<https://www.freecodecamp.org/news/how-javascript-implements-oop/>

JavaScript is not a class-based object-oriented language. But it still has ways of using object oriented programming (OOP).

According to Wikipedia(https://en.m.wikipedia.org/wiki/Class-based\_programming), class-based programming is

a style of Object-oriented programming (OOP) in which inheritance occurs via defining classes of objects, instead of inheritance occurring via the objects alone

JavaScript isn't a classed-based langauge – it's is a prototype-based langauge.

According to Mozilla's documentaion:

A prototype-based language has the notion of a prototypical object, an object used as a template from which to get the initial properties for a new object.

**The \_\_proto\_\_ property**

This points to the object which is used as a prototype.

This is the property on every object that gives it access to the **Object prototype property**

Every object has this property by default, which refers to the Object Protoype except when configured otherwise (that is, when the object's \_\_proto\_\_ is pointed to another prototype).

The **Object.create()** method creates a **new object**, using an **existing object as the prototype of the newly created object.**



Object.create uses the argument passed to it to become the prototype.



This technique is referred to as **PROTOTYPE CHAINING**.

Note that: the new keyword approach does the same thing as Object.create() but only makes it easier as it does some things automatically for you.

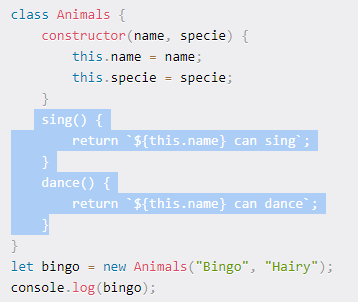
### Object + Function Combination

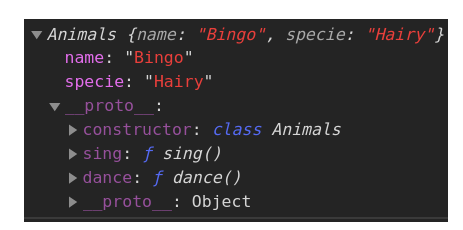
You are probably confused by the fact that DogObject is a function (function DogObject(){}) and it has properties accessed with a dot notation. This is referred to as a function object combination.

When functions are declared, by default they are given a lot of properties attached to it. Remember that functions are also objects in JavaScript data types.

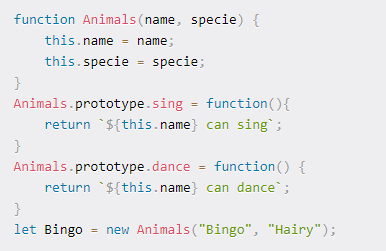
## Now, Class

JavaScript introduced the class keyword in ECMAScript 2015. It makes JavaScript seem like an OOP language. But it is just **syntatic sugar over the existing prototyping technique**. It continues its prototyping in the background but makes the outer body look like OOP





The \_\_proto\_\_ references the Animals prototype (which in turn references the Object prototype).



From this, we can see that the constructor defines the major features while everything outside the constructor (sing() and dance()) are the bonus features (**prototypes**).