Adding Behavior With Methods

- Due No Due Date
- Points 1
- Submitting a website url

[(https://github.com/learn-co-curriculum/phase-1-adding-behavior-with-methods)](https://github.com/learn-co-curriculum/phase-1-adding-behavior-with-methods) [(https://github.com/learn-co-curriculum/phase-1-adding-behavior-with-methods/issues/new)](https://github.com/learn-co-curriculum/phase-1-adding-behavior-with-methods/issues/new)

Learning Goals

- Write methods that use instance data and parameter data

Introduction

In review, with *Object-Oriented programming (OOP)*, we can use classes to represent concepts such as students, books, comments, posts, or even animals.

We should only have to define the properties and methods of a class once. Different *instances* of this class will all have the same properties and methods. Specific values for those properties will be different between instances. Mickey and Minnie are both `Mouse` instances that have a property called `name` , but the value of that property for each is different.

With knowledge of `constructors` , we can use JavaScript's `class` es as a template for *instances*.

Write a Method That Uses Instance Data and Parameter Data

To practice *OOP* concepts, let's create 3 `class` es that use `constructor` methods. These `constructor` s will assign properties based on initial parameters. We'll also write methods that leverage these properties.

1. Create `class` es `Cat` , `Dog` , and `Bird`
2. Each of these `class` es will accept the *parameters* `name` and `sex` and will store those values as *properties*.

```
class Cat {
  //...
}

class Dog {
  //...
}

class Bird {
  //...
}
```

For each `class` , create the method `speak` .

- For an *instance* of `Cat` , speak returns " `name` says meow!",
- For an *instance* of `Dog` , speak returns " `name` says woof!"
- For an *instance* of `Bird` , speak returns conditional output. If the *instance* of `Bird` is `male` , speak returns "It's me! `name` , the parrot!". If it is not `male` , speak returns " `name` says squawk!".

Conclusion

We've learned to instantiate class instances, or "objects" in JavaScript. The constructor function allows us to easily define and standardize the instances we create. Good work!

Resources

- **Mozilla Developer Network (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this)** - **this (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/this)**
- **JavaScript—Multiple Ways to Create Objects (https://codeburst.io/various-ways-to-create-javascript-object-9563c6887a47)**