

A close-up photograph of a person's hands and torso. The person is wearing a light-colored ribbed sweater over a tan apron. They are holding a small, dark rectangular tray with a metal frame. Their left hand, which has a ring with a pink stone, is pointing towards the tray. The background is dark and out of focus.

Classes

As classes são um tipo especial de função  
que atuam como um template para a  
criação de objetos

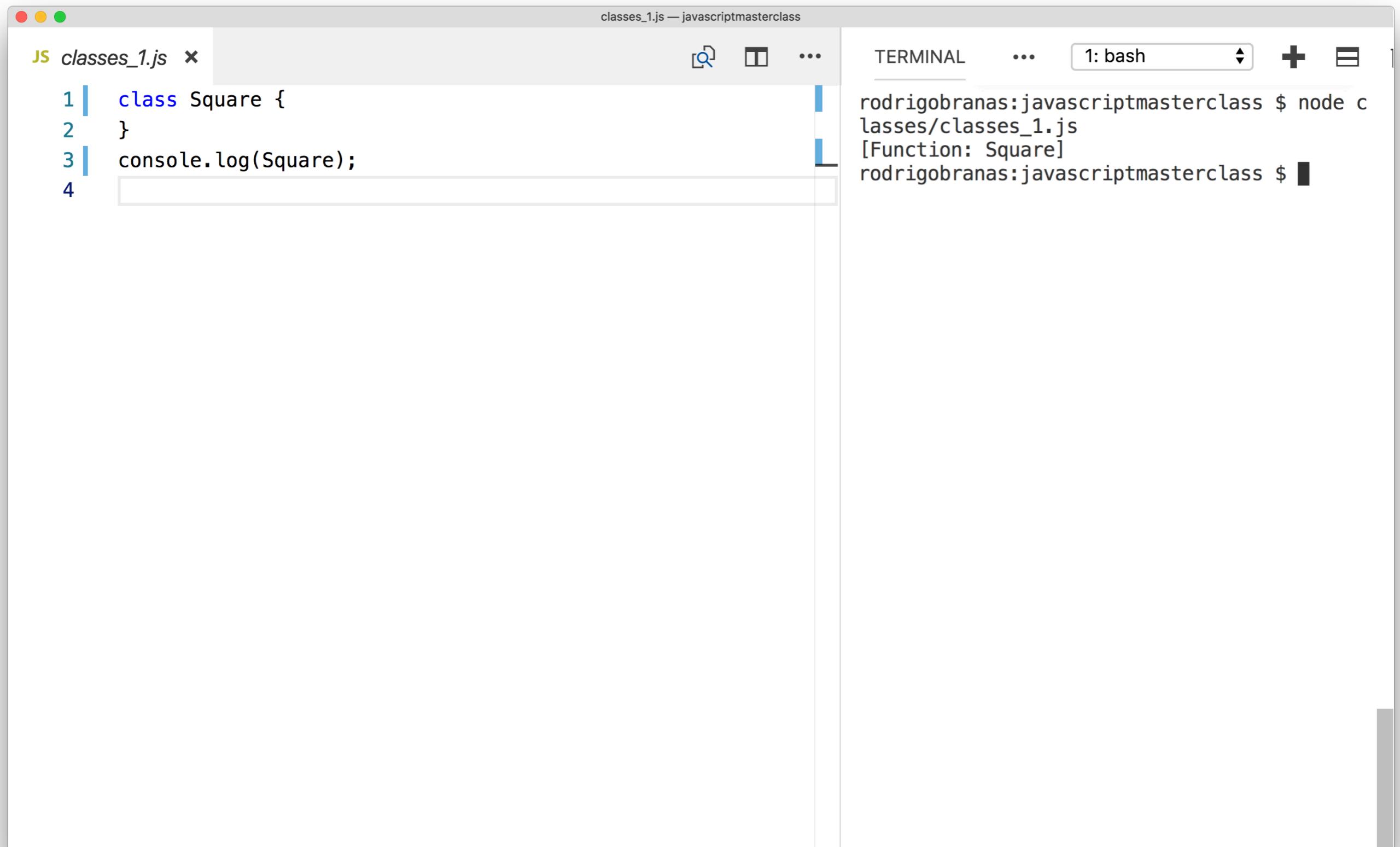
classes\_1.js — javascriptmasterclass

JS classes\_1.js x

1 | class Square {  
2 | }  
3 | console.log(Square);  
4 |

TERMINAL ... 1: bash

rodrigobranas:javascriptmasterclass \$ node classes/classes\_1.js  
[Function: Square]  
rodrigobranas:javascriptmasterclass \$



A screenshot of a macOS desktop environment. On the left is a code editor window titled "classes\_2.js — javascriptmasterclass". The editor contains the following JavaScript code:

```
JS classes_2.js x
1 const Square = class {
2 }
3 console.log(Square);
4
```

The code defines a class named "Square" and logs its function object to the console. On the right is a terminal window titled "TERMINAL" with the identifier "1: bash". The terminal shows the command \$ node classes/classes\_2.js being run, followed by the output [Function: Square].

classes\_3.js — javascriptmasterclass

JS classes\_3.js x

1 | const Square = class {  
2 | }  
3 | console.log(typeof Square);  
4 |

TERMINAL ... 1: bash

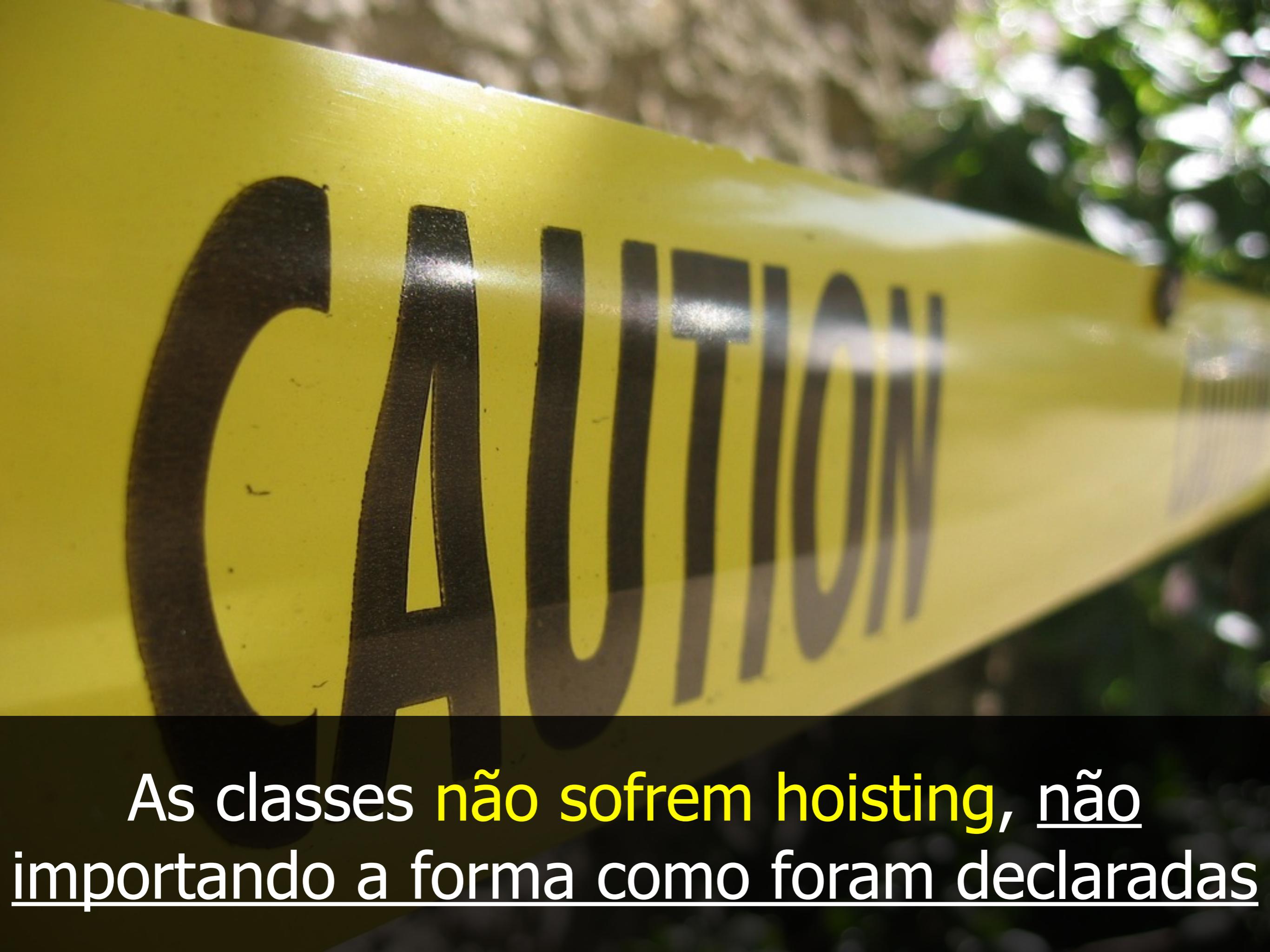
rodrigobranas:javascriptmasterclass \$ node c  
lasses/classes\_3.js  
function  
rodrigobranas:javascriptmasterclass \$

This screenshot shows a macOS desktop environment with a terminal window open. The terminal window has a title bar 'classes\_3.js — javascriptmasterclass'. On the left, there's a tab labeled 'JS classes\_3.js x' with a close button. The main area contains four lines of code: 'const Square = class {}', 'console.log(typeof Square);', and two blank lines. To the right of the code is a terminal pane with a search icon, a refresh icon, and a three-dot menu icon. Below these icons is a vertical toolbar with a magnifying glass, a square, and a three-dot menu. The terminal pane itself shows the command 'node classes/classes\_3.js' being run, followed by the output 'function', and then the prompt '\$' again.

A screenshot of a macOS desktop environment showing a terminal window. The terminal window has a title bar "classes\_4.js — javascriptmasterclass". The main pane displays the following text:

```
rodrigobranas:javascriptmasterclass $ node classes/classes_4.js
Square {}
rodrigobranas:javascriptmasterclass $
```

The terminal window includes standard macOS-style controls at the top right: a magnifying glass icon, a refresh icon, a three-dot menu icon, and a close button.

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is slightly curved, and the background shows some green foliage.

CAUTION

As classes não sofrem hoisting, não  
importando a forma como foram declaradas

A screenshot of a macOS terminal window titled "classes\_5.js — javascriptmasterclass". The window has three panes: a code editor on the left, a terminal on the right, and a status bar at the bottom.

The code editor pane shows the following JavaScript code:

```
JS classes_5.js x
1 const square = new Square();
2 const Square = class {
3 }
4 console.log(square);
5
```

The terminal pane shows the output of running the script with Node.js:

```
rodrigobranas:javascriptmasterclass $ node classes/classes_5.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_5.js:1
(function (exports, require, module, __filename, __dirname) { const square = new Square();
^
ReferenceError: Square is not defined
    at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_5.js:1:78)
        at Module._compile (module.js:643:30)
        at Object.Module._extensions..js (module.js:654:10)
        at Module.load (module.js:556:32)
        at tryModuleLoad (module.js:499:12)
        at Function.Module._load (module.js:491:3)
        at Function.Module.runMain (module.js:684:10)
        at startup (bootstrap_node.js:187:16)
        at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```

A screenshot of a terminal window titled "classes\_6.js — javascriptmasterclass". The window has tabs for "classes\_6.js" and "TERMINAL". The code editor shows the following JavaScript code:

```
JS classes_6.js x
1 const square = new Square();
2 class Square {
3 }
4 console.log(square);
5
```

The terminal pane displays the output of running the script with "node". It shows the command being run, the file path, and the resulting error message:

```
rodrigobranas:javascriptmasterclass $ node classes/classes_6.js
/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_6.js:1
(function (exports, require, module, __filename, __dirname) { const square = new Square();
^
ReferenceError: Square is not defined
    at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_6.js:1:78)
        at Module._compile (module.js:643:30)
        at Object.Module._extensions..js (module.js:654:10)
        at Module.load (module.js:556:32)
        at tryModuleLoad (module.js:499:12)
        at Function.Module._load (module.js:491:3)
        at Function.Module.runMain (module.js:684:10)
        at startup (bootstrap_node.js:187:16)
        at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```

As classes são formadas por 3 tipos de membros: **constructor**, **prototype methods** e **static methods**

O **constructor** é invocado no momento da instânciação de uma classe e serve para inicializar um determinado objeto

classes\_7.js — javascriptmasterclass

JS classes\_7.js ×

```
1 class Square {  
2     constructor() {  
3     }  
4 }  
5 const square = new Square();  
6 console.log(square);  
7
```

TERMINAL ... 1: bash

```
rodrigobranas:javascriptmasterclass $ node c  
lasses/classes_7.js  
Square {}  
rodrigobranas:javascriptmasterclass $
```

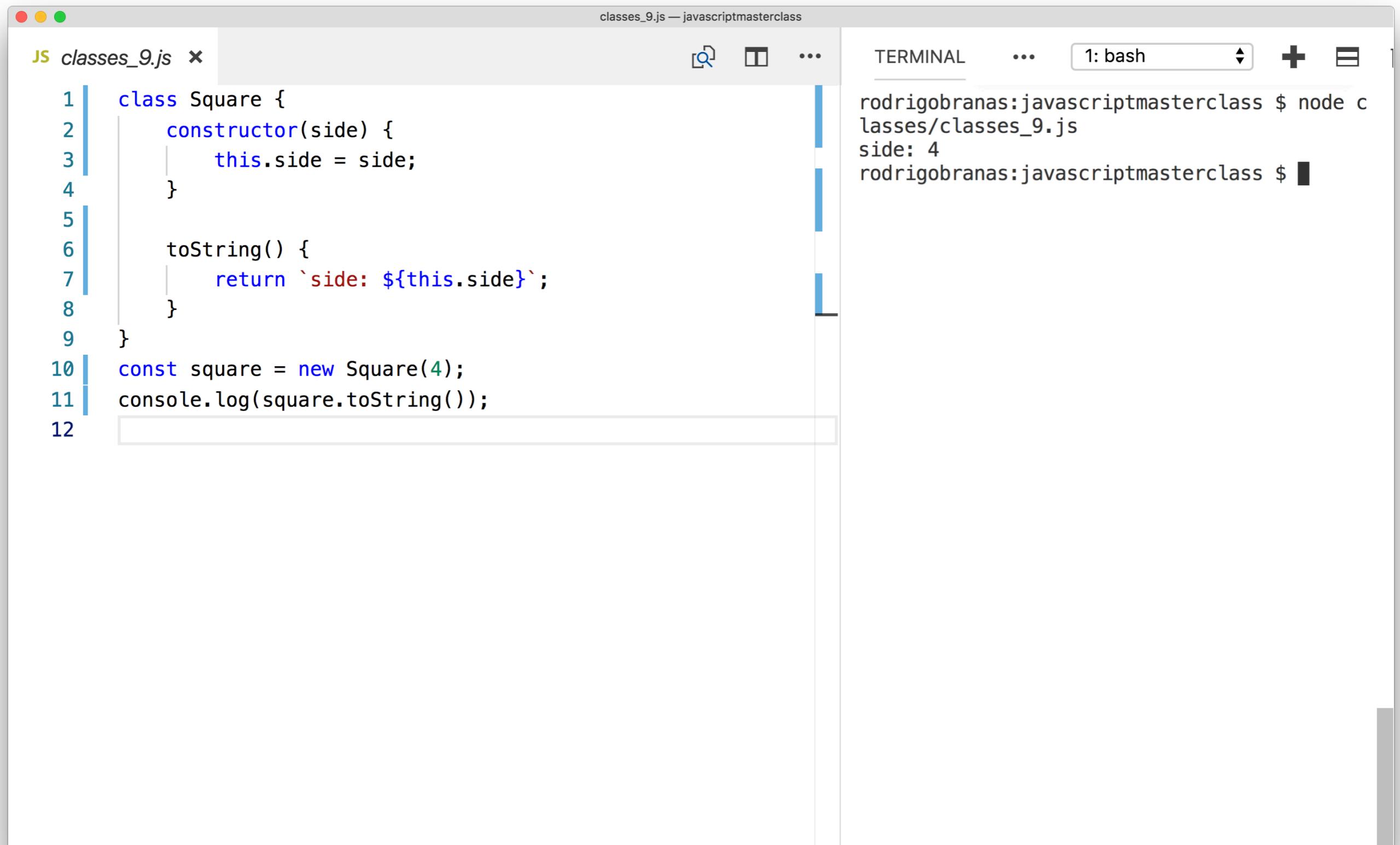
A screenshot of a macOS desktop environment. On the left is a code editor window titled "classes\_8.js — javascriptmasterclass". The code editor contains the following JavaScript code:

```
JS classes_8.js x
1 class Square {
2     constructor(side) {
3         this.side = side;
4     }
5 }
6 const square = new Square(4);
7 console.log(square);
8
```

The code defines a class named "Square" with a constructor that takes a parameter "side" and assigns it to the "side" property of the instance. It then creates a variable "square" and logs its value to the console.

To the right of the code editor is a terminal window titled "TERMINAL" with the identifier "1: bash". The terminal shows the command \$ node classes/classes\_8.js being run, followed by the output: Square { side: 4 }. The terminal prompt is rodrigobranas:javascriptmasterclass \$.

Os **prototype methods** dependem de uma  
instância para serem invocados



A screenshot of a Mac OS X desktop environment showing a terminal window and a code editor. The terminal window is titled 'classes\_9.js — javascriptmasterclass' and contains the command 'node classes/classes\_9.js' followed by the output 'side: 4'. The code editor window shows the file 'classes\_9.js' with the following content:

```
JS classes_9.js x
1 class Square {
2     constructor(side) {
3         this.side = side;
4     }
5
6     toString() {
7         return `side: ${this.side}`;
8     }
9 }
10 const square = new Square(4);
11 console.log(square.toString());
12
```

A screenshot of a terminal window titled "classes\_10.js — javascriptmasterclass". The terminal shows the command \$ node classes/classes\_10.js being run, followed by the output side: 4 area: 16.

```
JS classes_10.js x classes_10.js — javascriptmasterclass TERMINAL ... 1: bash + = rodrigobranas:javascriptmasterclass $ node classes/classes_10.js side: 4 area: 16 rodrigobranas:javascriptmasterclass $
```

```
1 class Square {
2     constructor(side) {
3         this.side = side;
4     }
5
6     calculateArea() {
7         return Math.pow(this.side, 2);
8     }
9
10    toString() {
11        return `side: ${this.side} area: ${this.calculateAre
12    }
13 }
14 const square = new Square(4);
15 console.log(square.toString());
16
```

Os static methods não dependem de uma  
instância para serem invocados

A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled 'classes\_11.js — javascriptmasterclass' and contains the command 'node classes/classes\_11.js'. The output shows the execution of the JavaScript code, which defines a class 'Square' with methods for calculating area and returning a string representation. A static method 'fromArea' is also defined to create a square from its area. The code is run in a bash terminal.

```
JS classes_11.js × classes_11.js — javascriptmasterclass TERMINAL ... 1: bash + ⌂ 1  
rodrigobranas:javascriptmasterclass $ node classes/classes_11.js  
side: 4 area: 16  
rodrigobranas:javascriptmasterclass $
```

```
1  class Square {  
2      constructor(side) {  
3          this.side = side;  
4      }  
5  
6      calculateArea() {  
7          return Math.pow(this.side, 2);  
8      }  
9  
10     toString() {  
11         return `side: ${this.side} area: ${this.calculateAre  
12     }  
13  
14     static fromArea(area) {  
15         return new Square(Math.sqrt(area));  
16     }  
17 }  
18 const square = Square.fromArea(16);  
19 console.log(square.toString());  
20
```

As classes funcionam de forma similar as  
funções construtoras

classes\_12.js — javascriptmasterclass

JS classes\_12.js x

TERMINAL ... 1: bash + =

```
function Square(side) {
  this.side = side;
}

Square.prototype.calculateArea = function() {
  return Math.pow(this.side, 2);
}

Square.prototype.toString = function() {
  return `side: ${this.side} area: ${this.calculateArea()}`;
}

Square.fromArea = function(area) {
  return new Square(Math.sqrt(area));
}

const square = Square.fromArea(16);
console.log(square.toString());
```

rodrigobranas:javascriptmasterclass \$ node classes/classes\_12.js  
side: 4 area: 16  
rodrigobranas:javascriptmasterclass \$

É possível criar uma hierarquia de classes  
por meio da palavra-chave **extends**

```
classes_13.js — javascriptmasterclass
JS classes_13.js ×
1 class Square {
2     constructor(side) {
3         this.side = side;
4     }
5
6     calculateArea() {
7         return Math.pow(this.side, 2);
8     }
9
10    toString() {
11        return `side: ${this.side} area: ${this.calculateArea()}`;
12    }
13
14    static fromArea(area) {
15        return new Square(Math.sqrt(area));
16    }
17}
18const square = Square.fromArea(16);
19console.log(square.toString());
20
21class Circle {
22    constructor(radius) {
23        this.radius = radius;
24    }
25    calculateArea() {
26        return Math.PI * this.radius * this.radius;
27    }
28}
```

TERMINAL ... 1: bash

```
rodrigobranas:javascriptmasterclass $ node classes/classes_13.js
side: 4 area: 16
radius: 10 area: 314.1592653589793
rodrigobranas:javascriptmasterclass $
```

Ao declarar um construtor na subclass é necessário **invocar o construtor da superclass** por meio **super()** antes de utilizar a referência this

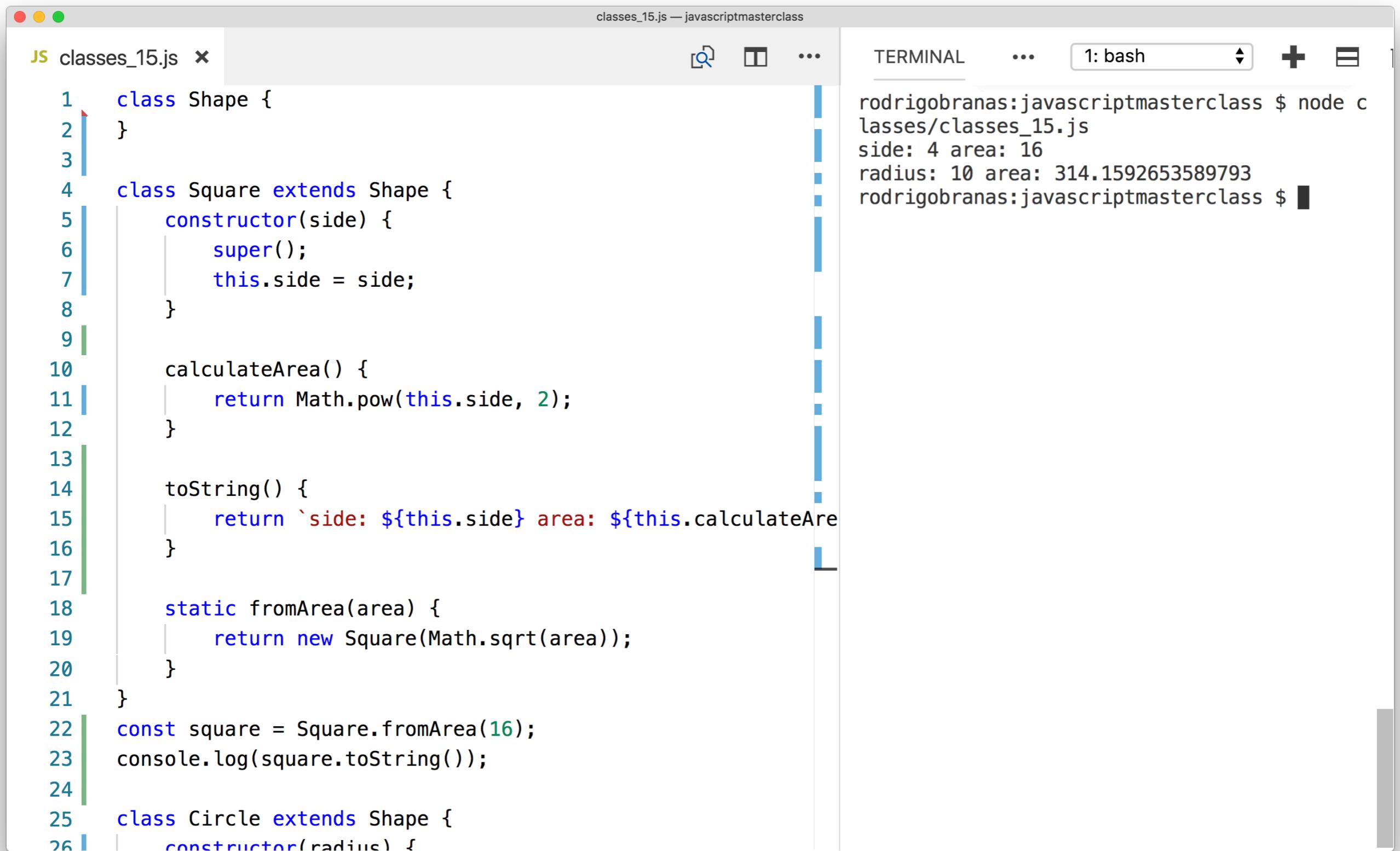
classes\_14.js — javascriptmasterclass

JS classes\_14.js x

```
1  class Shape {  
2  }  
3  
4  class Square extends Shape {  
5      constructor(side) {  
6          this.side = side;  
7      }  
8  
9      calculateArea() {  
10         return Math.pow(this.side, 2);  
11     }  
12  
13     toString() {  
14         return `side: ${this.side} area: ${this.calculateArea()}`;  
15     }  
16  
17     static fromArea(area) {  
18         return new Square(Math.sqrt(area));  
19     }  
20 }  
21 const square = Square.fromArea(16);  
22 console.log(square.toString());  
23  
24 class Circle extends Shape {  
25     constructor(radius) {  
26         this.radius = radius;  
27     }  
28     calculateArea() {  
29         return Math.PI * this.radius * this.radius;  
30     }  
31     toString() {  
32         return `radius: ${this.radius} area: ${this.calculateArea()}`;  
33     }  
34 }  
35  
36 const circle = Circle.fromArea(16);  
37 console.log(circle.toString());
```

TERMINAL ... 1: bash

```
rodrigobranas:javascriptmasterclass $ node classes/classes_14.js  
/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_14.js:6  
    this.side = side;  
^  
  
ReferenceError: Must call super constructor  
in derived class before accessing 'this' or  
returning from derived constructor  
    at new Square (/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_14.js:6:9)  
        at Function.fromArea (/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_14.js:18:16)  
            at Object.<anonymous> (/Users/rodrigobranas/development/workspace/javascriptmasterclass/classes/classes_14.js:21:23)  
                at Module._compile (module.js:643:30)  
                    at Object.Module._extensions..js (module.js:654:10)  
                        at Module.load (module.js:556:32)  
                            at tryModuleLoad (module.js:499:12)  
                                at Function.Module._load (module.js:491:3)  
                                    at Function.Module.runMain (module.js:684:10)  
                                        at startup (bootstrap_node.js:187:16)  
rodrigobranas:javascriptmasterclass $
```



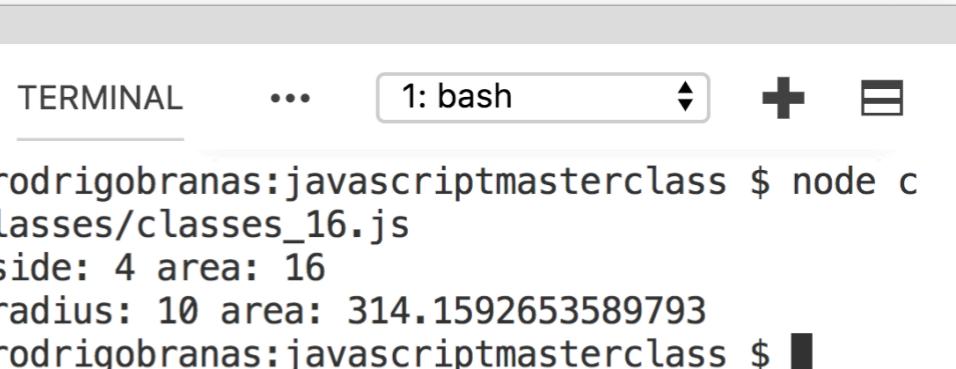
The screenshot shows a terminal window with the following output:

```
rodrigobranas:javascriptmasterclass $ node classes/classes_15.js
side: 4 area: 16
radius: 10 area: 314.1592653589793
rodrigobranas:javascriptmasterclass $
```

The code in the file classes\_15.js defines a Shape class and a Square class that extends it. It also includes a static fromArea method and a console.log statement.

```
JS classes_15.js x
classes_15.js — javascriptmasterclass
TERMINAL ... 1: bash + ⌂ 1

1  class Shape {
2  }
3
4  class Square extends Shape {
5      constructor(side) {
6          super();
7          this.side = side;
8      }
9
10     calculateArea() {
11         return Math.pow(this.side, 2);
12     }
13
14     toString() {
15         return `side: ${this.side} area: ${this.calculateAre
16     }
17
18     static fromArea(area) {
19         return new Square(Math.sqrt(area));
20     }
21 }
22 const square = Square.fromArea(16);
23 console.log(square.toString());
24
25 class Circle extends Shape {
26     constructor(radius) {
```



A screenshot of a macOS desktop environment showing a terminal window and a code editor. The terminal window is titled '1: bash' and contains the following text:

```
rodrigobranas:javascriptmasterclass $ node classes/classes_16.js
side: 4 area: 16
radius: 10 area: 314.1592653589793
rodrigobranas:javascriptmasterclass $
```

The code editor window is titled 'classes\_16.js — javascriptmasterclass'. It displays the following JavaScript code:

```
1  class Shape {
2      toString() {
3          return `area: ${this.calculateArea()}`;
4      }
5  }
6
7  class Square extends Shape {
8      constructor(side) {
9          super();
10         this.side = side;
11     }
12
13     calculateArea() {
14         return Math.pow(this.side, 2);
15     }
16
17     toString() {
18         return `side: ${this.side} ${super.toString()}`;
19     }
20
21     static fromArea(area) {
22         return new Square(Math.sqrt(area));
23     }
24 }
25 const square = Square.fromArea(16);
26 console.log(square.toString());
```