# CSC309: Programming on the Web
# Lab 2

Carlos Marciano (TA)

University of Toronto, Department of Computer Science

Fall 2021

carlos.marciano@mail.utoronto.ca

Slides available at: https://cemarciano.github.io/static/documents/lab2.pdf

# Class inheritance VS Prototype inheritance (1)

## Class Inheritance

- Acts as a blueprint, and then an instance can be created using that class code.

```
class Car {
    constructor() {
        this.type = 'vehicle'
    }
}

class Toyota extends Car {
    constructor(color) {
        super(); // Calls Car's constructor
        this.brand = 'Toyota'
        this.color = color
    }
}

const myToyota = new Toyota('red')
console.log(myToyota.type) // will print "vehicle"
```

## Prototype Inheritance

- Objects delegating to other objects. One instance uses another instance as its "property delegate", if a property cannot be found within itself.

```
const car = {
    type: 'vehicle'
};

const toyota = {
  brand: 'Toyota',
  color: 'red'
};

// setting toyota's prototype delegate to be car.
Object.setPrototypeOf(toyota, car);
// Will delegate to object *car* and will output "vehicle":
console.log(toyota.type);
```

# Class inheritance VS Prototype inheritance (2)

**Class Inheritance**

- Factory cannot be modified at run-time, only its resulting objects can, individually.

**Prototype Inheritance**

- "Factory" can be modified at run-time, because the factory itself is an object.

In either case, inheritance is achieved by linking objects using the prototype attribute.