

# JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

# HELLO!

1. Pull changes from the `svodnik/jsd5` repo to your computer
2. Navigate to the `starter-code` folder

---

**JAVASCRIPT DEVELOPMENT**

---

# **PROTOTYPAL INHERITANCE**

# LEARNING OBJECTIVES

At the end of this class, you will be able to

- Explain the difference between literal and constructed objects.
- Write a constructor for a JavaScript object.
- Explain prototypal inheritance and its purpose.
- Recognize the difference between prototypal and classical inheritance.
- Create and extend prototypes.

# AGENDA

- Objects and constructors
- Prototypal inheritance

## **Checkin on Unit #2 Project – Feedr**

- The **biggest challenge I have faced or anticipate facing** in completing the Feedr project is \_\_\_\_\_.
- (If you have already started on the Feedr project:) My **biggest accomplishment** so far in working on Feedr is \_\_\_\_\_.

**What are some advantages of using templates in JavaScript?**

# **CREATE A TORTOISE OBJECT**





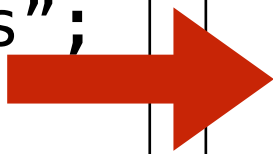
# PROTOTYPES

- › Every object in JS has a `prototype` property, which is a reference to another object
- › The object that the `prototype` property points to is generally an instance of the constructor object
- › Any properties/methods defined on an object's prototype are available on the object itself, without defining those properties/methods a second time
- › The relationship between objects that have a prototypal relationship with each other is known as the **prototype chain**

# Using the prototype property

```
function Dog(name, breed) {  
  this.name = name;  
  this.breed = breed;  
}  
Dog.prototype.species = "Canis Canis";  
Dog.prototype.bark = function() {  
  return "Woof! I'm " + this.name;  
}
```

**Dog.prototype**



```
{  
  species: "Canis Canis",  
  bark: function() {  
    return "Woof! I'm " +  
      this.name;  
  }  
}
```

# Using the prototype property

```
var spot = new Dog("Spot", "Beagle");
```

spot object (constructed)

individual properties created  
by the constructor function

inherited from  
Dog.prototype object

```
{  
  name: "Spot",  
  breed: "Beagle",  
  species: "Canis Canis",  
  bark: function() {  
    return "Woof! I'm " + this.name;  
  }  
}
```

# PROTOTYPE TERMINOLOGY

- `prototype`: a model used to create instances
- `prototype` property: a reference to another object that is generally an instance of the constructor object
- `__proto__` (or “dunder proto”): a property used by web browsers that indicates an object’s parent in the prototype chain

# CLASS VS PROTOTYPE

- In class-based languages, classes are objects that
  - manufacture new objects
  - define the behavior of the objects they manufacture
- In JavaScript
  - a constructor function manufactures new objects
  - a prototype property defines the behavior of new objects manufactured by the constructor

# Object.create()

- Creates a new object
- Sets prototype of new object to be existing object
- Some differences under the hood, but essentially equivalent to using the new keyword
- Example:
  - `var me = Object.create(Person)`
  - equivalent to `var me = new Person();`

# LEARNING OBJECTIVES – REVIEW

- Explain the difference between literal and constructed objects.
- Write a constructor for a JavaScript object.
- Explain prototypal inheritance and its purpose.
- Recognize the difference between prototypal and classical inheritance.
- Create and extend prototypes.

## **NEXT CLASS PREVIEW**

### **this and the module pattern**

- › Understand and explain Javascript context.
- › Implement the module pattern in your code.



**Exit Tickets!**

# Q&A