Constructors, Prototypes, and the Prototype Chain

- You can extend the prototype chain and allow objects to inherit behaviors from other prototypes

Method 1: Using the Object Returned by the Constructor

- Ex: We have a Dog prototype and the Object prototype but we want an Animal prototype in between
- `Dog.prototype = new Animal(arguments)`
- This approach uses the object that the Animal constructor returns as the value assigned to `Dog.prototype`
    - Any object you create form the Dog constructor will by default have the properties of the object returned from the Animal constructor
    - Deviates from the idea that only behavior is shared through the prototype chain
    - Could make it so that users of the Dog constructor aren't aware of the existence of all the properties its inheriting

Method 2: Using the Object created by `Object.create(Animal.prototype)`

- `Dog.prototype = Object.create(Animal.prototype)`
- Assigns the object returned by Object.create on Animal.prototype to `Dog.prototype`
- Doing this leverages the fact that `Object.create` returns have their `__proto__` property set to the object that was passed in as an argument
- The benefit is that there are no new properties that can be found on the created object and it is only "behavior" that is shared