
# Object Oriented JavaScript

# Introduction

JavaScript is very different from Ruby in object oriented programming.

# Introduction

Unlike Ruby, in JavaScript there are several ways to create our own object.

# Objects in JavaScript

## A basic way to create an object.

```
var animal = {
  name: "Shadow",
  noise: "Brrrr",

  shout: function () {
    console.log("Mooooo!");
  },

  makeNoise: function () {
    // more about `this` momentarily
    console.log(this.noise + "!!!!");
  }
};
```

# `this` in JavaScript

`this` is a magical variable that gets set when we call a function

```
animal.makeNoise();
```

With the variable `this` we can access the attributes of that object.

# `this` in JavaScript

Inside the `makeNoise` function we can access the noise attribute via `this`.

```
var animal = {
  name: "Whitey",
  noise: "Brrrr",

  shout: function () {
    console.log("Mooooo!");
  },

  makeNoise: function () {
    // more about `this` momentarily
    console.log(this.noise + "!!!!");
  }
};
```

# `this` in JavaScript

When `makeNoise` or any method attached to `animal` is called, inside that method `this` *is* `animal`.

```
var animal = {
  // ...

  makeNoise: function () {
    // more about `this` momentarily
    console.log(this.noise + "!!!!");
  }
};
```

# Classes

JavaScript doesn't have a keyword to create classes like `class` in Ruby. In JavaScript we use `function`.

# Creating a class

## We create a function

```
var Animal = function () {};
```

# Creating an instance

We call the function with the `new` keyword

```
var animal = new Animal();
```

# Ruby vs JavaScript

## Javascript constructor

```
var Animal = function () {};
```

## Creating an instance

```
var dog = new Animal();
```

## Ruby Constructor

```
class Animal
end
```

## Creating an instance

```
dog = Animal.new
```

# `new` keyword

Invoking a function with the special `new` keyword calls the function with `this` set to a new blank object. This blank object will be returned by the constructor as the new `Animal` instance.

```
var animal = new Animal();
```

# Constructor and `this`

Here, `Animal` is a constructor.

```
var Animal = function() {
   this.name = "Buk";

   this.noise = "Brrrr";

   this.makeNoise = function () {
      console.log(this.noise + "!!!!");
   }
};
```

# The Instance

When we create an instance, what we have is a new object.

```
var animal = new Animal();
var anotherAnimal = new Animal();
```

# How to customize

Let's add parameters to the constructor.

```
var Animal = function(name, noise) {
   this.name = name;

   this.noise = noise;

   this.makeNoise = function () {
      console.log(this.noise + "!!!!");
   }
};
```

# The Instance

Since it's a new object, we can set different values for its attributes.

```
var animal = new Animal("Buk", "Brrr");
var anotherAnimal = new Animal("Chinaski", "Pffff");

animal.makeNoise()
// "Brrr!!!!"

anotherAnimal.makeNoise()
// "Pffff!!!!"
```

# Object Prototype

Both Animals have identical `makeNoise` functions.

```
var Animal = function(name, noise) {
    this.name = name;

    this.noise = noise;

    this.makeNoise = function () {
        console.log(this.noise + "!!!!");
    }
};
```

# Object Prototype

But they don't share a single function. Each instance has it's own identical `makeNoise`.

```
var Animal = function(name, noise) {
    this.name = name;

    this.noise = noise;

    this.makeNoise = function () {
        console.log(this.noise + "!!!!");
    }
};
```

# Object Prototype

## We are duplicating code unnecessarily!

```
var Animal = function(name, noise) {
    this.name = name;

    this.noise = noise;

    this.makeNoise = function () {
        console.log(this.noise + "!!!!");
    }
};
```

# Object Prototype

Objects can get their functions from their `prototype`.
If we use the prototype, all instances share a single function.

```
var Animal = function(name, noise) {
    this.name = name;
    this.noise = noise;
};

Animal.prototype.makeNoise = function () {
    console.log(this.noise);
};
```

# Exercise

- Create the `Car` constructor. It should take two parameters, model and noise
- Each car should have 3 attributes: model, noise and number of wheels
- Model and noise are set with the parameters and number of wheels will be 4 for all the instances of `Car`
- Create a method to print the noise of that car