

JAVASCRIPT DEVELOPMENT

Sasha Vodnik, Instructor

HELLO!

- 1. Pull changes from the JS-SF-16-resources repo to your computer
- 2. Open the 13-prototypal-inheritance folder in your editor

PROTOTYPAL INTERIOR PROTOT

LEARNING OBJECTIVES

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

- Distinguish between classical and prototypal inheritance
- Explain the difference between literal and constructed objects.
- Write a constructor for a JavaScript object.
- Explain prototypal inheritance and its purpose.
- Create and extend prototypes.
- Work with prototypes using class keyword syntax

AGENDA

- Objects and constructors
- Prototypal inheritance
- The class keyword

PROTOTYPAL INHERITANCE

WEEKLY OVERVIEW

WEEK 7

Project 2 lab / Prototypal inheritance

WEEK 8

Closures & this / CRUD & Firebase

WEEK 9

Deploying your app / React

ACTIVITY



KEY OBJECTIVE

Check in on Feedr

TYPE OF EXERCISE

Pairs

TIMING

6 min

- 1. Share your biggest accomplishment so far in working on Feedr
- 2. Share a challenge you've encountered. If you overcame it, share how; if not, brainstorm with your partner on an approach.
- 3. Share your next step.



EXIT TICKET QUESTIONS

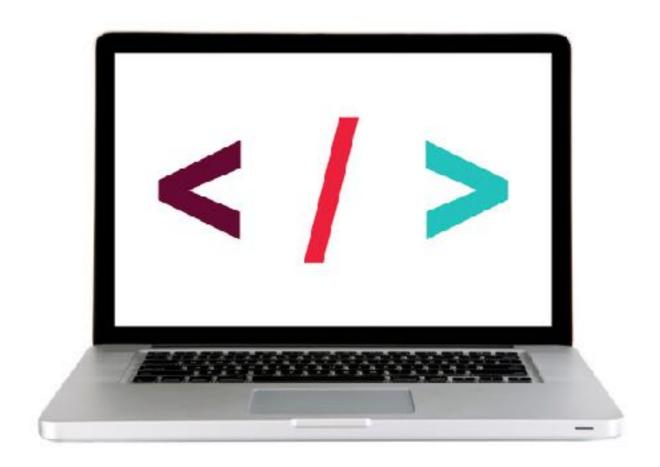
- Are there any collections or resources of pre-made HTML templates out there to be used as a front-end for JS apps? Would be helpful for making standalone apps after the class is over.
- If we continue to look for free APIs, and sign up for them could this open us up to a cybersecurity threat?
- What's a API-first website? people use this term when they talk about modern web app development

OBJECTS AND INSTANCE

JAVASCRIPT DEVELOPMENT

CONSTRUCTORS

LET'S TAKE A CLOSER LOOK



EXERCISE — CREATE A MAKECAR FUNCTION



TYPE OF EXERCISE

Individual/pair

LOCATION

▶ start files > 01-make-car-exercise

TIMING

8 min

1. In app.js, Define a function called makeCar() that takes two parameters (model, color), makes a new object literal for a car using those params, and returns that object.

PROTOTYPAL INHERITANCE

CLASS VS PROTOTYPE

CLASS-BASED LANGUAGE

class

-manufactures new objects

defines behavior of manufactured objects

JAVASCRIPT

constructor

manufactures new objects

prototype

 defines behavior of manufactured objects

CONSTRUCTOR FUNCTIONS

LET'S TAKE A CLOSER LOOK



EXERCISE — MAKE A CAR CONSTRUCTOR FUNCTION



TYPE OF EXERCISE

Individual/pair

LOCATION

▶ start files > 03-constructor-exercise

TIMING

8 min

- 1. In app.js, write a constructor function to replace our makeCar function from earlier.
- 2. Your constructed objects should include the same properties as in the 1-make-car-function exercise.

EXERCISE — LITERAL VS CONSTRUCTED OBJECTS



TYPE OF EXERCISE

• Groups of 2 or 3

TIMING

3 min

- 1. Spend 30 seconds thinking about the difference between literal and constructed objects.
- 2. Form a pair or group of 3, then take turns explaining how you understand the difference between the two.
- 3. Be prepared to share your thoughts with the class.

JAVASCRIPT DEVELOPMENT

PROTOTYPES

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

```
let 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

LET'S TAKE A CLOSER LOOK



EXERCISE — MAKE A MONKEY CONSTRUCTOR FUNCTION AND PROTOTYPE



TYPE OF EXERCISE

Individual/pair

LOCATION

> start files > 06-prototypes-exercise

TIMING

8 min

- 1. In app.js, create a Monkey constructor that meets the specs described.
- 2. Create 3 objects using your Monkey constructor and verify that all properties and methods of each have the expected values.

LET'S TAKE A CLOSER LOOK



LAB - BUILD A PROTOTYPE CHAIN



TYPE OF EXERCISE

Individual/pair

LOCATION

start files > 08-prototypes-lab

TIMING

10 *min*

- 1. Create an Item constructor using the specs in the start file.
- 2. Create Clothing and Household constructors and use Item as the prototype for each.
- 3. Test your work in the browser console.
- 4. If you finish early, work on the bonus items described in app.js.

JAVASCRIPT DEVELOPMENT

CLASSES IN JAVASCRIPT

CLASS VS PROTOTYPE

CLASS-BASED LANGUAGE

class

-manufactures new objects

-defines behavior of manufactured objects

JAVASCRIPT

constructor

manufactures new objects

prototype

 defines behavior of manufactured objects

PROTOTYPAL INHERITANCE

CLASS KEYWORD

```
function Vehicle(color) {
  this.color = color;
}
```

```
class Vehicle {
  constructor(color) {
    this.color = color;
  }
}
```

CONSTRUCTOR

CLASS KEYWORD (ES6)

CLASS + EXTENDS & SUPER KEYWORDS

PROTOTYPAL INHERITANCE

CONSTRUCTOR + PROTOTYPE

EXTENDS AND SUPER KEYWORDS

```
function Car(color) {
    this.color = color;
}

Car.prototype = new Vehicle();

class Car extends Vehicle() {
    constructor(color) {
        super(color);
    }
}
```

LAB - REFACTOR CODE TO USE CLASSES



TYPE OF EXERCISE

Individual/pair

LOCATION

▶ start files > 10-classes-lab

TIMING

until 9:25

- 1. Refactor the Item, Clothing, and Household prototypes and constructors to recreate the same prototype chain using class keyword syntax.
- 2. Test your work in the browser console.

Exit Tickets!

(Class #13)

LEARNING OBJECTIVES - REVIEW

- Distinguish between classical and prototypal inheritance
- Explain the difference between literal and constructed objects.
- Write a constructor for a JavaScript object.
- Explain prototypal inheritance and its purpose.
- Create and extend prototypes.
- Work with prototypes using class keyword syntax

NEXT CLASS PREVIEW Closures & this

- Understand and explain Javascript context.
- Understand and explain closures.
- Instantly invoke functions.
- Predict what the this keyword refers to in different situations.

