we're dealing with data, it's not enough to just know one property or value, there is instead a lot of related information we're concerned with. If for instance we wanted to plan out our perfect taco, we might imagine all of the individual ingredients we'd want to add onto it! The tortilla or shell, the protein, maybe cheese and lots of toppings! We can express all of this as separate variables in JavaScript!

```
var tortilla = "soft corn tortilla";
var protein = "tinga chicken";
var cheese = "cotija cheese";
var toppings = ["lettuce", "pico de gallo", "guacamole"];
```

This works just fine for one taco, but what if we want to make a second taco with different ingredients? Should we start adding numbers to the end of our variables? Something like `tortilla2`, `protein2`, `cheese2`, and `toppings2`? Instead, we could create separate tacos and allow them both to have properites that can share a common name by using Objects! An object allows us to combine multiple values like an array, but unlike how an array uses consecutive numbers for the locations an object allows us to use any word we would like. We'll commonly refer to these as `key value pairs`. We can rewrite our taco like below, using a colon `:` to separate the `key` from the `value` and using commas `,` to sperate each `key value pair`.

```
var taco1 = {
    "tortilla": "soft corn tortilla",
    "protein":  "tinga chicken",
    "cheese":   "cotija cheese",
    "toppings": ["lettuce", "pico de gallo", "guacamole"]
}
```

Now that we have our taco in an object we can access the values that are stored using the dot `.` syntax.

```
console.log("Tortilla: " + taco1.tortilla);
console.log("Protein:  " + taco1.protein);
console.log("Cheese:   " + taco1.cheese);
console.log("Toppings: " + taco1.toppings);
```

As delicious as it would be, there are no languages that have a built in property called `.cheese` in the objects that are made using it. It turns out that all of our properties can be accessed this way which will make it easy for us to remember how to get them. If we use really generic keys like `x`, `y`, and `z` we could still access the information but it would be hard to know what it's for (unless it's a 3 dimensional point perhaps?).

Methods - giving actions to objects

Objects are great at storing information in a logical way, but objects can provide even more than that. Our objects are also able to contain actions they can **do**. It's hard to imagine our taco as **doing** much beyond being highly delectable, but we can take the `console.log()` lines

from above to make some functionality.

```
var taco1 = {
    "tortilla": "soft corn tortilla",
    "protein":  "tinga chicken",
    "cheese":   "cotija cheese",
    "toppings": ["lettuce", "pico de gallo", "guacamole"],
    "tacoInfo": function() {
        console.log("Tortilla: " + taco1.tortilla);
        console.log("Protein:  " + taco1.protein);
        console.log("Cheese:   " + taco1.cheese);
        console.log("Toppings: " + taco1.toppings);
    }
}

// we can now get all the delicious taco facts by
taco1.tacoInfo(); // note we call this like a function because it is a function
```

It can't be helped... We'll have to use 'this'

Just like in your favorite anime when the protagonists must use some secret weapon (usually *that*) to save the day, JavaScript objects can save us from having the retype the same variable names many times! The special keyword that will let us do this is actually called `this`. Using `this` we can rewrite the above like so.

```
var taco1 = {
    "tortilla": "soft corn tortilla",
    "protein":  "tinga chicken",
    "cheese":   "cotija cheese",
    "toppings": ["lettuce", "pico de gallo", "guacamole"],
    "tacoInfo": function() {
        console.log("Tortilla: " + this.tortilla);
        console.log("Protein:  " + this.protein);
        console.log("Cheese:   " + this.cheese);
        console.log("Toppings: " + this.toppings);
    }
}

// we can now still get all the delicious taco facts by
taco1.tacoInfo(); // note tacoInfo still gets called like a function
```

It's time to complete your weekly discussion post, in order to be counted "present" on attendance.

Click here to navigate over and participate.