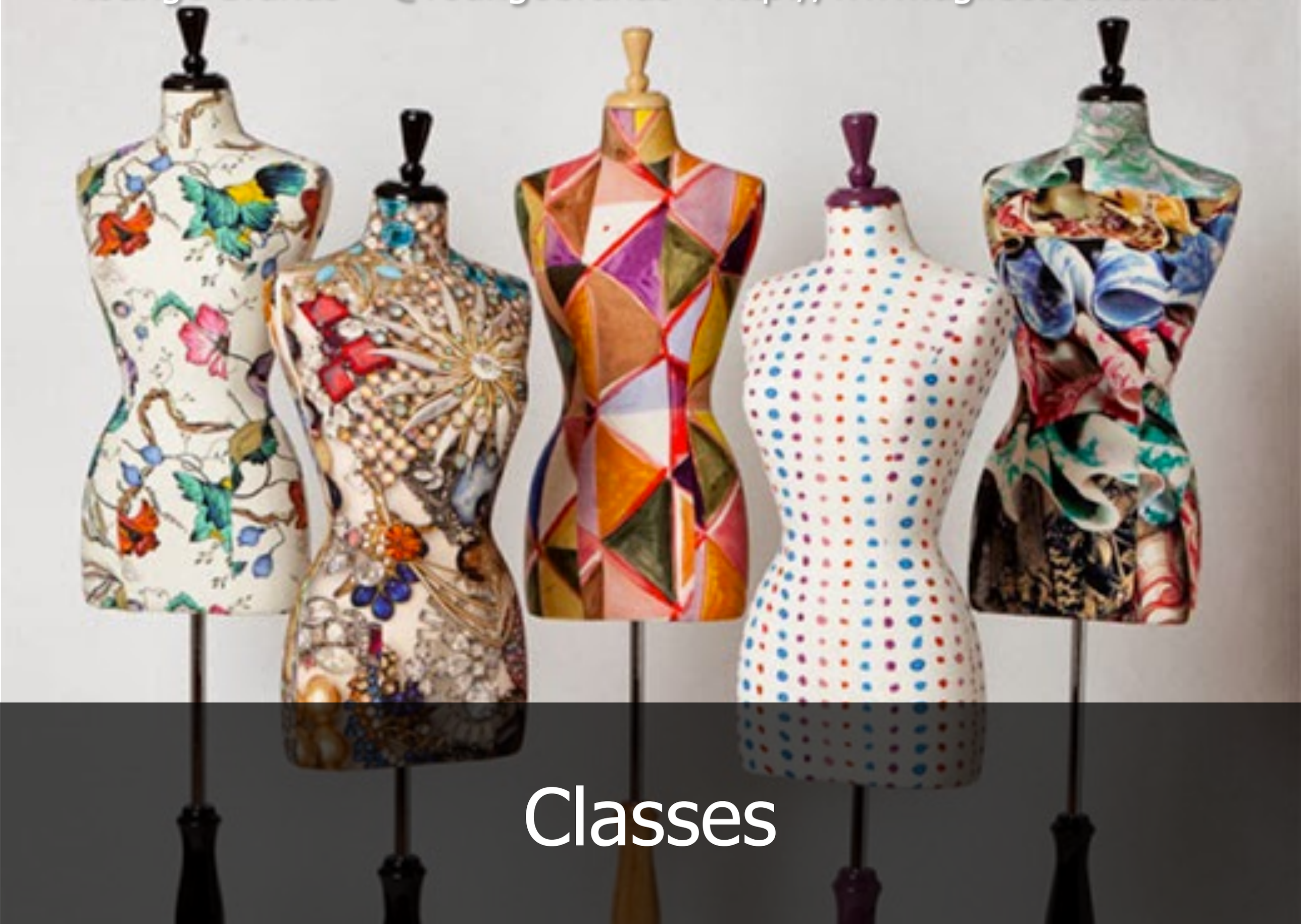

Classes

As classes funcionam da mesma forma que as funções construtoras, mas oferecem uma sintaxe mais atrativa.

```
1.    class Shape {
2.    }
```

Criando uma class

```
1.    class Shape {
2.      constructor(type) {
3.        this.type = type;
4.      }
5.    }
```

Definindo um constructor

```
1.    class Shape {
2.      constructor(type) {
3.        this.type = type;
4.      }
5.
6.      getType() {
7.        return this.type;
8.      }
9.    }
```

Definindo um método

```
1.    class Shape {
2.      constructor(type) {
3.        this.type = type;
4.      }
5.
6.      getType() {
7.        return this.type;
8.      }
9.    }
10.
11.   var triangle = new Shape("triangle");
```

Instanciando uma classe

```
1.    class Shape {
2.      constructor(type) {
3.        this.type = type;
4.      }
5.
6.      getType() {
7.        return this.type;
8.      }
9.    }
10.
11.   var triangle = new Shape("triangle");
12.   triangle.getType();
```

Invocando um método

```
1.    class Shape {
2.      constructor(type) {
3.        this.type = type;
4.      }
5.
6.      getType() {
7.        return this.type;
8.      }
9.    }
10.
11.   class Triangle extends Shape {
12.   }
```

Estendendo uma classe

```
1.      class Shape {
2.        constructor(type) {
3.          this.type = type;
4.        }
5.
6.        getType() {
7.          return this.type;
8.        }
9.      }
10.
11.     class Triangle extends Shape {
12.       constructor(a, b, c) {
13.         super("Triangle");
14.         this.a = a;
15.         this.b = b;
16.         this.c = c;
17.       }
18.     }
```

Definindo um constructor

```
1.      class Shape {
2.        constructor(type) {
3.          this.type = type;
4.        }
5.
6.        getType() {
7.          return this.type;
8.        }
9.      }
10.
11.     class Triangle extends Shape {
12.       constructor(a, b, c) {
13.         super("Triangle");
14.         this.a = a;
15.         this.b = b;
16.         this.c = c;
17.       }
18.     }
19.
20.     var triangle = new Triangle(1,2,3);
```

Invocando o constructor

```
1.      class Triangle extends Shape {
2.        constructor(a, b, c) {
3.          super("Triangle");
4.          this.a = a;
5.          this.b = b;
6.          this.c = c;
7.        }
8.
9.        get area() {
10.         return this.a * this.b * this.c;
11.       }
12.     }
13.
14.     var triangle = new Triangle(1,2,3);
```

Definindo um getter

```
1.      class Triangle extends Shape {
2.        constructor(a, b, c) {
3.          super("Triangle");
4.          this.a = a;
5.          this.b = b;
6.          this.c = c;
7.        }
8.
9.        get area() {
10.         return this.a * this.b * this.c;
11.       }
12.   }
13.
14.   var triangle = new Triangle(1,2,3);
15.   triangle.area;
```

Invocando um getter

```
1.      class Triangle extends Shape {
2.        constructor(a, b, c) {
3.          super("Triangle");
4.          this.a = a;
5.          this.b = b;
6.          this.c = c;
7.        }
8.
9.        static calculateArea(a, b, c) {
10.         return a * b * c;
11.       }
12.     }
13.
14.     Triangle.calculateArea(1,2,3);
```

Definindo um método estático