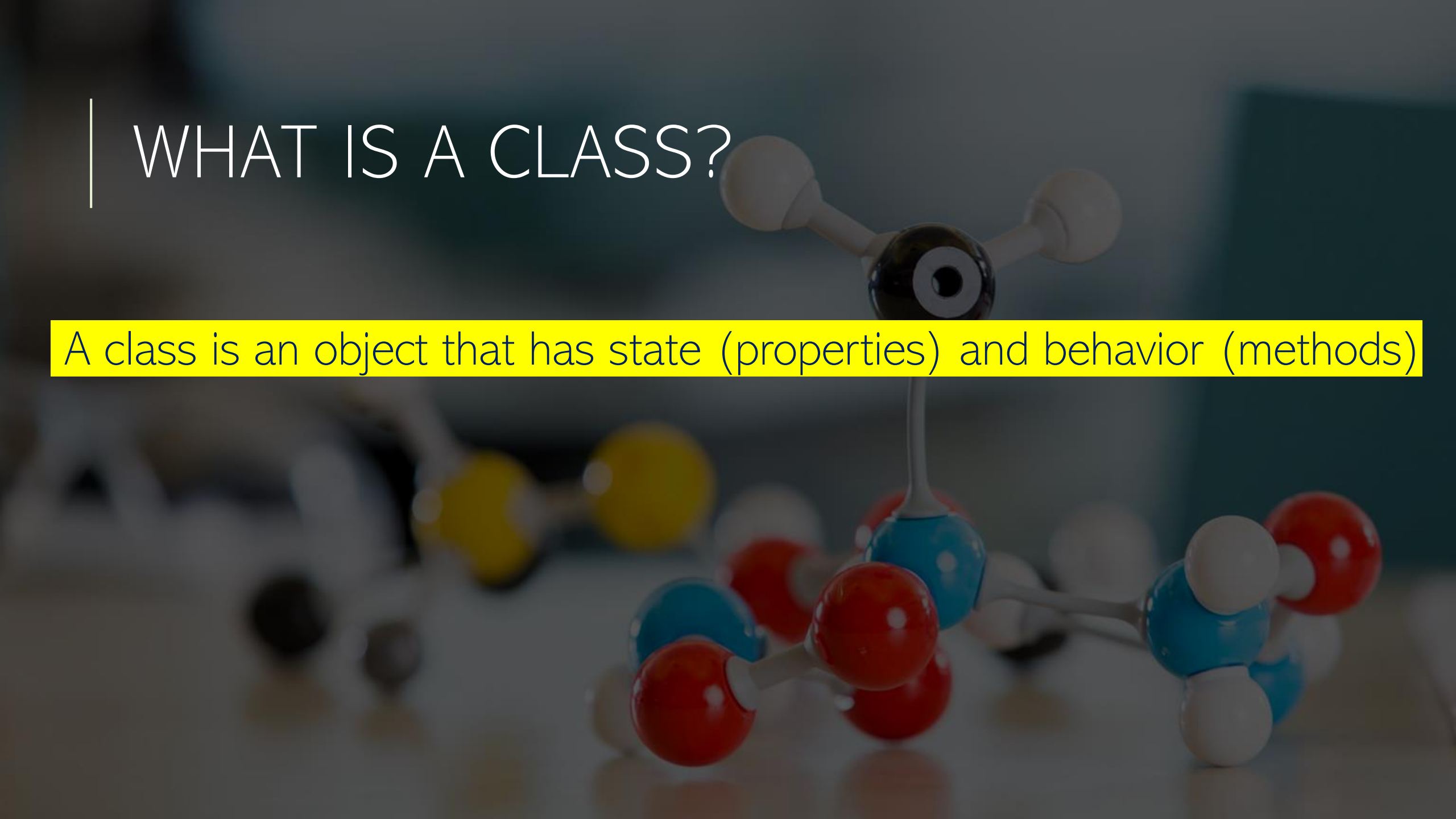


Metropolitan College
Boston University
Computer Science Department

| WHAT IS A CLASS?

A class is an object that has state (properties) and behavior (methods)



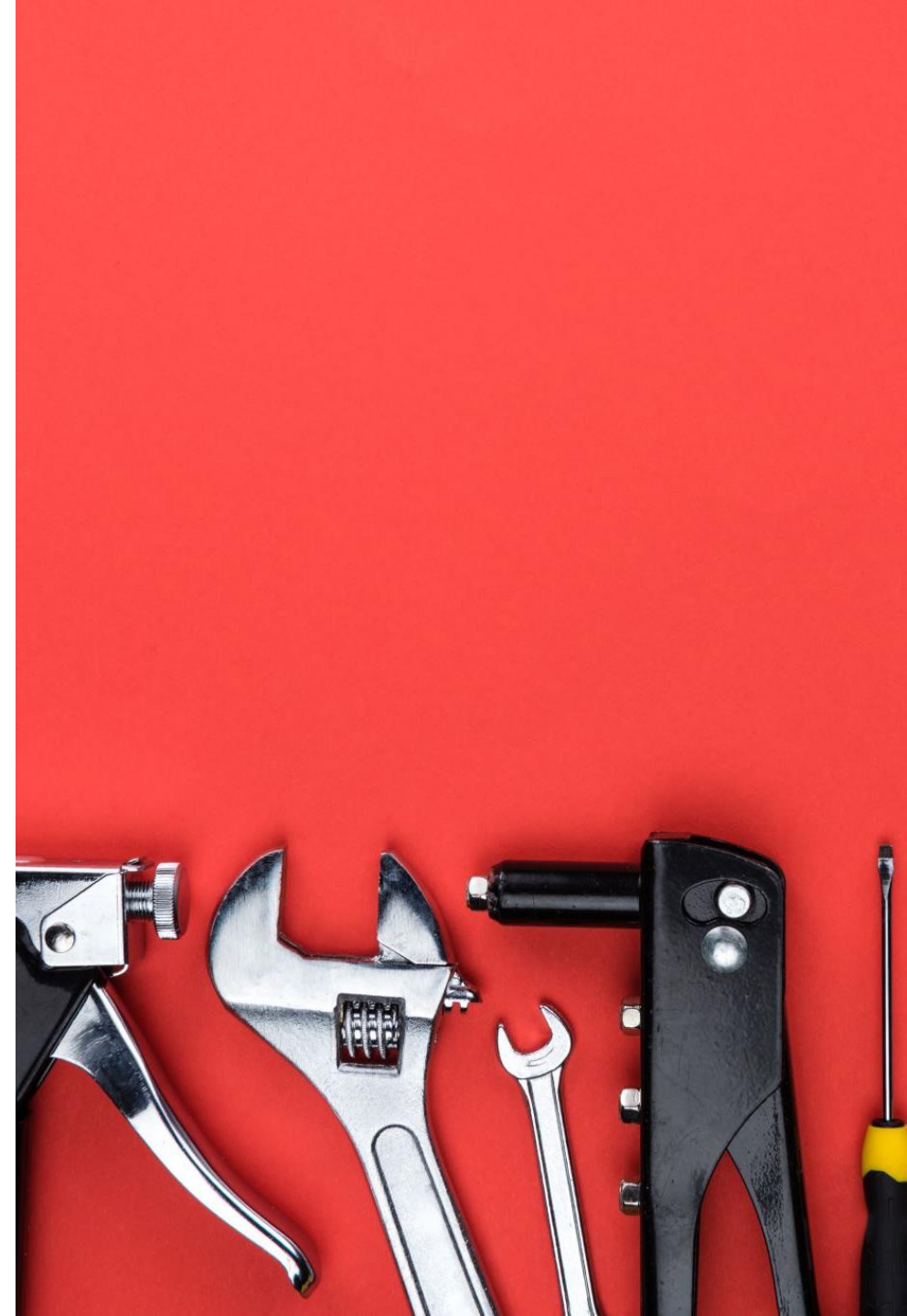
BUILT-IN TYPES

Reference types



(Array, String, Date, Set, Map..)

Primitive types


(boolean, numeric, NULL..)



ES6 classes - OTHER

 **Baseline** Widely available across major browsers  

ES6 classes are syntactical sugar to provide a much simpler and clearer syntax to create objects and deal with inheritance.

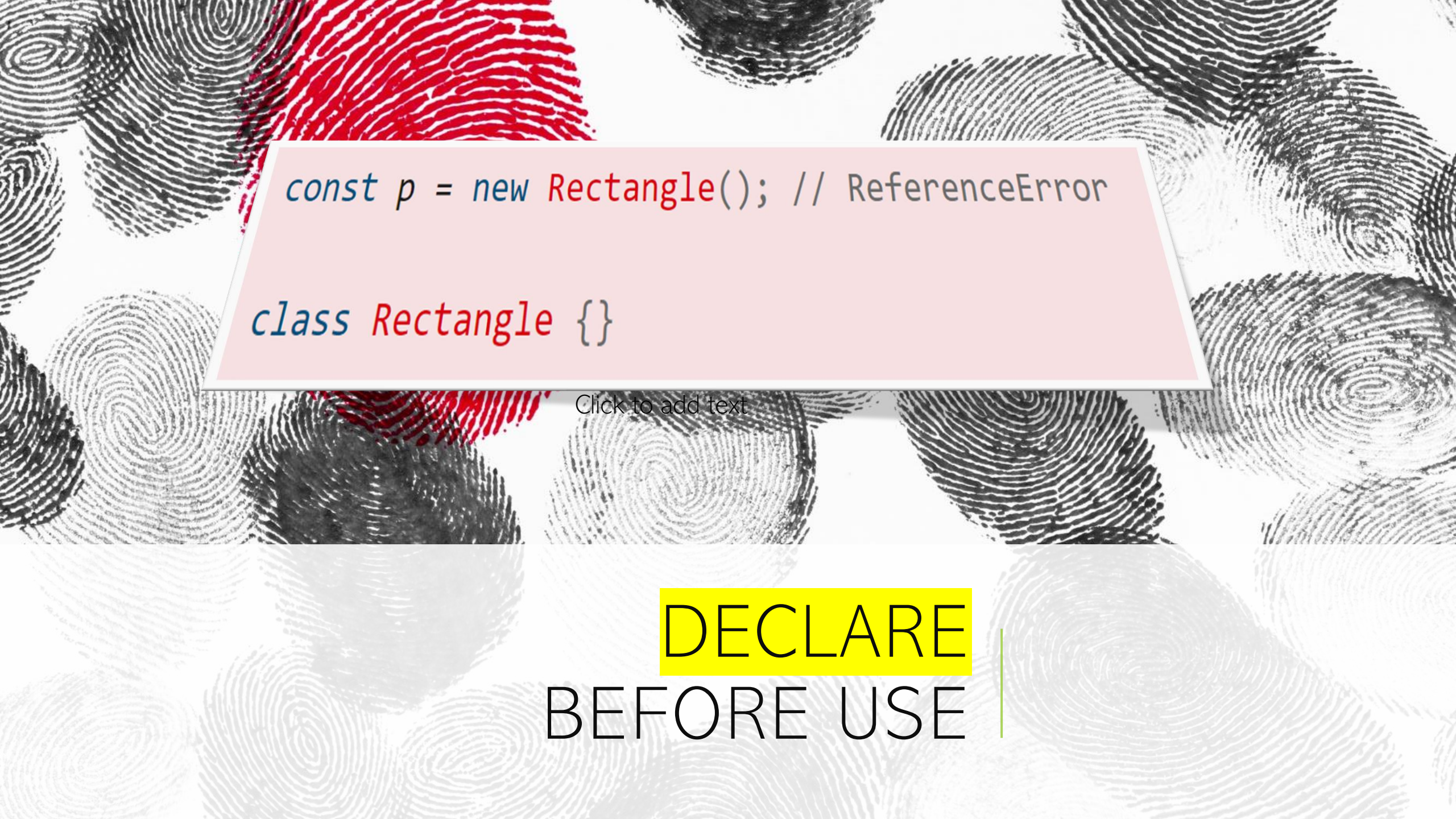
Current aligned Usage relative Date relative Filtered **All** 

Chrome	Edge [*]	Safari	Firefox	Opera
4-41				
 42-48	12	3.1-8	2-44	 10-28
49-119	13-119	9-17.1	45-120	 29-35
120	120	17.2	121	36-105
121-123	121	17.3-TP	122-124	106

You can use it
with all modern
browsers

```
class Rectangle {  
    constructor(height, width) {  
        this.height = height;  
        this.width = width;  
    }  
}
```

CLASSIC
EXAMPLE



```
const p = new Rectangle(); // ReferenceError  
  
class Rectangle {}
```

Click to add text

DECLARE
BEFORE USE

CLASS EXPRESSIONS

```
// unnamed
let Rectangle = class {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
};
console.log(Rectangle.name);
// output: "Rectangle"

// named
let Rectangle = class Rectangle2 {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
};
console.log(Rectangle.name);
// output: "Rectangle2"
```

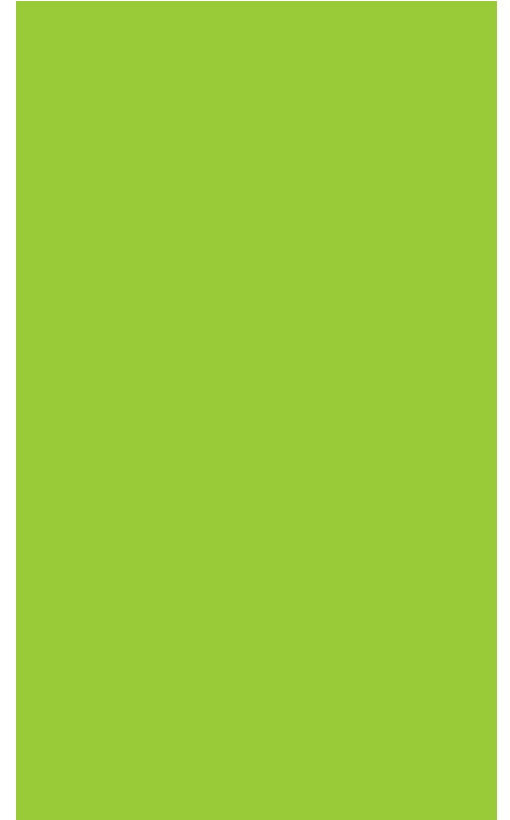
Note: Class **expressions** must be declared before they can be used

BODY OF A CLASS

```
class Rectangle {  
    constructor(height, width) {  
        this.height = height;  
        this.width = width;  
    }  
    // Getter  
    get area() {  
        return this.calcArea();  
    }  
    // Method  
    calcArea() {  
        return this.height * this.width;  
    }  
}
```

Start of class body

End of class body



A dark, atmospheric photograph of a Venetian canal. On the left, there are ornate historic buildings with arched windows and balconies. The canal is filled with water, and a small boat is visible in the distance. In the background, a large dome, likely St. Mark's Basilica, is visible under a cloudy sky. The overall tone is moody and historical.

You can declare only one
constructor function within your
class.

CONSTRUCTOR
FUNCTION

INSTANCE PROPERTIES

01 ➤ You can define properties **within the constructor**

02 ➤ By default, properties & methods **are public**

```
class Rectangle {  
    constructor(height, width) {  
        this.height = height;  
        this.width = width;  
    }  
}
```



PRIVATE: USE THE



Private instance properties are called
"hash names"

```
class Rectangle {  
  #height = 0;  
  #width;  
  
  constructor(height, width) {  
    this.#height = height;  
    this.#width = width;  
  }  
}
```



PRIVATE:

NO ACCESS OUTSIDE CLASS

```
class Human {  
  #bloodType;  
  constructor(type) {  
    this.#bloodType = type;  
  }  
}
```

// in your code someplace...

```
let me = new Human("A-Pos");  
console.info(me.#bloodType); //error: 'undefined'
```


USING ACCESSORS WITH PRIVATE ATTRIBUTES

```
class Human {  
    #bloodType;  
    constructor(type) {  
        this.#bloodType = type;  
    }  
    get typeOfBlood() {  
        return this.#bloodType;  
    }  
}
```

```
class Point {  
  constructor(x, y) {  
    this.x = x;  
    this.y = y;  
  }  
  
  static displayName = "Point";  
  static distance(a, b) {  
    const dx = a.x - b.x;  
    const dy = a.y - b.y;  
  
    return Math.hypot(dx, dy);  
  }  
}
```

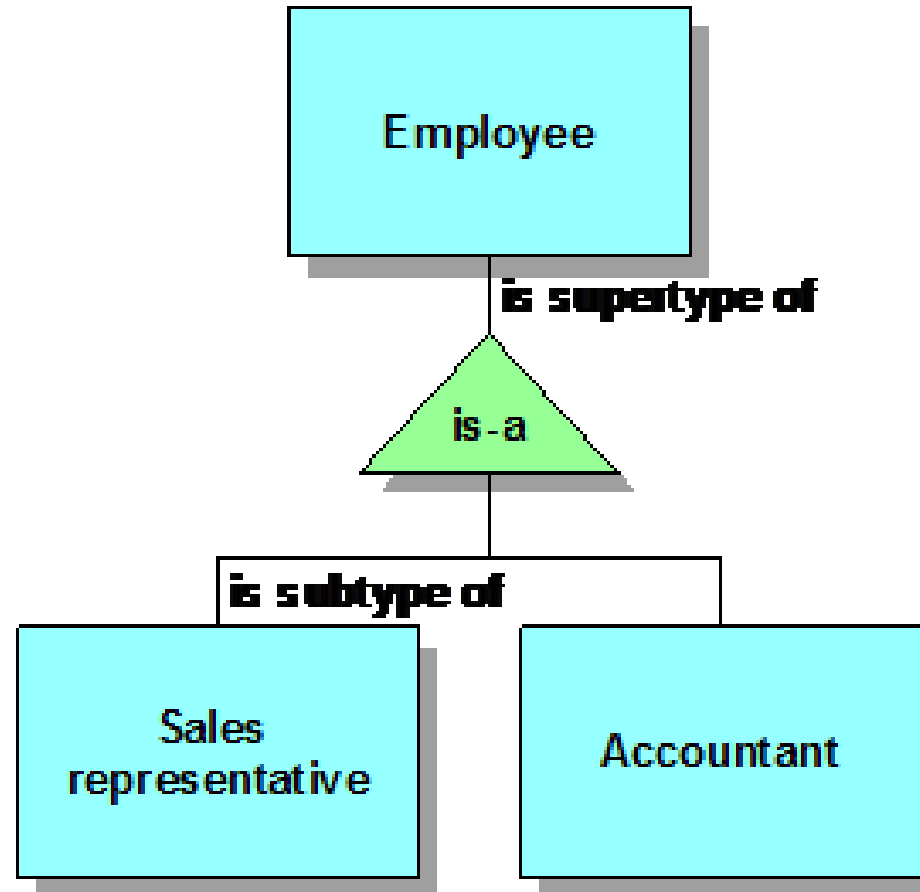
```
const p1 = new Point(5, 5);  
const p2 = new Point(10, 10);  
p1.displayName; // undefined  
p1.distance;    // undefined  
p2.displayName; // undefined  
p2.distance;    // undefined  
  
console.log(Point.displayName); // "Point"  
console.log(Point.distance(p1, p2)); // 7.0710678118654755
```

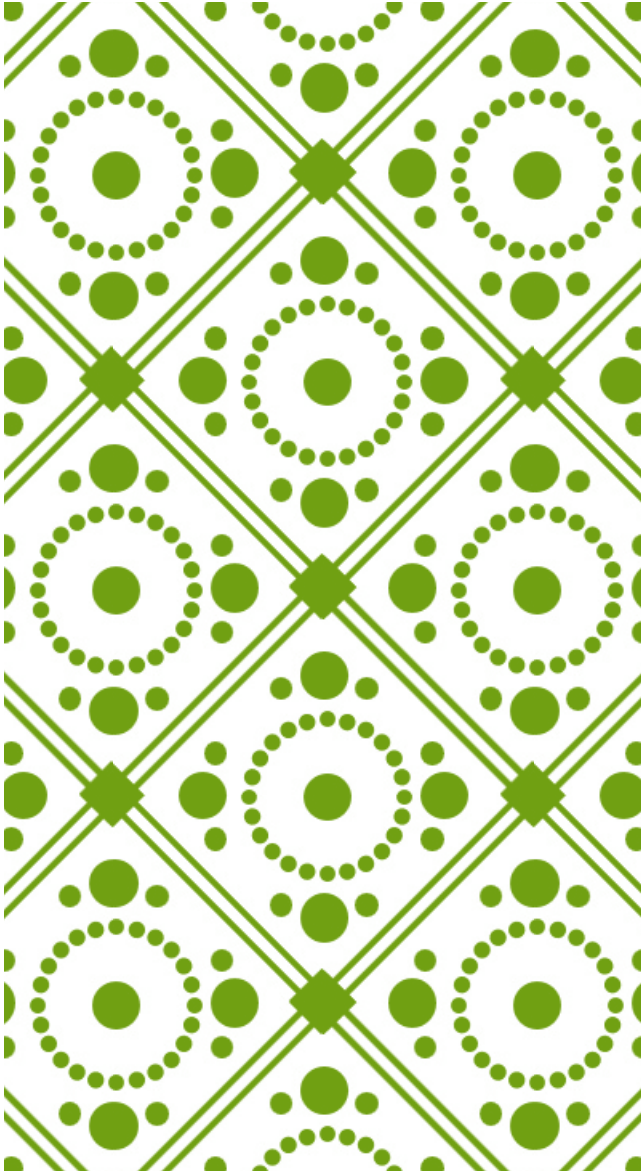
STATIC METHODS AND PROPERTIES

SUBCLASSES

GENERALIZATION

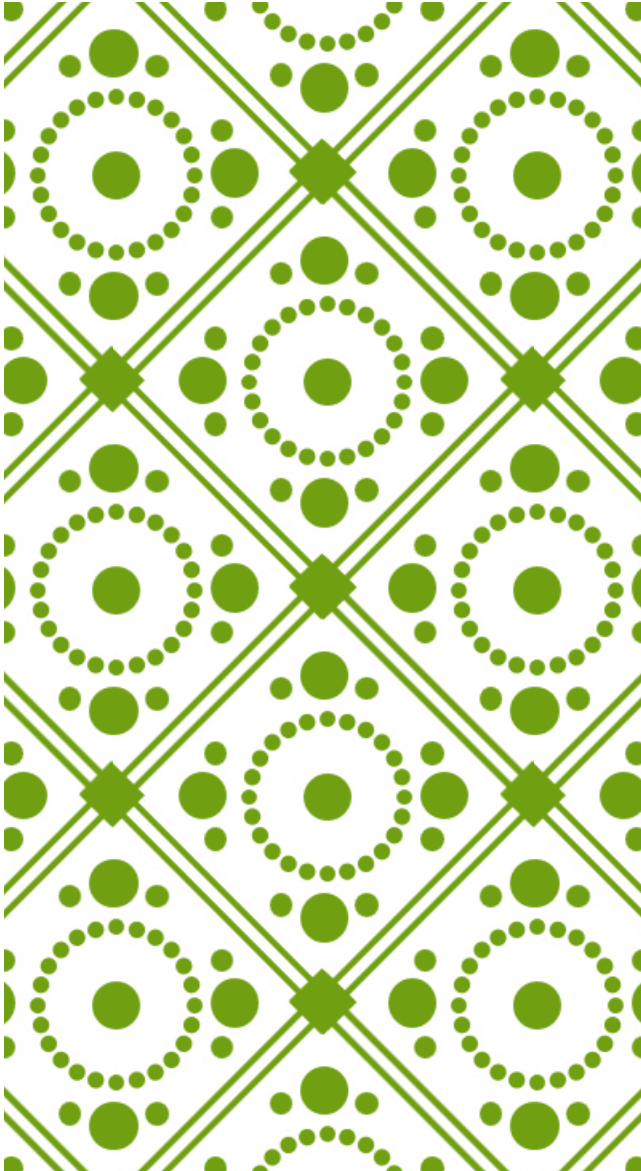
A subclass is a type of class that inherits from a base class





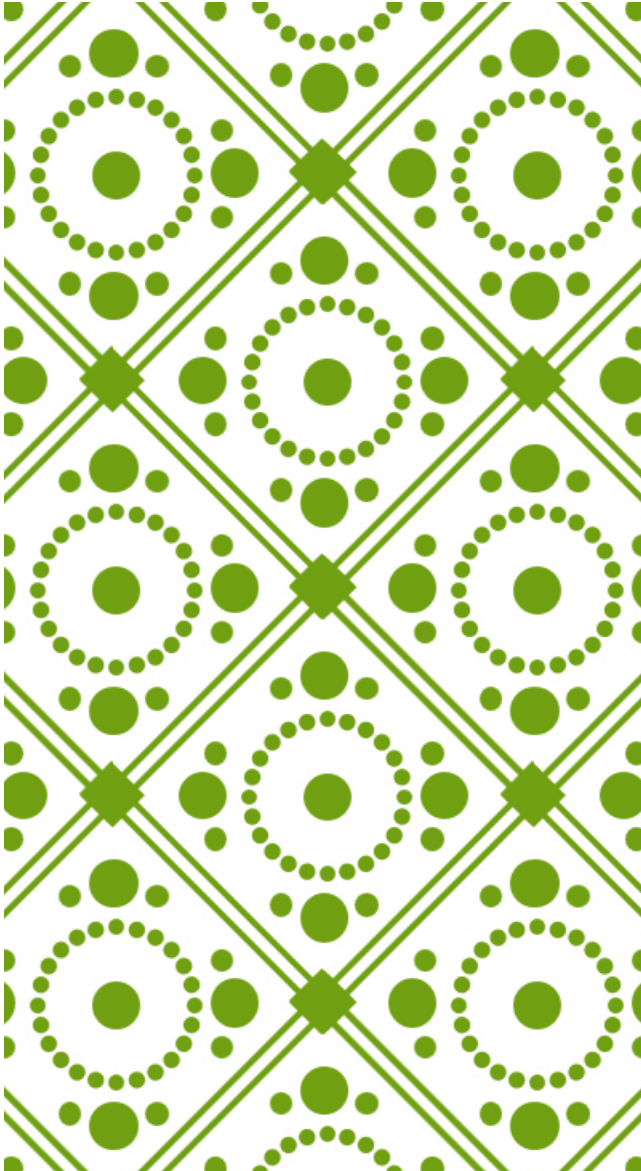
super()

Use `super()` to access your methods and attributes in your parent class.



```
class Cat {  
  constructor(name) {  
    this.name = name;  
  }  
  
  speak() {  
    console.log(`${this.name} makes a noise.`);  
  }  
}
```

A PARENT CLASS



```
class Lion extends Cat {  
  speak() {  
    super.speak();  
    console.log(`${this.name} roars.`);  
  }  
}  
  
let l = new Lion('Fuzzy');  
l.speak();  
// Fuzzy makes a noise.  
// Fuzzy roars.
```

A GENERALIZED TYPE