Getter and Setter Methods

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

[(https://github.com/learn-co-curriculum/phase-1-getter-and-setter-methods)](https://github.com/learn-co-curriculum/phase-1-getter-and-setter-methods) [(https://github.com/learn-co-curriculum/phase-1-getter-and-setter-methods/issues/new)](https://github.com/learn-co-curriculum/phase-1-getter-and-setter-methods/issues/new)

Learning Goals

- Build getter and setter methods to calculate computed properties

Introduction

We've seen so far that we can write methods inside our `class` es that allow us to access and change properties. These methods work fine in some cases. However, we know about additional JS syntax that we can use: `get` and `set` . With `get` , we can return calculated or dynamic data based on object properties, and with `set` we can change a property in a controlled way. In this lab, we will be building pseudo-properties to illustrate these concepts.

Build Getter and Setter Method to Calculate Computed Properties

Let's look at a quick example for review:

```
class Bird {
  #phrase;

  constructor(name) {
    this.name = name;
  }

  set phrase(phrase) {
    this.#phrase = phrase;
  }

  get speak() {
    return `${this.name} says ${this.#phrase || "squawk"}`;
  }
}

const daffy = new Bird("Daffy");
daffy.speak; // => 'Daffy says squawk'
daffy.phrase = "it's rabbit season!";
daffy.speak; // => 'Daffy says it's rabbit season!'
```

Our `Bird` class accepts the parameter of `name`, which is set each time a new instance of `Bird` is created. When `phrase` is set, our new `Bird` instance can speak a phrase. If it is not set, it will squawk.

```
const buddy = new Bird("Buddy");
buddy.phrase = "What'cha doin'?";
buddy.speak; // returns 'Buddy says What'cha doin'?'
```

Building on this concept, we're going to build our own shape calculator!

- First, let's create a class of `Circle`
- `Circle` will accept 1 parameter, `radius`, and use `this.radius` to store the value
- Use `Math.PI` to get an accurate measurement of pi (https://en.wikipedia.org/wiki/Pi) (π)
- Define getter methods for `diameter`, `circumference`, and `area` which will calculate each value using `this.radius` and pi
- Define setter methods for `diameter`, `circumference`, and `area` which will accept values for each calculation, calculate the *radius* based on the input value and *set* `this.radius` accordingly
  - **Hint:** You will need to use Math.sqrt() (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/sqrt) in your `area` setter method

For reference, here are the formulas for calculating diameter, circumference and area:

- Diameter = radius • 2
- Circumference = π • diameter
- Area = π • radius^2

> Don't forget about **PEMDAS** **(https://en.wikipedia.org/wiki/Order_of_operations)** !

All instances of `Circle` should be able to calculate the `diameter` , `circumference` , and `area` based on the given `radius` . All instances should also be able to set `this.radius` by setting a value to `diameter` , `circumference` , or `area` .

Conclusion

Getter and setter methods are very useful for doing things behind-the-scenes in JavaScript. Using a setter, you can call a function each time the value of a pseudo-property is changed, making sure all data on a `class` instance is consistent. Using a getter, you can return a computed value as though it were a property! Just as you can set and retrieve basic information from properties, you can also perform a number of functions that will "automagically" spit out the output you want.

Resources

- **Property getters and setters** **(https://javascript.info/property-accessors)**