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

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

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_1.js
[Function: Square]
rodrigobranas:javascriptmasterclass $
```

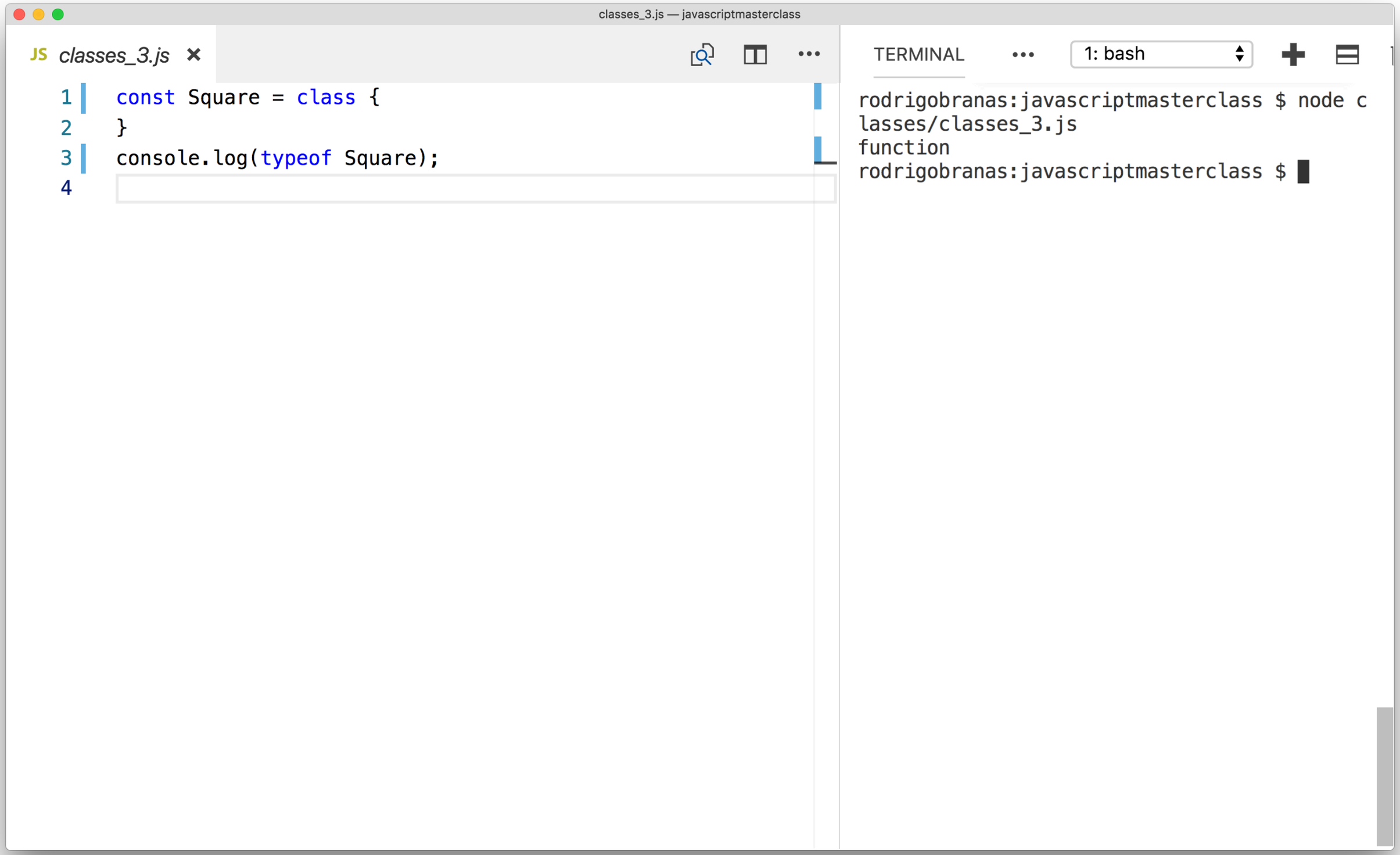
classes_2.js — javascriptmasterclass

JS classes_2.js

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

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_2.js
[Function: Square]
rodrigobranas:javascriptmasterclass $
```


classes_3.js — javascriptmasterclass
classes_3.js
const Square = class {
}
console.log(typeof Square);
TERMINAL
1: bash
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_3.js
function
rodrigobranas:javascriptmasterclass $

classes_4.js — javascriptmasterclass

JS classes_4.js

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

TERMINAL 1: bash

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

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

classes_5.js — javascriptmasterclass

JS classes_5.js

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

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_5.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/classes/classes_5.js:1
(function (exports, require, module, __filen
ame, __dirname) { const square = new Square(
);
                ^

ReferenceError: Square is not defined
    at Object.<anonymous> (/Users/rodrigobra
nas/development/workspace/javascriptmastercl
ass/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:68
4:10)
    at startup (bootstrap_node.js:187:16)
    at bootstrap_node.js:608:3
rodrigobranas:javascriptmasterclass $
```

classes_6.js — javascriptmasterclass

classes_6.js

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

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_6.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/classes/classes_6.js:1
(function (exports, require, module, __filen
ame, __dirname) { const square = new Square(
);
                ^

ReferenceError: Square is not defined
    at Object.<anonymous> (/Users/rodrigobra
nas/development/workspace/javascriptmastercl
ass/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:68
4:10)
    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 instanciação 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 $
```

classes_8.js — javascriptmasterclass

JS classes_8.js

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

TERMINAL ••• 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_8.js
Square { side: 4 }
rodrigobranas:javascriptmasterclass $
```

Os prototype methods dependem de uma instância para serem invocados

classes_9.js — javascriptmasterclass

JS classes_9.js

```
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
```

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_9.js
side: 4
rodrigobranas:javascriptmasterclass $
```

classes_10.js — javascriptmasterclass

JS classes_10.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.calculateAre
12     }
13 }
14 const square = new Square(4);
15 console.log(square.toString());
16
```

TERMINAL ••• 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_10.js
side: 4 area: 16
rodrigobranas:javascriptmasterclass $
```

Os static methods não dependem de uma instância para serem invocados

classes_11.js — javascriptmasterclass

JS classes_11.js

```
class Square {
    constructor(side) {
        this.side = side;
    }

    calculateArea() {
        return Math.pow(this.side, 2);
    }

    toString() {
        return `side: ${this.side} area: ${this.calculateAre
    }

    static fromArea(area) {
        return new Square(Math.sqrt(area));
    }
}
const square = Square.fromArea(16);
console.log(square.toString());
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_11.js
side: 4 area: 16
rodrigobranas:javascriptmasterclass $
```

As classes funcionam de forma similar as funções construtoras

classes_12.js — javascriptmasterclass

classes_12.js

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

TERMINAL

1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/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

```
class Square {
    constructor(side) {
        this.side = side;
    }

    calculateArea() {
        return Math.pow(this.side, 2);
    }

    toString() {
        return `side: ${this.side} area: ${this.calculateAre
    }

    static fromArea(area) {
        return new Square(Math.sqrt(area));
    }
}
const square = Square.fromArea(16);
console.log(square.toString());

class Circle {
    constructor(radius) {
        this.radius = radius;
    }

    calculateArea() {
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/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

```
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.calculateAre
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;
```

TERMINAL 1: bash

```
rodrigobranas:javascriptmasterclass $ node c
lasses/classes_14.js
/Users/rodrigobranas/development/workspace/j
avascriptmasterclass/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/deve
lopment/workspace/javascriptmasterclass/clas
ses/classes_14.js:6:9)
    at Function.fromArea (/Users/rodrigobran
as/development/workspace/javascriptmastercla
ss/classes/classes_14.js:18:16)
    at Object.<anonymous> (/Users/rodrigobra
nas/development/workspace/javascriptmastercl
ass/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:68
4:10)
    at startup (bootstrap_node.js:187:16)
rodrigobranas:javascriptmasterclass $
```

classes_15.js — javascriptmasterclass

JS classes_15.js

```
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) {
```

TERMINAL

1: bash

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

classes_16.js — javascriptmasterclass

JS classes_16.js

```
class Shape {
    toString() {
        return `area: ${this.calculateArea()}`;
    }
}

class Square extends Shape {
    constructor(side) {
        super();
        this.side = side;
    }

    calculateArea() {
        return Math.pow(this.side, 2);
    }

    toString() {
        return `side: ${this.side} ${super.toString()}`;
    }

    static fromArea(area) {
        return new Square(Math.sqrt(area));
    }
}
const square = Square.fromArea(16);
console.log(square.toString());
```

TERMINAL 1: bash

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