Inheritance and the prototype chain

Prototype

In JavaScript, functions are able to have properties. All functions have a special property named **prototype**.

Each object has a private property which holds a link to another object called its **prototype**.

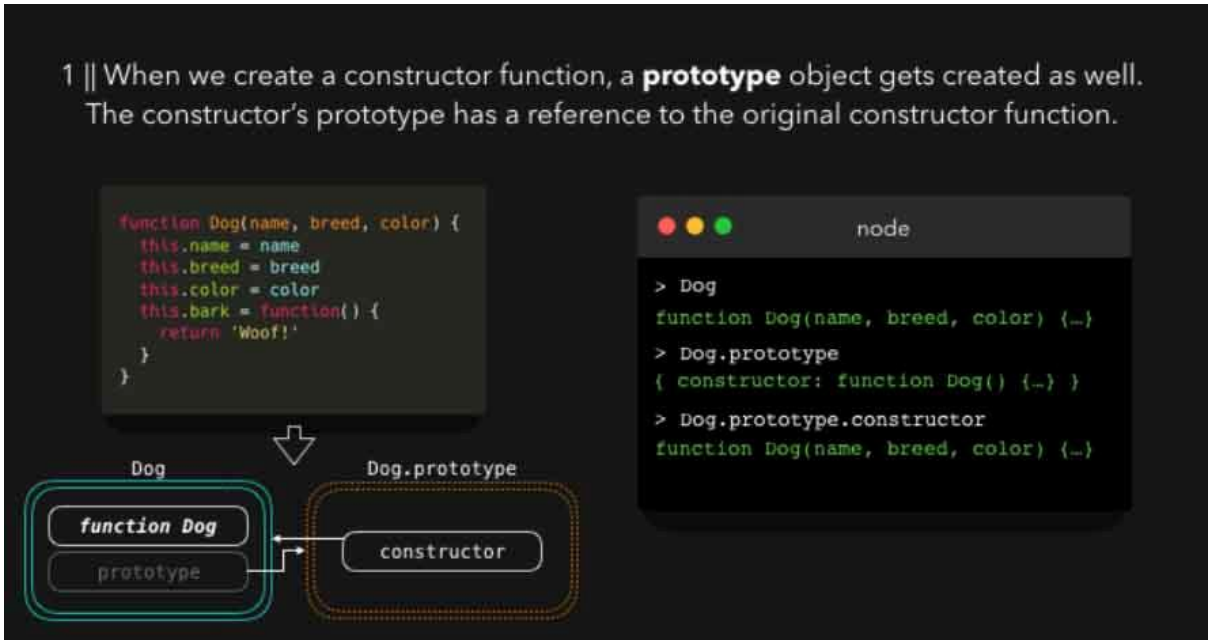
Nearly all objects in JavaScript are instances of Object which sits on the top of a prototype chain.

When trying to access a property of an object, the property will not only be sought on the object but on the prototype of the object, the prototype of the prototype, and so on until either a property with a matching name is found or the end of the prototype chain is reached.

Let us create a constructor function for Dog.

```
function Dog(name, breed, color) {
  this.name = name
  this.breed = breed
  this.color = color
  this.bark = function() {
    return 'Woof!'
  }
}
```

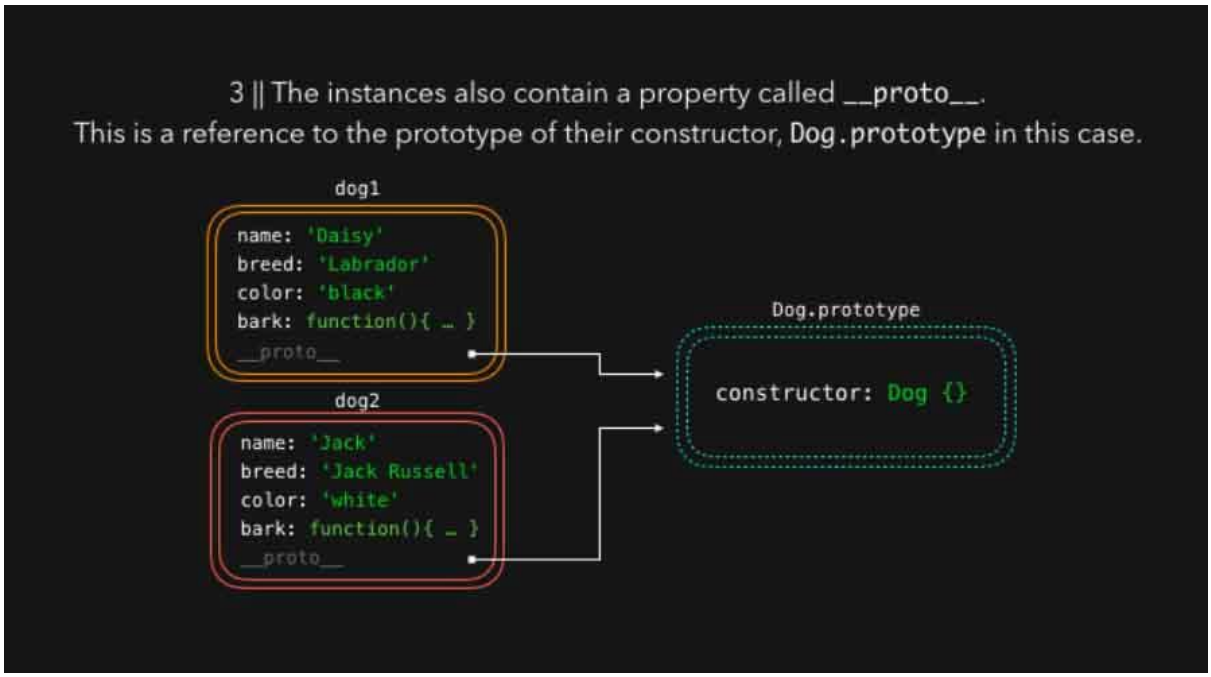
When we created Dog constructor function, we also created another object called **"prototype".** By default, this object contains a *constructor* property, which is simply a reference to the original constructor function.



Now,let us create some Dog objects.

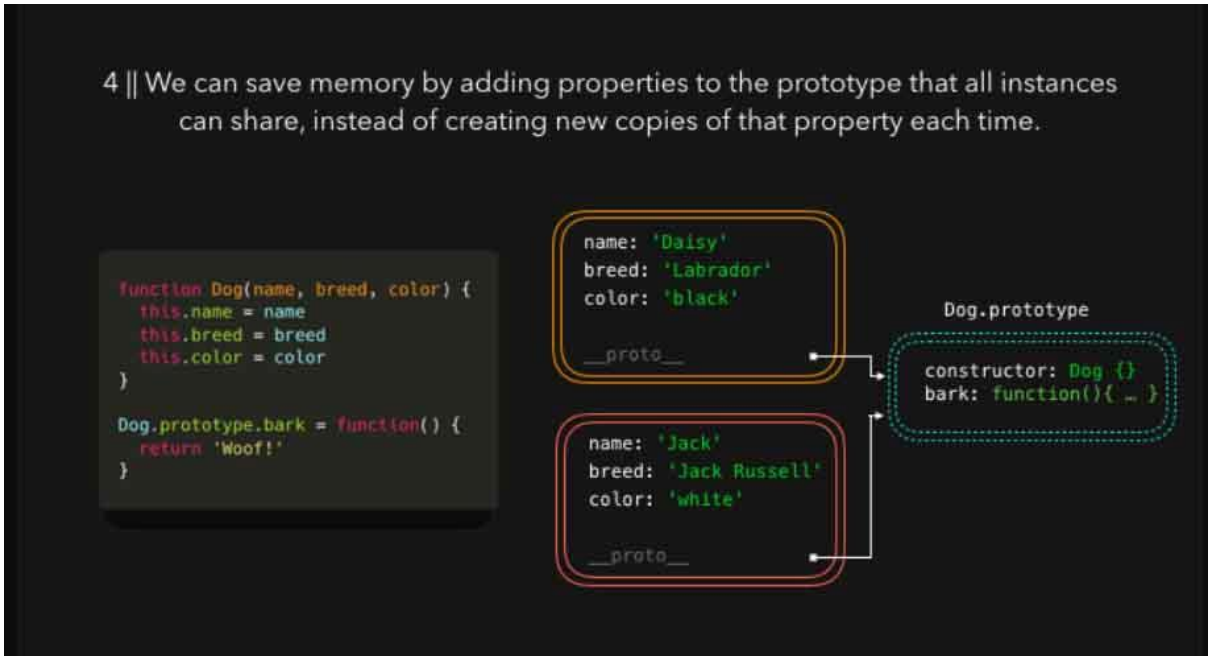
```
const dog1 = new Dog(
  "Daisy",
  "Labrador",
  "black"
)

const dog2 = new Dog(
  "Jack",
  "Jack Russell",
  "white"
)
```

When print that object to console, we can see a special property **“_ _ proto _ _”** for each object. When we expand it, it exactly look like “Dog.prototype” object. That means **“ _ _ proto _ _ “** is a reference to **Dog.prototype** object**.** This is called **prototypal inheritance** is all about: each instance of the constructor has access to the prototype of the constructor.



Use of prototypal inheritance

Sometimes we have properties that all instances share. For example the bark function in this case: it's the exact same for every instance, why create a new function each time we create a new dog, which consumes memory every time. Instead, we can add it to the **Dog.prototype** object



Whenever we try to access a property on the instance, the engine first searches locally to see if the property is defined on the object itself. However, if it can't find the property we're trying to access, the engine **walks down the prototype chain** through the __proto__ property