

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

```
class Shape {
```

Criando uma class

```
1.
     class Shape {
12.
       constructor(type) {
:3.
         this.type = type;
¦4.
```

Definindo um constructor

```
11.
     class Shape {
       constructor(type) {
12.
3.
        this.type = type;
¦4.
5.
6.
       getType() {
        return this.type;
8.
```

Definindo um método

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

Instanciando uma classe

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

Invocando um método

```
class Shape {
¦1.
2.
3.
        constructor(type) {
          this.type = type;
¦4.
5.
¦6.
        getType() {
<del>1</del>7.
          return this.type;
¦8.
¦9.
10.
111.
       class Triangle extends Shape {
12.
```

Estendendo uma classe

```
¦1.
        class Shape {
2.
3.
          constructor(type) {
           this.type = type;
4.
¦5.
¦6.
          getType() {
           return this.type;
8.
¦9.
10.
111.
        class Triangle extends Shape {
12.
          constructor(a, b, c) {
13.
           super("Triangle");
          this.a = a;
14.
15.
          this.b = b;
<u>!</u>16.
           this.c = c;
!17.
¦18.
```

Definindo um constructor

```
1.
        class Shape {
         constructor(type) {
13.
           this.type = type;
4.
¦5.
6.
         getType() {
           return this.type;
8.
9.
10.
        class Triangle extends Shape {
111.
12.
         constructor(a, b, c) {
           super("Triangle");
<u>!</u>13.
           this.a = a;
114.
           this.b = b;
115.
           this.c = c;
16.
i 17.
18.
19.
20.
        var triangle = new Triangle(1,2,3);
```

Invocando o constructor

```
class Triangle extends Shape {
¦1.
|2.
|3.
|4.
         constructor(a, b, c) {
          super("Triangle");
          this.a = a;
¦5.
          this.b = b;
¦6.
          this.c = c;
¦7.
8.
9.
         get area() {
10.
          return this.a * this.b * this.c;
111.
12.
13.
14.
       var triangle = new Triangle(1,2,3);
```

Definindo um getter

```
class Triangle extends Shape {
¦1.
2.
3.
4.
         constructor(a, b, c) {
           super("Triangle");
           this.a = a;
¦5.
          this.b = b;
¦6.
           this.c = c;
8.
9.
         get area() {
10.
           return this.a * this.b * this.c;
111.
12.
13.
14.
        var triangle = new Triangle(1,2,3);
        triangle.area;
<u>!</u>15.
```

Invocando um getter

```
class Triangle extends Shape {
¦1.
2.
3.
4.
         constructor(a, b, c) {
          super("Triangle");
          this.a = a;
¦5.
          this.b = b;
¦6.
          this.c = c;
8.
¦9.
         static calculateArea(a, b, c) {
10.
          return a * b * c;
111.
12.
13.
14.
       Triangle.calculateArea(1,2,3);
```

Definindo um método estático