Constructors

What is an object?

Activity: Raining Cats and Dogs

See 01-RainingCatsAndDogs-NoCon in class repo for instructions

Review: 01-RainingCatsAndDogs-NoCon

- How do we create an object?
- How do we declare properties and values?
- How do we create a method?
- How do we access an object property?
- How do we use an object method?
- What is this?

What is DRY?

Cats vs. Dogs

```
var dogs = {
  raining: true,
  noise: "Woof!",
  makeNoise: function() {
    if (this.raining === true) {
      console.log(this.noise);
    }
  }
};
```

```
var cats = {
  raining: false,
  noise: "Meow!",
  makeNoise: function() {
    if (this.raining === true) {
      console.log(this.noise);
    }
  }
};
```

There must be a better way!

Constructors!

Demo/Review: 02-RainingCatsAndDogs-Con

- Open rainingCatsAndDogs-con.js
- What’s different?

What is a constructor?

What is the coding convention for declaring constructors?

What is the keyword we use to create an object?

Activity: Character Creation (30 min)

See 03-CharacterCreate in class repo for instructions...

Review: 03-CharacterCreate

Building on Constructors

What if we want to add new and/or unique properties and methods to a constructed object?

```
dogs.exercise = true;
```

```
cat.nap = function() {
// some code here...
}
```

Activity: Tamagotchi (40 min w/Partner)

See 04-Tamagotchi in class repo for instructions

Review: 04-Tamagotchi

Activity: Programmers

See 05-Programmers in class repo for instructions

EVERYONE must do!

Because you’re all programmers now :)

Review: 05-Programmers

prototype

- Constructors are prototypes
- All JavaScript objects inherit the properties and methods from their prototype.
- You cannot add a new property to a prototype the same way as you add a new property to an existing object, because the prototype is not an existing object.
- The JavaScript prototype property allows you to add new properties and methods to an existing prototype

RTFM https://www.w3schools.com/js/js_object_prototypes.asp

In this example, the process of creating new Programmers could get rather tiresome since we are constantly having to modify our code each and every time.

There must be a better way!

Demo: Programmers with Prompt

- Note: The creation of our new Programmer object and the calling of its printInfo() are located within the .then statement.
  - This is done to ensure that the constructor can appropriately grab the answers while also making certain that the printInfo() method is only run when there is a populated object. Doing otherwise would return an error.

Activity: Multiple Programmers

Modify 06-ProgrammersWithPrompt to create more than one object


Canine Influenza Vaccine
H3N8

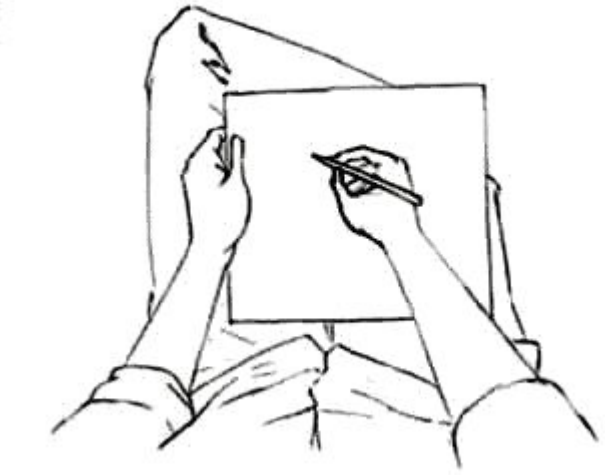

MWA HA HA HA HA HA HA

There must be a better way!

Recursion, see *Recursion*

Activity: Recursion

- Visit google.com
- Search: recursion



Recursion: Factorial

- You may remember from math class that the factorial of a number n is the product of all positive integers less than or equal to n.
- In other words, the factorial of 5 is 5 x 4 x 3 x 2 x 1.
  - Answer: 120
- The mathematical notation for this is 5!.
- 5! === 5 x 4!
- 4! === 4 x 3!
- Etc.

RTFW https://en.wikipedia.org/wiki/Factorial

Recursion: Factorial

```
function factorial( n ) {
 if ( n === 1 ) {
   return 1;
 }
 return n * factorial( n - 1 );
}
```

1. factorial(3) is 3 x factorial(2).
2. factorial(2) is 2 x factorial(1).
3. factorial(1) meets our if condition, so it's just 1.

||

1 x 2 x 3 equals 6

Demo: Factorial

Demo: Programmer’s Loop

- programmersLoop-noRecursion.js
- It appears that inquirer is asking the same question five times at once and is only accepting a single response.
- This creates five Programmer objects that are all exactly the same as each other which is not useful at all. What gives!?
- Our for loop is the culprit here. It is not waiting for the previous instance of inquirer to complete before moving onto the next one. As such, inquirer is being run multiple times at the same time.

Demo: Programmer’s Loop

- programmersLoop-recursion.js
- The way in which we solve this is by utilizing a coding technique called "recursion" to call upon inquirer only after the last instance has been completed.
- While this code works, it is still not 100% effective since our objects are still only obtainable within our function and we are not able to access them afterwards. In order to counteract this, we will want to push all of our objects into an array so that we can call upon them individually at a later time.

Demo: Programmer’s Loop

- programmersLoop-recursionArray.js

Activity: Team Manager

See 08-TeamManager in the class repo for instructions


4GIFs
.com