JavaScript Objects

CIT261 | Mallory Di Bartolo

What is an object?

- variables that contain many values
- ways to encapsulate related information and actions
- think associative array, but instead of indexing by number you're indexing by name:value pairs

- Basic components:
  - Object
  - Properties (information)
  - Methods (actions)

Creating Objects: Object Literal

```
var bike = {
        name = “Paddywagon”,
        make = “Kona”,
        type = “Road Bike”
        go = function() {
                return this.name + “is moving”;
                }
        };
```

Creating Objects: Constructor

```
function Bike (name, make, type) {
        this.name = name;
        this.make = make;
        this.type = type;
        }

let bike1 = new Bike(‘Paddywagon’, ‘Kona’, ‘Road Bike’);
let bike2 = new Bike(‘Remedy’, ‘Trek’, ’Mountain Bike’);
```

Creating Objects: Create Method

Object.create()

- more attribute options
    - <u>configurable:</u> boolean; default true. decides whether or not we can change certain attributes and whether or not we can delete the property using the delete keyword.

    - <u>enumerable</u>: boolean; default true. decides whether or not we can loop through the properties of an object.

    - <u>writable</u>: boolean; decides if the value can be changed or not.

Object.defineproperty()- define a new or modify an existing property.

Object.create() continued…

```
let bike= Object.create(Object.prototype,
        {
        name : {
                value: ‘Paddywagon’,
                writeable: false,
        }
        make: {
                value: ‘Kona’,
                }
        type: {
                value: ‘Road Bike’,
        }

        });
```

prototype:
- a link between a constructor function and an object

Creating Objects: Classes

```
class Bike {
        constructor(name, make, type) {
                this.name = name;
                this.make = make;
                this.type = type;

                }
        }

let bike1 = new Bike(’Paddywagon’, ‘Kona’, ’Roadbike’);
```

Inheritance w/ Parameters

```
function Bike(name, make, type) {
        this.name = name;
        this.make = make;
        this.type = type;
        };

function CheapBike (name, make, type, price) {
        Bike.call(this, name, make, type)
        this.price = price;
        }
```

Inheritance w/o Parameters

```
function Bike() {
	this.name = “Paddywagon”;
	this.make = “Kona”;
	this.type = “Road Bike”;
}

function CheapBike {
	Bike.call(this);
	this.price = “100”;
}
```

Instantiation Patterns in JavaScript

1. Functional - create a function w/ empty objects inside, properties and methods are added inside the function.
2. Functional-Shared – see above, but methods are defined in another function. Object is extended with these methods.
3. Prototypal- methods are attached to the objects prototype. Use the Object.create method.
4. Pseudoclassical- create properties using the “this” keyword. Methods are assigned to the prototype. Keyword ”new” creates a new object.