O Javascript

que você nunca viu

quem sou eu_

lead solutions architect_

Nextcode

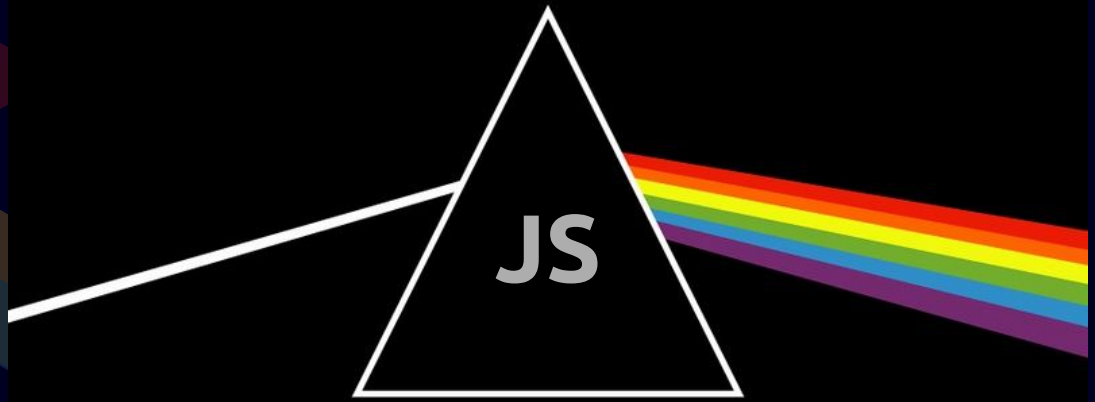
MVP Microsoft® Most Valuable Professional

lsantos.dev

O Javascripto <3

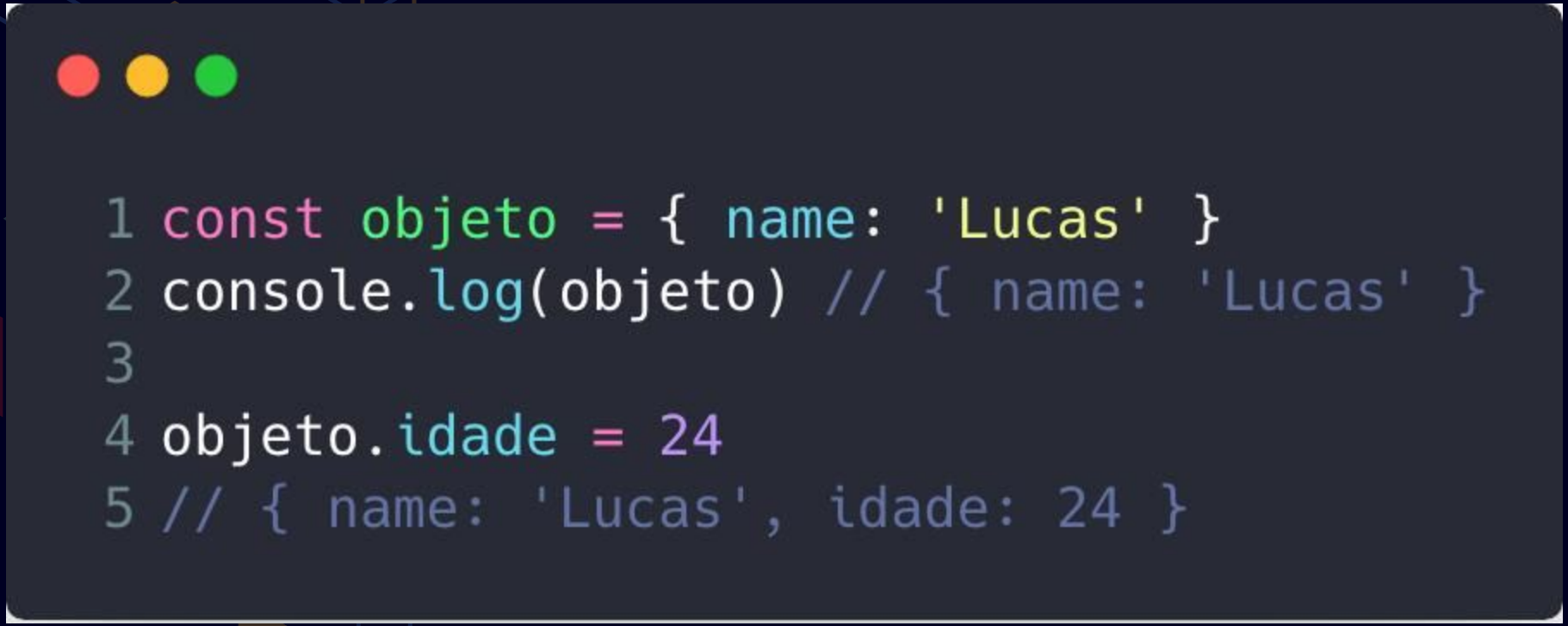
Removendo Callbacks

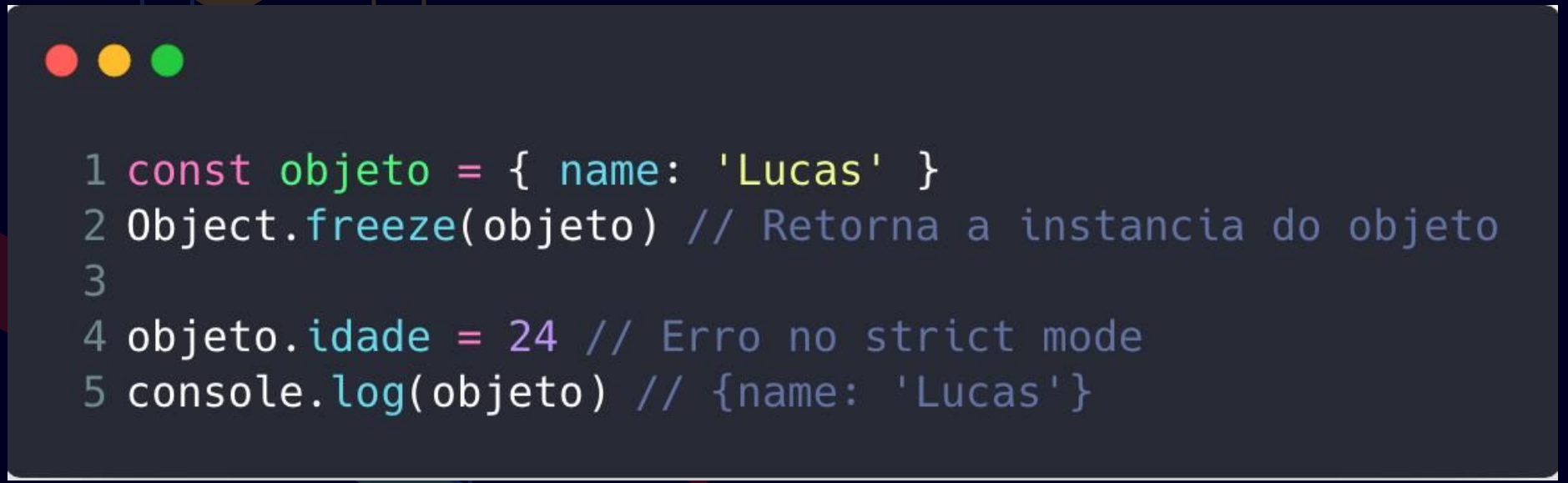
```
1 function comCallback (param, cb) {
2     const x = param
3     return cb(x)
4 }
5
6 function promisified (param) {
7     return new Promise ((resolve, reject) => {
8         return comCallback(param, resolve)
9     })
10 }
```



The Dark Side of
Javascript

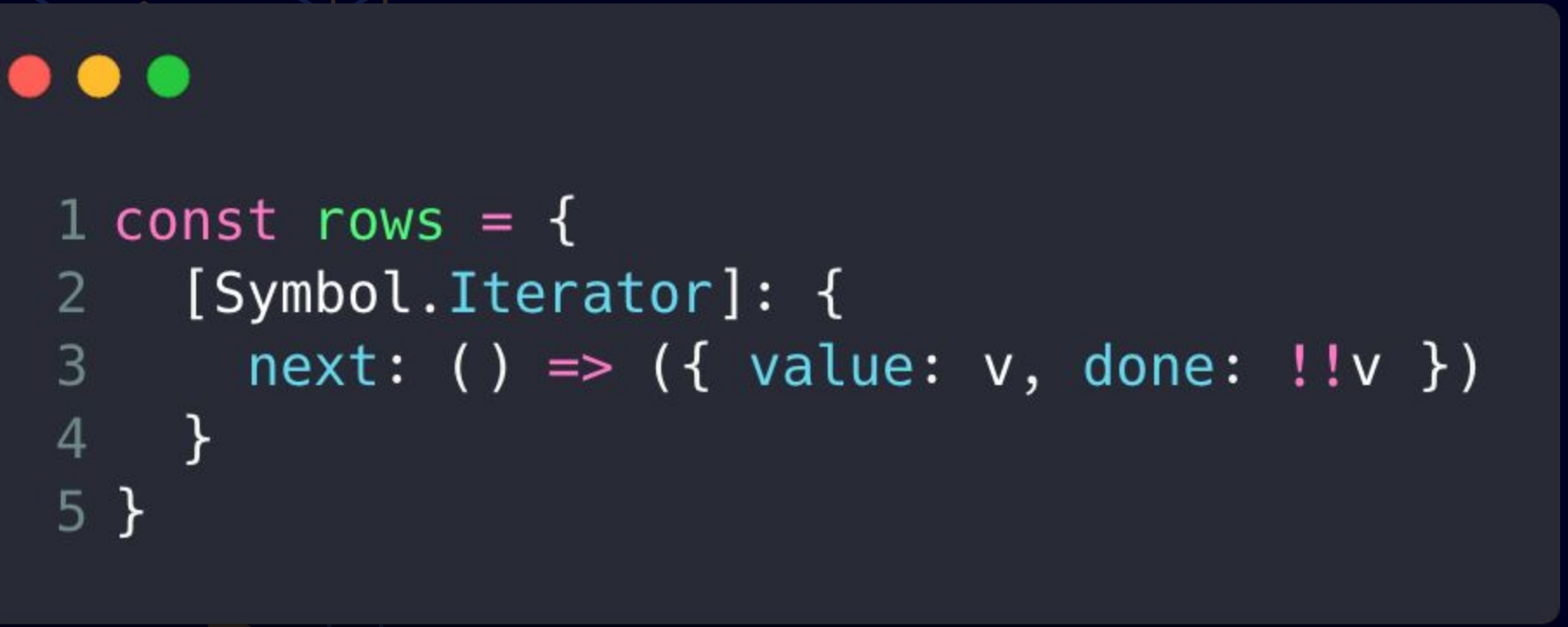
Object.freeze


1 const objeto = { name: 'Lucas' }
2 console.log(objeto) // { name: 'Lucas' }
3
4 objeto.idade = 24
5 // { name: 'Lucas', idade: 24 }


1 const objeto = { name: 'Lucas' }
2 Object.freeze(objeto) // Retorna a instancia do objeto
3
4 objeto.idade = 24 // Erro no strict mode
5 console.log(objeto) // {name: 'Lucas'}

Iteradores Assíncronos_


1 for (let row of rows) {
2 // row
3 }


1 const rows = {
2 [Symbol.Iterator]: {
3 next: () => ({ value: v, done: !!v })
4 }
5 }

```
async function someFunction() {
    const myArray = [1, 2, 3]
    const connection = mysql.createPool({ options })
    let resolvedFinalArray = await Promise.all(myArray.map(async(value) => {
        const result = await asyncFunction(connection, value)
        finalValue.asyncFunctionValue = result.asyncFunctionValue
        return finalValue
    }));
    return functionThatUsesResolvedValues(resolvedFinalArray)
};
```

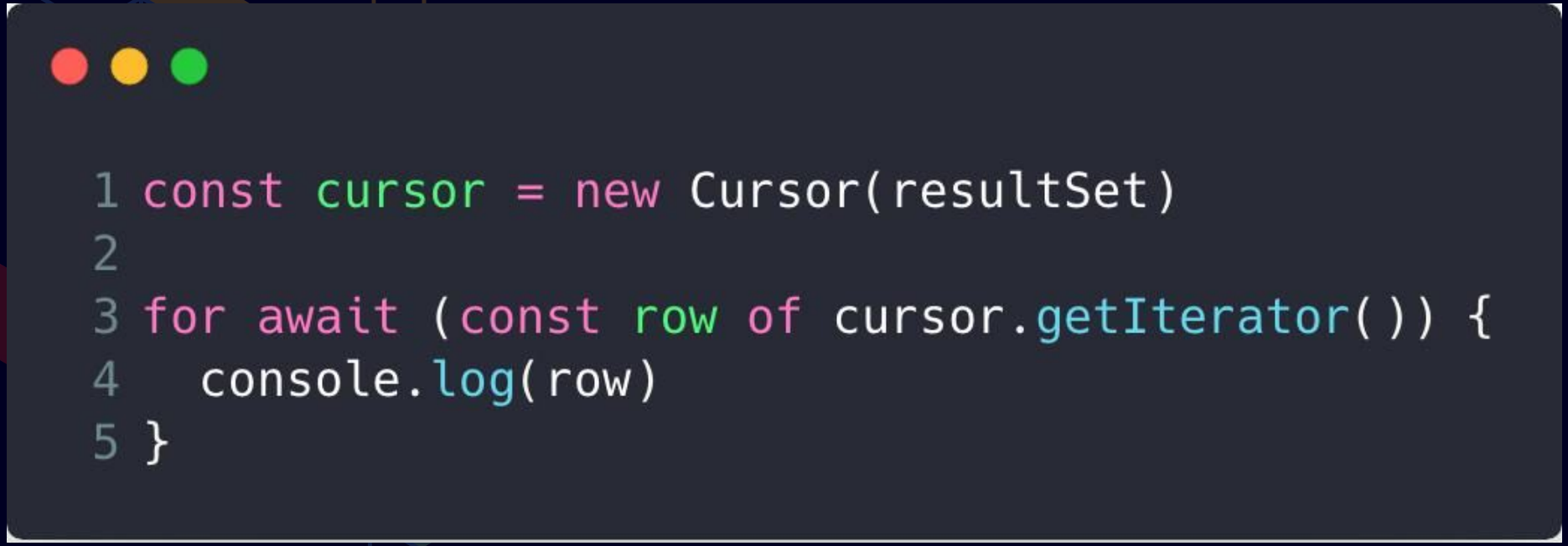

const rows = {
[Symbol.asyncIterator]: {
next: () => this.resultSet.getRow().then(
(row) => ({
value: row, done: row === undefined
})
)
}
}


for await (const row of rows) {
// row
}

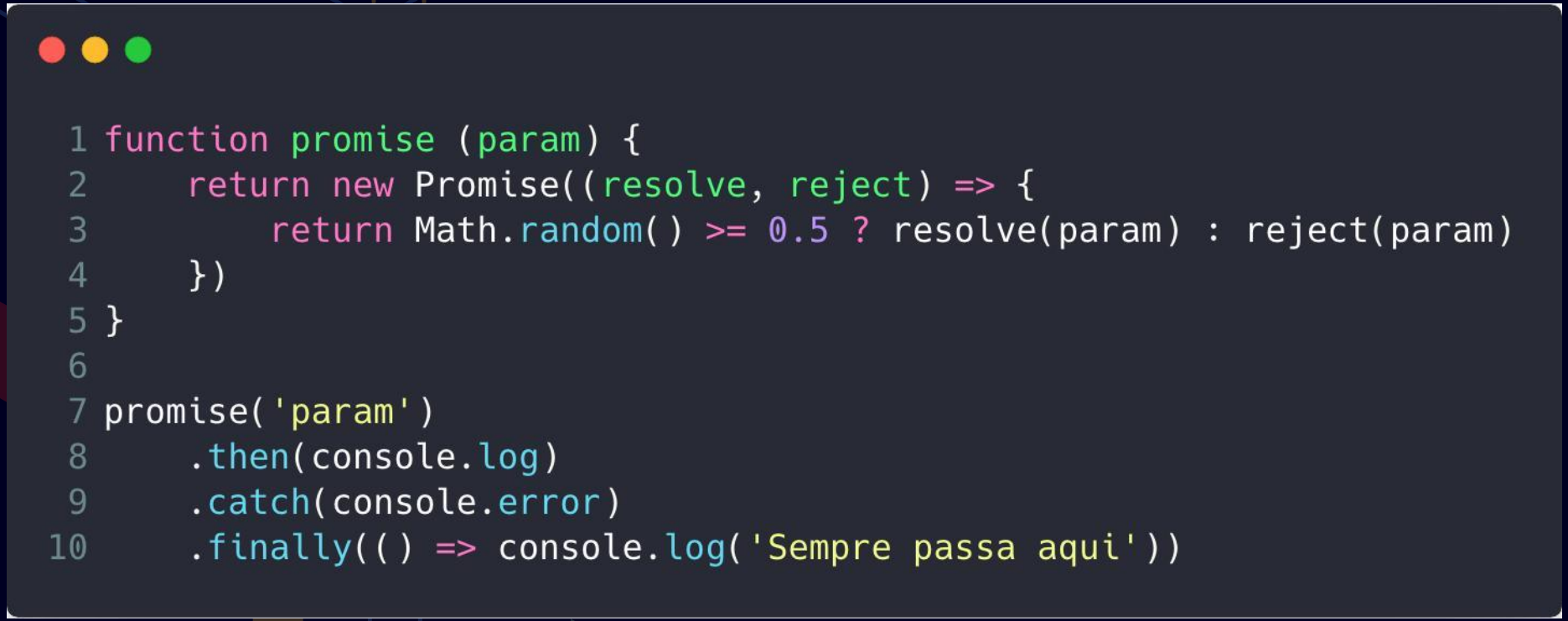
```
class Cursor {
  constructor (resultSet) {
    this.resultSet = resultSet
  }

  getIterator () {
    return {
      [Symbol.asyncIterator]: () => this._buildIterator()
    }
  }

  _buildIterator () {
    return {
      next: () => this.resultSet.getRow().then((row) => ({ value: row, done: row === undefined }))
    }
  }
}
```


const cursor = new Cursor(resultSet)
for await (const row of cursor.getIterator()) {
console.log(row)
}

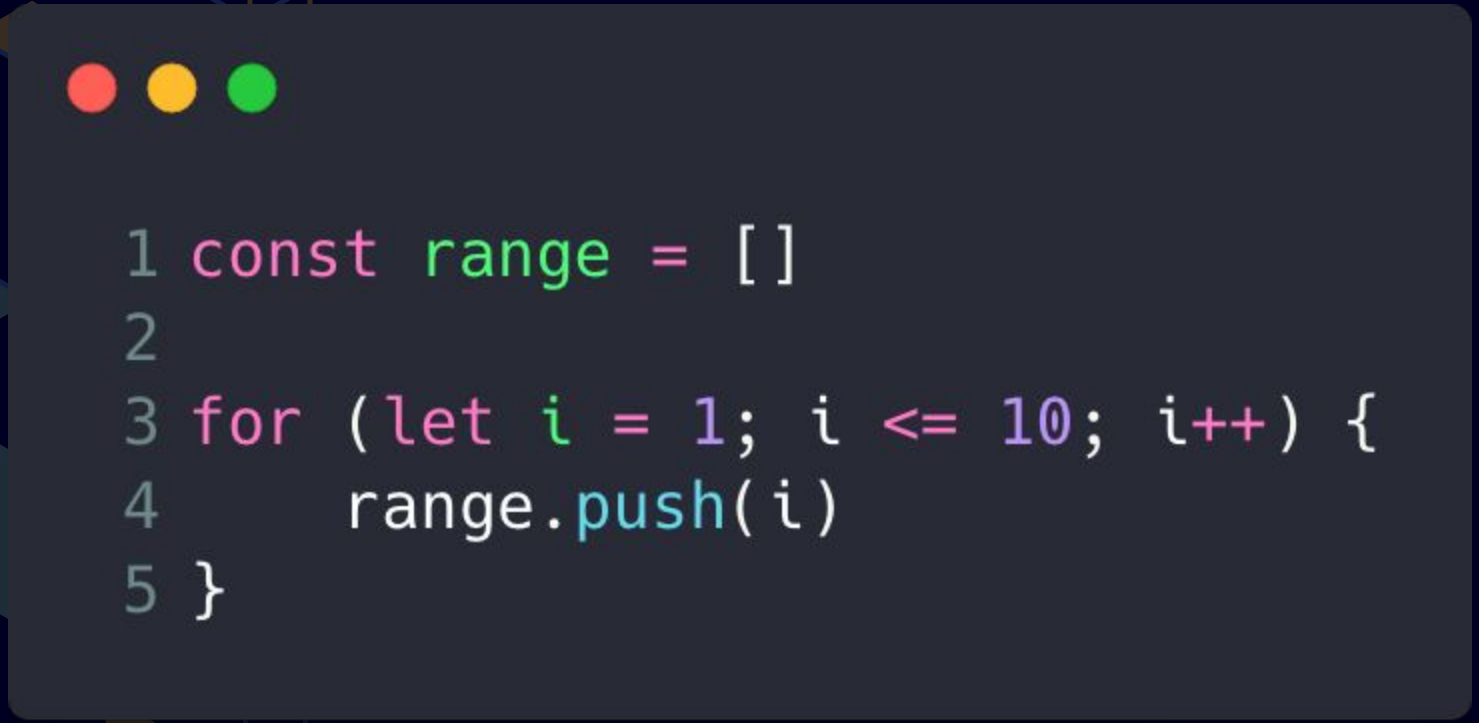
Promise.finally



```
function promise (param) {
    return new Promise((resolve, reject) => {
        return Math.random() >= 0.5 ? resolve(param) : reject(param)
    })
}

promise('param')
    .then(console.log)
    .catch(console.error)
    .finally(() => console.log('Sempre passa aqui'))
```

Ranges


1 const range = []
2
3 for (let i = 1; i <= 10; i++) {
4 range.push(i)
5 }


1 [...Array(tamanho).keys()]


function range(start, size) {
return [...Array(size).keys()].map(i => i + start);
}
range(1, 5) // [1,2,3,4,5]

Propriedades readonly_

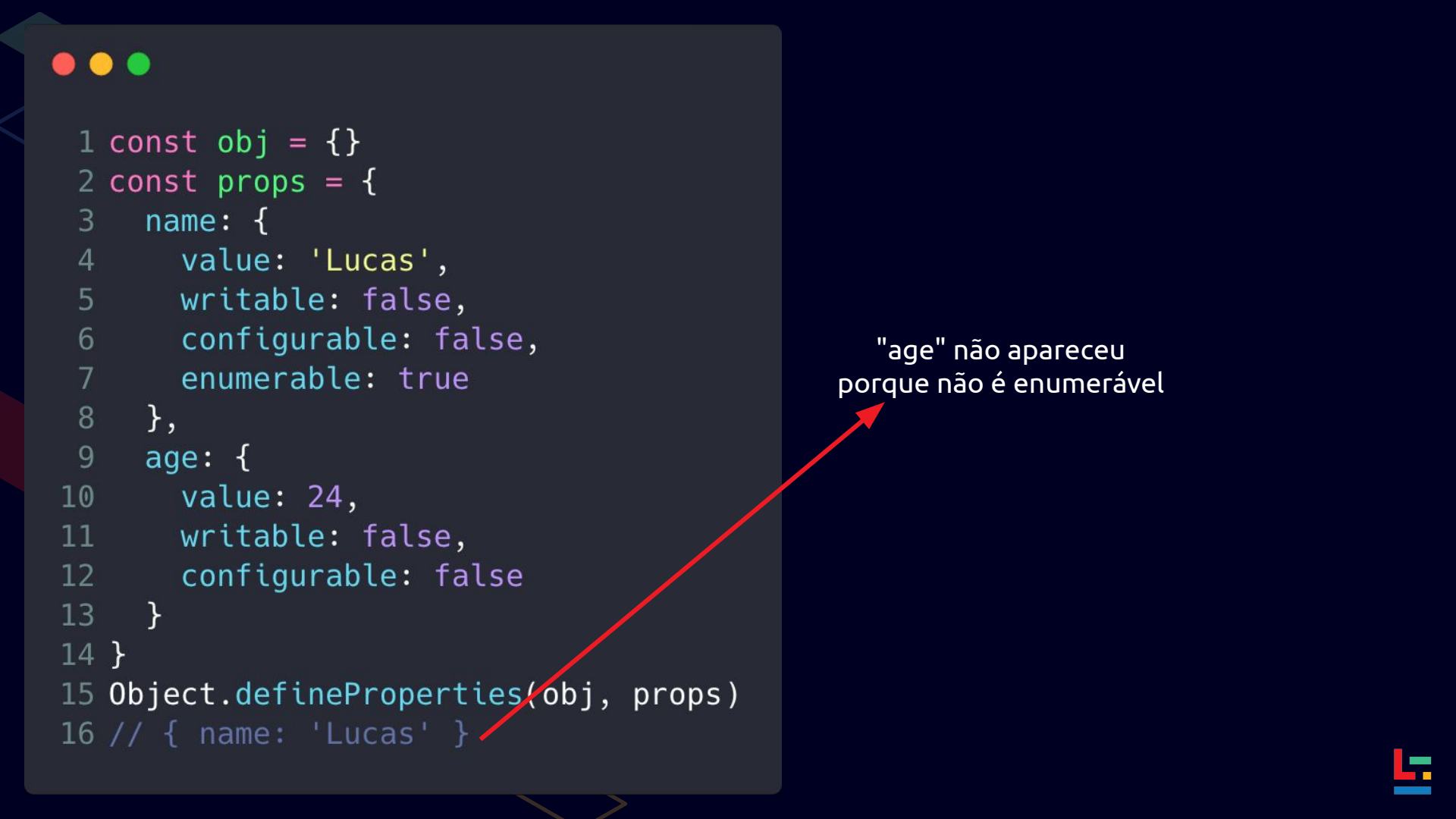
Object.defineProperty


const obj = {}
const props = {
writable: false,
value: 10,
enumerable: false,
configurable: false
}
Object.defineProperty(obj, 'propriedade', props)

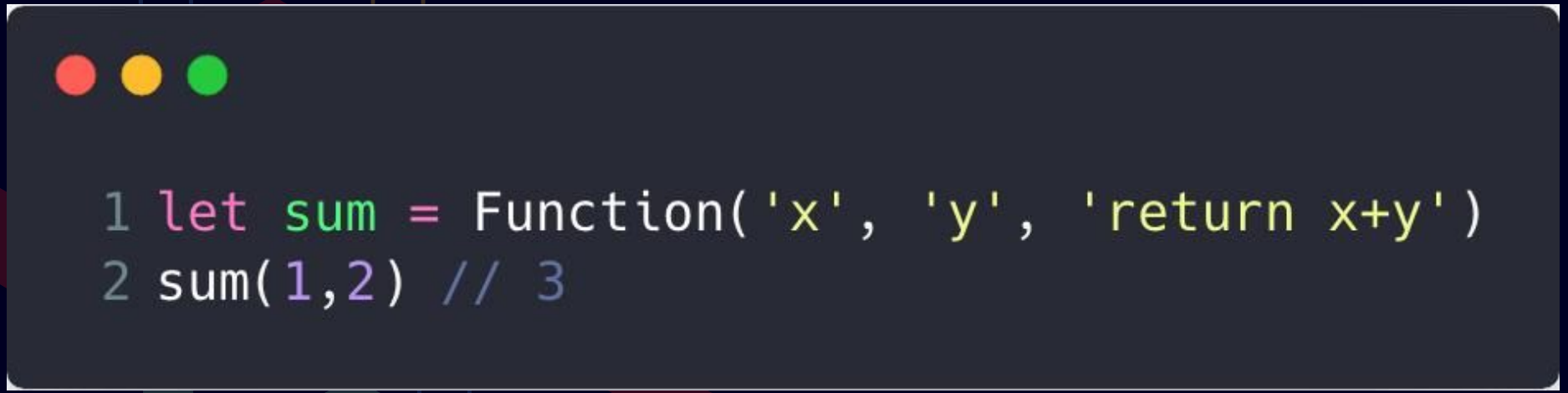
Object.defineProperties

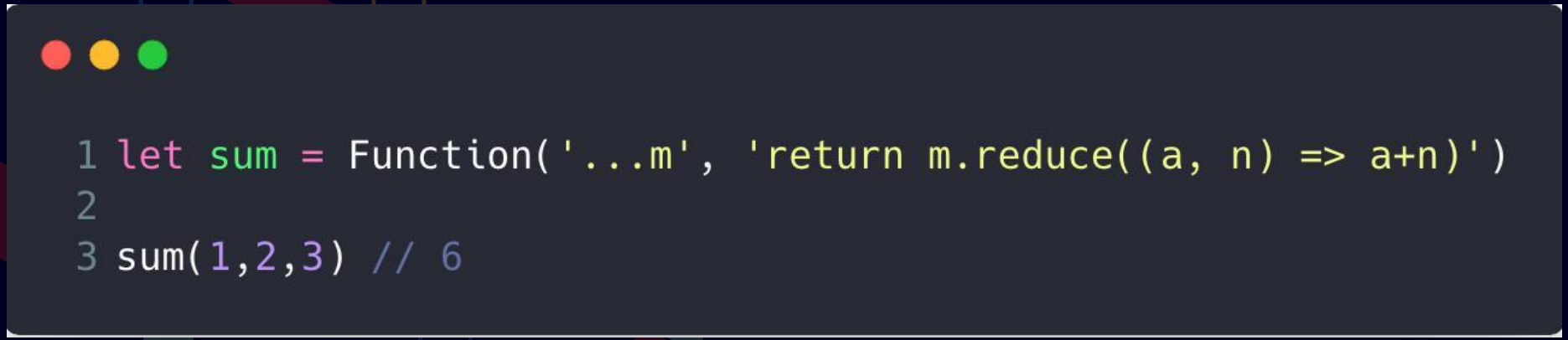
```
1 const obj = {}
2 const props = {
3   name: {
4     value: 'Lucas',
5     writable: false,
6     configurable: false,
7     enumerable: true
8   },
9   age: {
10    value: 24,
11    writable: false,
12    configurable: false
13  }
14 }
15 Object.defineProperties(obj, props)
16 // { name: 'Lucas' }
```

"age" não apareceu
porque não é enumerável



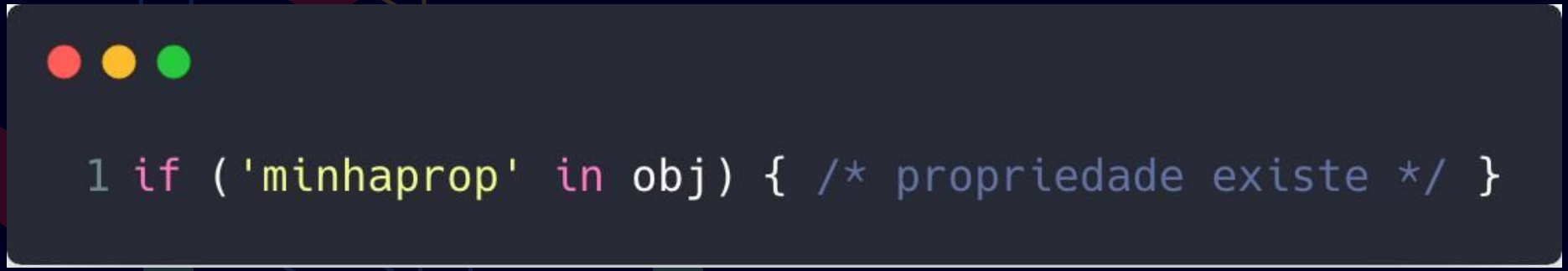
Criação dinâmica de funções_


let sum = Function('x', 'y', 'return x+y')
sum(1,2) // 3


let sum = Function('...m', 'return m.reduce((a, n) => a+n)')
sum(1,2,3) // 6

Operador in


1 if (Object.keys(obj).includes('minhaprop')) { /* propriedade existe */ }


1 if ('minhaprop' in obj) { /* propriedade existe */ }

Construção dinâmica por reflexão_

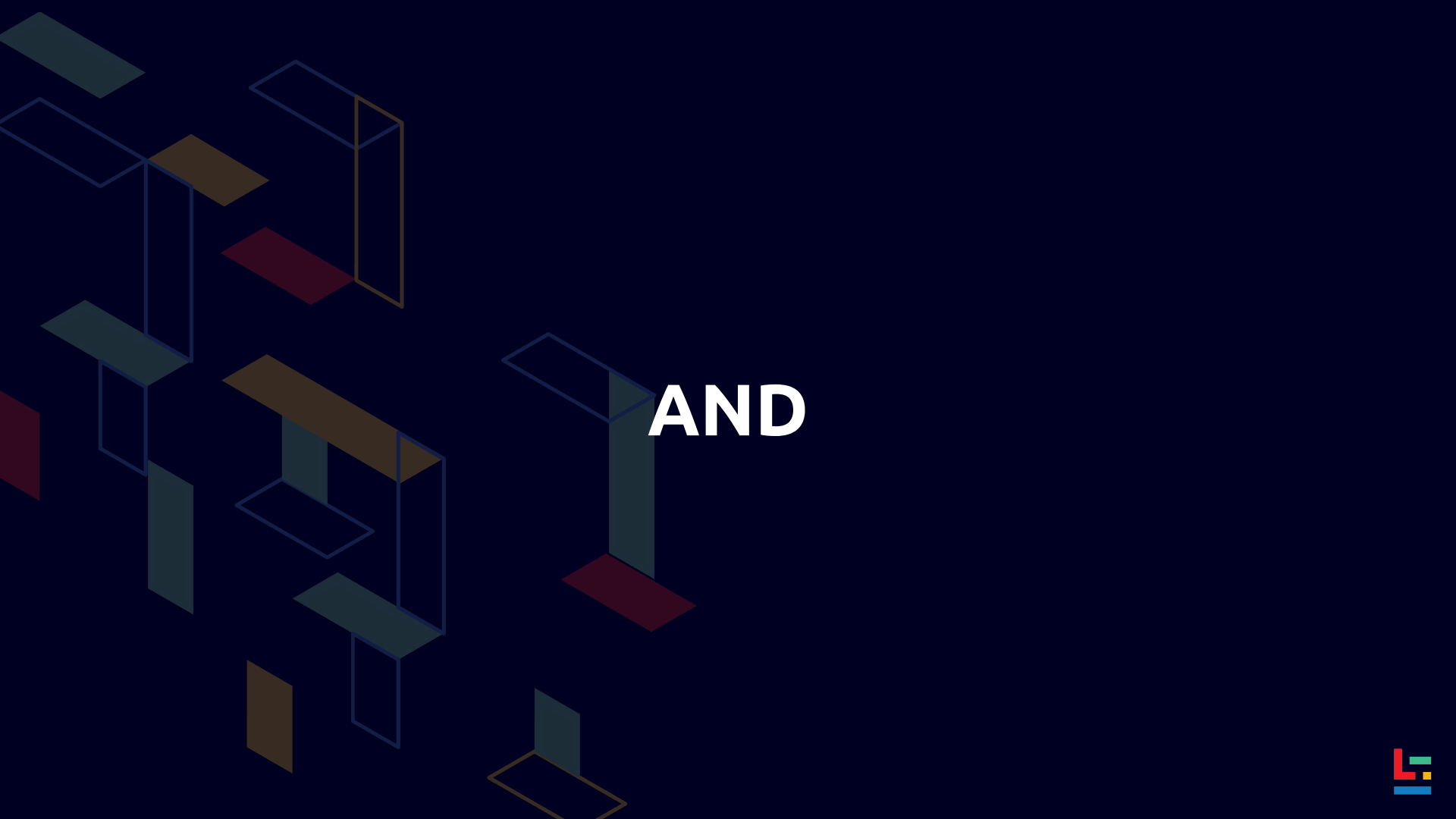
```
1 class Person {
2   constructor (name, age) {
3     this.name = name
4     this.age = age
5   }
6 }
7
8 const Lucas = new Person('Lucas', 24) // Person {name: 'Lucas', age: 24}
```

```
1 class Person {
2   constructor (name, age) {
3     this.name = name
4     this.age = age
5   }
6 }
7
8 const Lucas = Reflect.construct(Person, ['Lucas', 24]) // Person {name: 'Lucas', age: 24}
```


1 function sum (a,b) { return a+b }
2 const c = Reflect.construct(sum, [1,2]) // 3

Bitwise Operators_

AND




1 const valor1 = 5 // 0101 em binário
2 const valor2 = 1 // 0001 em binário
3
4 5 & 1 // 0001

O que?!

	A	B	C	D	Resultado
X	0	1	0	1	5
Y	0	0	0	1	1
Z	0	0	0	1	1

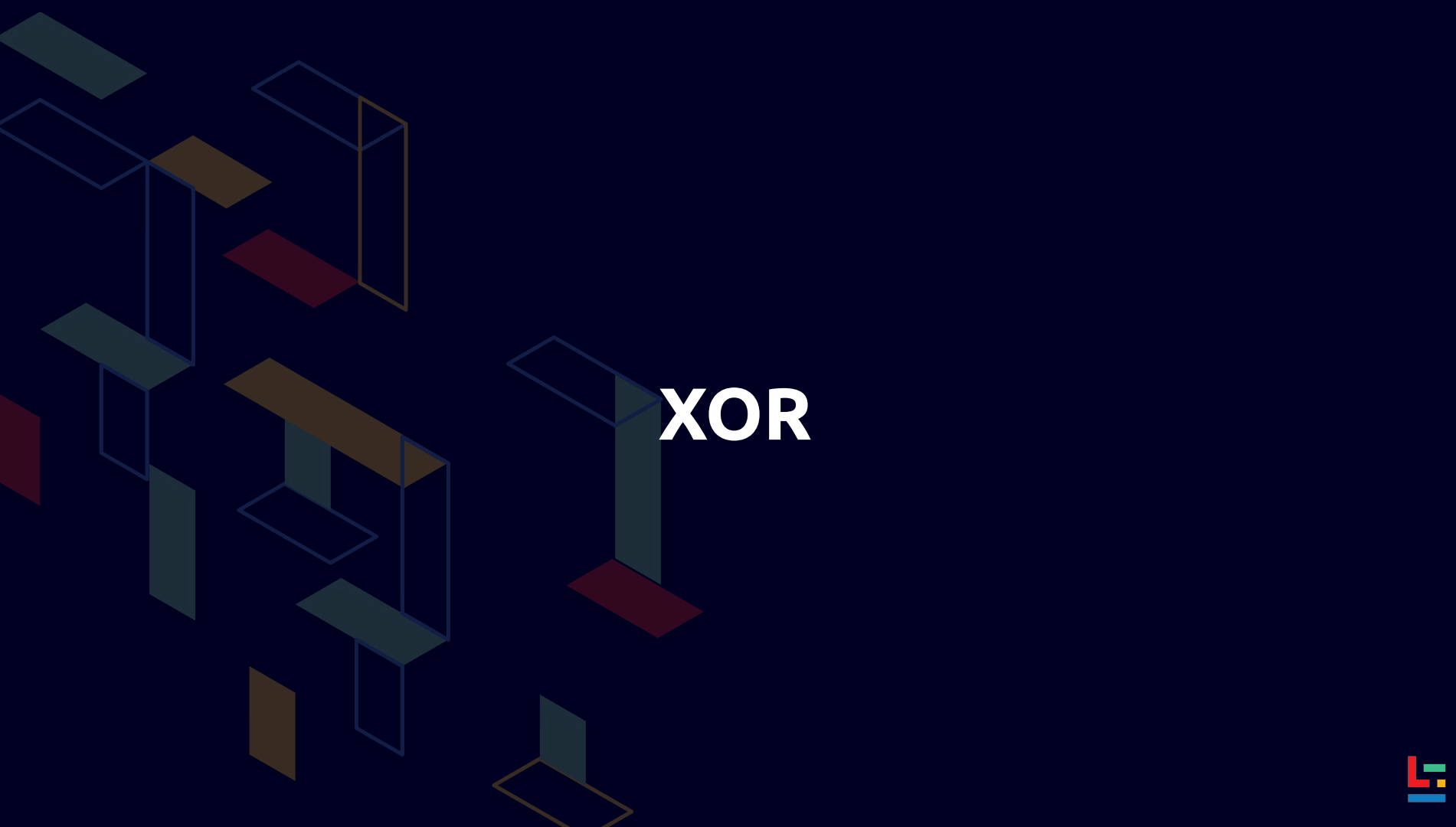
Exibindo dados binários como string

OR


1 const valor1 = 5 // 0101 em binário
2 const valor2 = 1 // 0001 em binário
3
4 5 | 1 // 0101

	A	B	C	D	Resultado
X	0	1	0	1	5
Y	0	0	0	1	1
Z	0	1	0	1	5

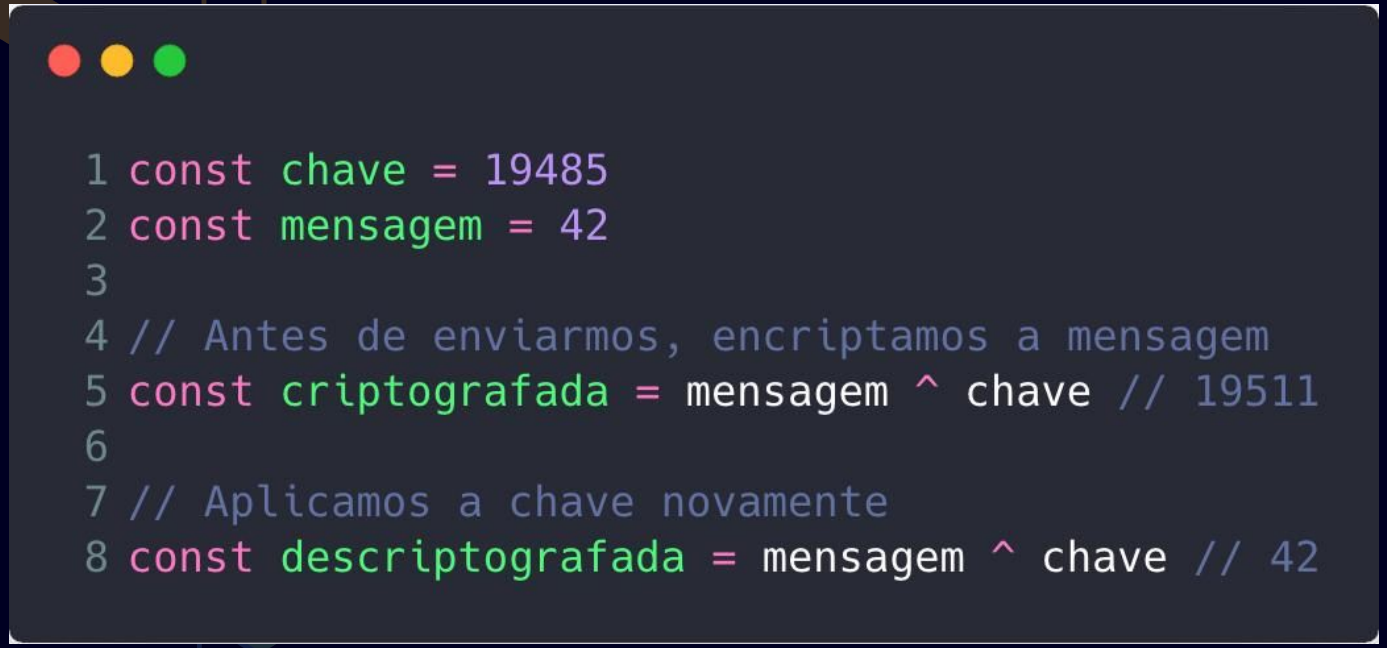
XOR



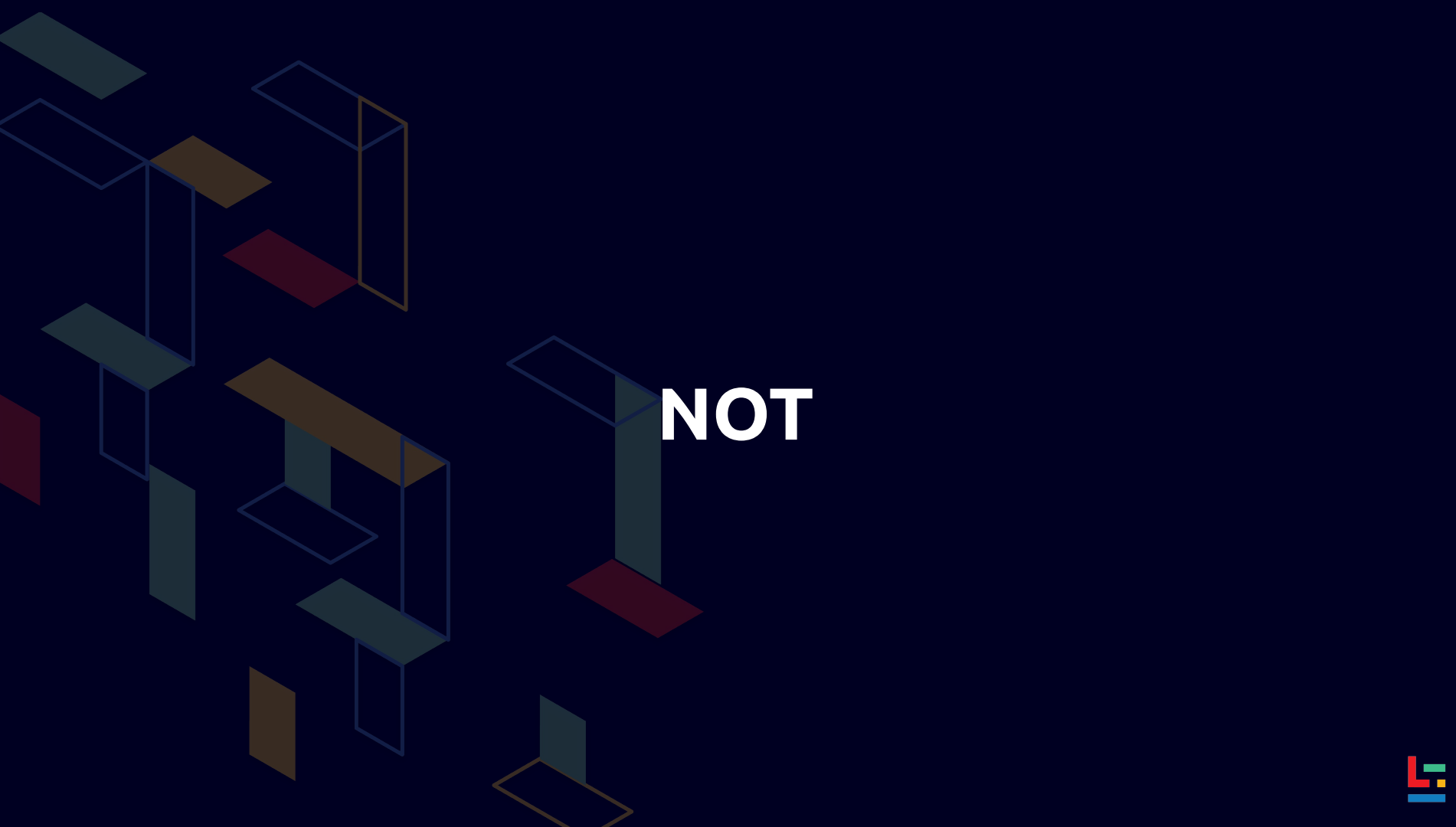

const valor1 = 5 // 0101 em binário
const valor2 = 1 // 0001 em binário
5 ^ 1 // 0100

	A	B	C	D	Resultado
X	0	1	0	1	5
Y	0	0	0	1	1
Z	0	1	0	0	4

Criptografia


1 const chave = 19485
2 const mensagem = 42
3
4 // Antes de enviarmos, encriptamos a mensagem
5 const criptografada = mensagem ^ chave // 19511
6
7 // Aplicamos a chave novamente
8 const descriptografada = mensagem ^ chave // 42

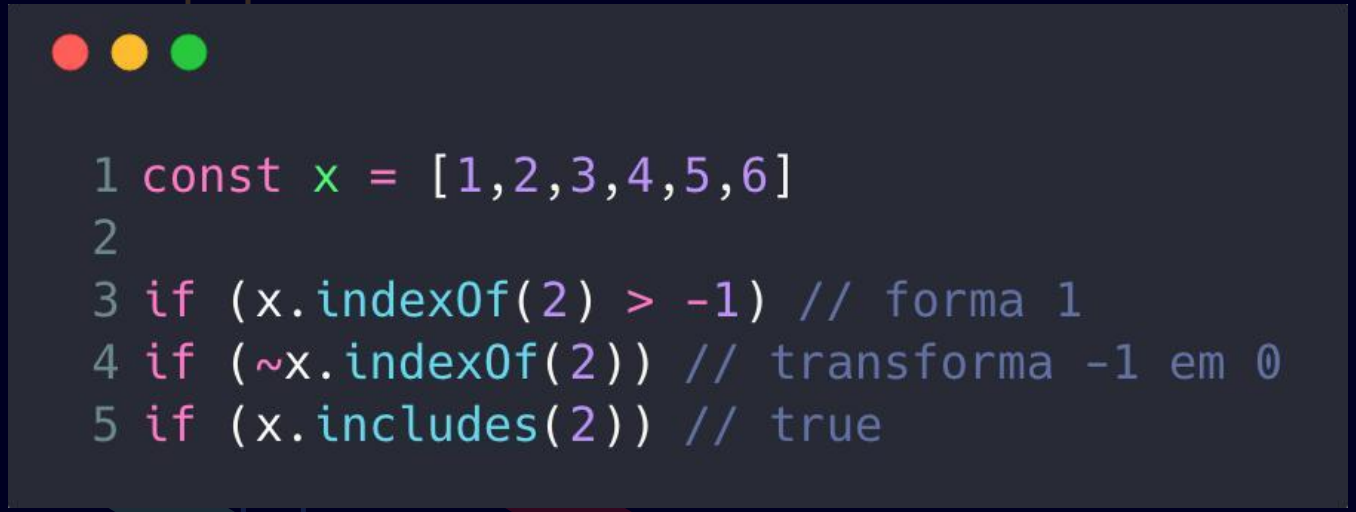
NOT



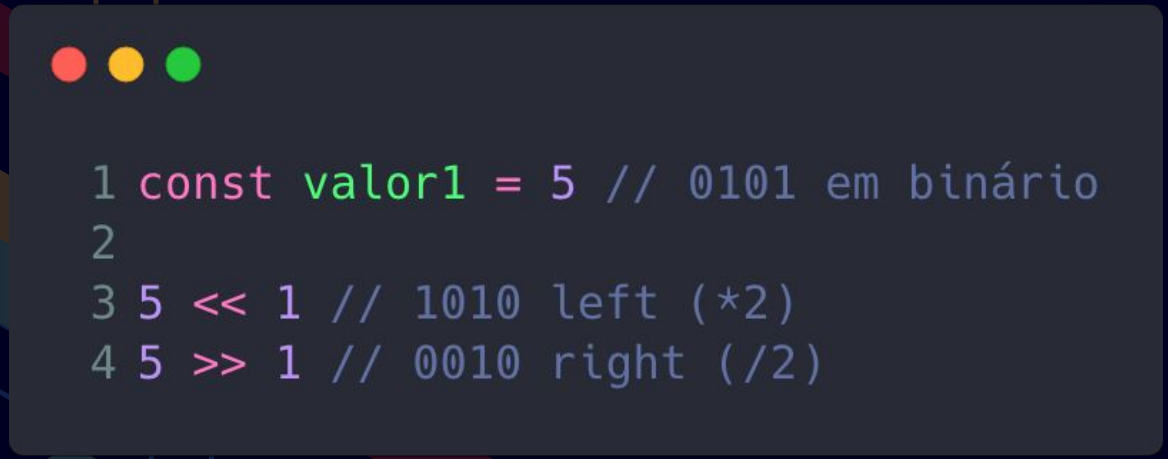

1 const valor1 = 5 // 0101 em binário
2
3 ~5 // 1010

	A	B	C	D	Resultado
X	0	1	0	1	5
Z	1	0	1	0	10

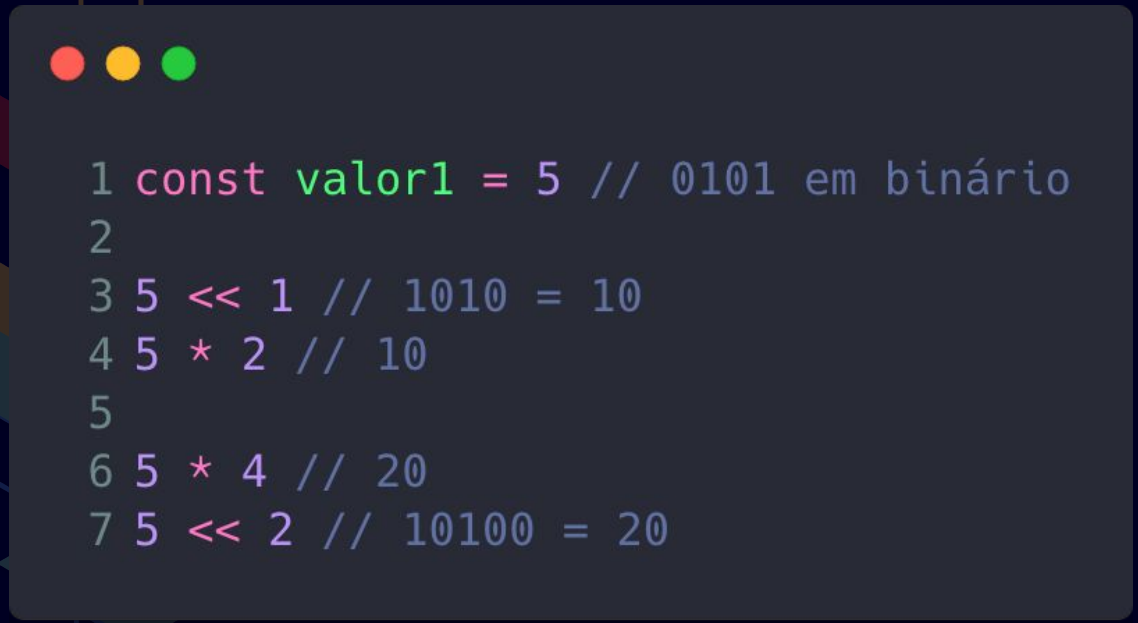
Condição de existência


1 const x = [1,2,3,4,5,6]
2
3 if (x.indexOf(2) > -1) // forma 1
4 if (~x.indexOf(2)) // transforma -1 em 0
5 if (x.includes(2)) // true

SHIFT


const valor1 = 5 // 0101 em binário
5 << 1 // 1010 left (*2)
5 >> 1 // 0010 right (/2)

Multiplicação e divisão


const valor1 = 5 // 0101 em binário
5 << 1 // 1010 = 10
5 * 2 // 10
5 * 4 // 20
5 << 2 // 10100 = 20

referências_

- imasters.com.br/desenvolvimento/o-lado-escuro-javascript
- medium.com/trainingcenter/entendendo-promises-de-uma-vez-por-todas-32442ec725c2
- imasters.com.br/desenvolvimento/campos-publicos-e-privados-em-classes-javascript-o-que-vem-por-ai-no-esnext
- medium.com/trainingcenter/iterators-em-javascript-880adef14495
- medium.com/trainingcenter/reflection-em-javascript-73fc0e702e2
- developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Reflect
- speakerdeck.com/khaosdoctor/desmistificando-heranca-e-prototipos-no-javascript

_Lucas Santos

obrigado_

/khaosdoctor

lsantos.dev

@khaosdoctor

@_staticvoid