JavaScript
part 3 - Object creation

CSC309

Mark Kazakevich

Today

- Last week, we started talking about functions and objects

- Today:
    - Some more ways to create objects

“Object-Oriented” JavaScript

- We’ve seen Objects in JS
  - But we’ve also seen how they’re not quite the same as that of other languages
    - For example, how `this` works

- What about something like **classes** and **inheritance**?
  - Do they exist in JS - and how are they different?

Classes

- What are some things that come to mind when you think of ‘classes’?

Classes

- Classes do not exist in JavaScript.
    - At least, not in the way you might think

- Instead of making ‘instances’ or copies of classes and putting them in some hierarchy..
    - JS works on a **delegation** framework
    - If a property can’t be found in an object, JS looks for that property in a *delegate object*
        - Delegate objects can be chained

Prototypes

- Prototypes are objects that are used by other objects to add delegate properties

- Prototypes are **not** superclasses - no instances are created
  - An object will just have a *reference* to its prototype
  - Multiple objects can have the same prototype object reference
    - No copies are made

Prototype demo

Prototypes

- Main purpose of a prototype is for fast object creation
- We will look at some ways to create objects using Prototypes

Object creation using functions

- One way to create an object is to use functions
- These functions are similar to **constructors** in Java
- Functions have their own prototype property that is used for object creation
- Let’s first see an example and then explain what’s happening

Function constructor demo using `new`

`new` keyword

- 4 things that **`new`** does:

1. Creates an empty object
2. Sets the new object's delegate prototype (the `__proto__`) to the constructor's `prototype`
3. Calls the constructor function with **`this`** set to the new object
4. Returns the new object

`__proto__` and prototype

- `__proto__` is the property of an **object** that points to the object's prototype

- `prototype` is the property of a **function** that is used as the prototype to add to the new object when that function is called **as a constructor**

`Object.create()`

- Another way to create objects using prototypes is by using **`Object.create(o)`**
  - Creates an object with **`o`** as the prototype

- Can create multiple objects with same prototype
  - But remember - all of their prototypes will point to the same reference!
    - No instances or copies

`class`

- ES6 includes support for the `class` keyword

- Looks like you can make a class in JS…
    - But it's not *really* a class

- Mostly, it's just a neat way to repackage prototypes and object creation in a way that's more digestible for object-oriented programmers
    - No private variables

`class` demo

`class` review

- JavaScript does not have classes

- Be careful when trying to use JS objects in the same way you have in other languages
  - No new instances of ‘superclasses’
  - Prototypes can be changed at will
    - No private members