CLOSURE

OBJETIVO

Como parametrizar sem comprometer

OBJETIVO

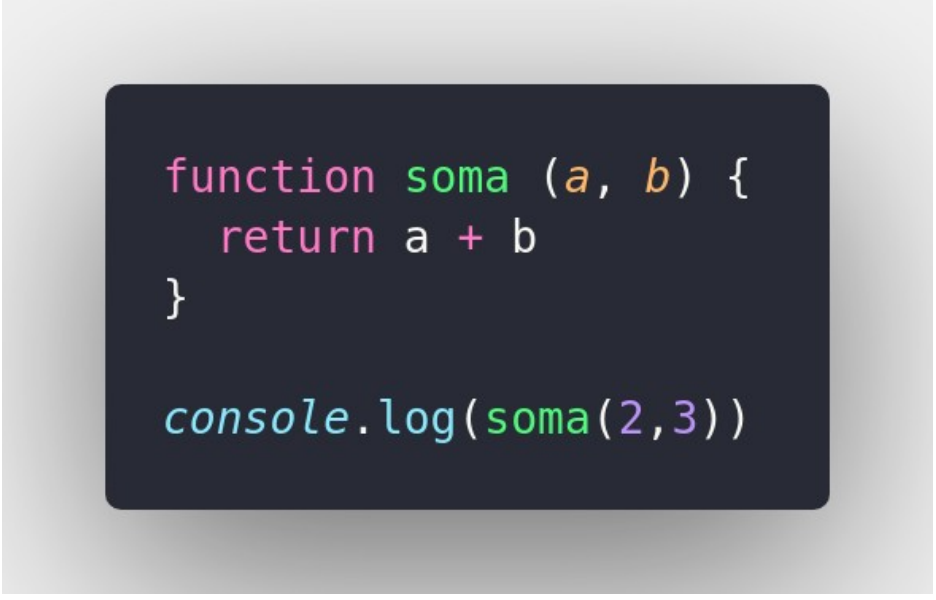
- Repita a si mesmo nos testes
- Resolva casos específicos
- Não configure internamente

Escopos comunicantes

Há 3 caminhos no javascript para passar valores para uma função:

1) Parâmetro
2) Closure
3) This

Parâmetro

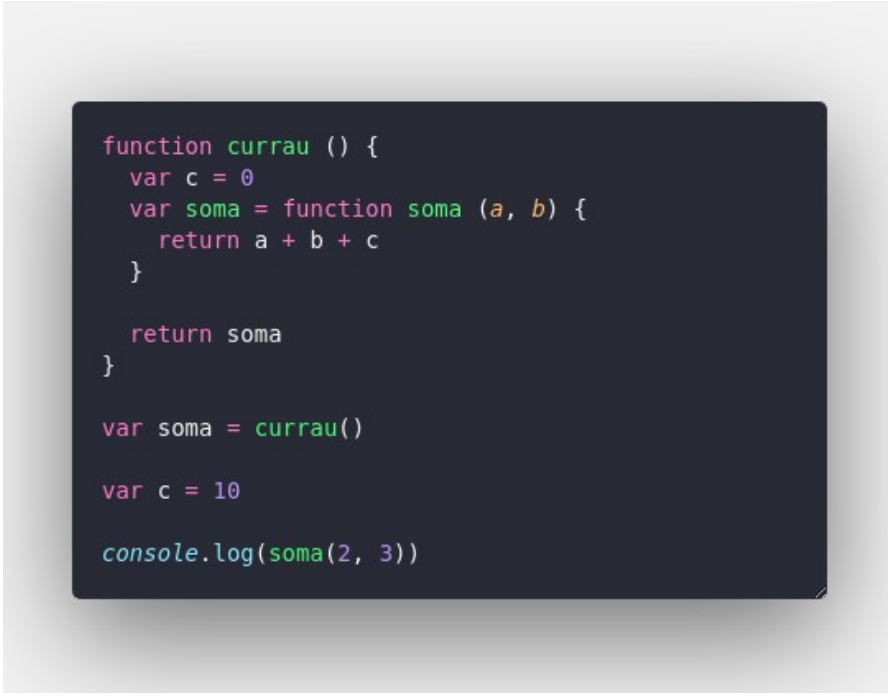


```
function soma (a, b) {
  return a + b
}

console.log(soma(2,3))
```

- Você explicitamente escolhe quais valores vai usar ao executar a função
- ‘a’ passa a ser 2 e ‘b’, 3

CLOSURE



```
function currau () {
  var c = 0
  var soma = function soma (a, b) {
    return a + b + c
  }

  return soma
}

var soma = currau()

var c = 10

console.log(soma(2, 3))
```

THIS

Não use this

THROW

- Throw – arremessar
- try/catch – tentar/pegar

ERRO HANDLING

```
Box(2)
  .then(x => x + 2)
  .then(x => x * 3)
  .unbox()
```

```
Box(2)
  .then(x => {
    console.log(`${x} + 2`)
    return x + 2
  })
  .then(x => {
    console.log(`${x} * 3`)
    return x * 3
  })
  .unbox()
```

ERRO HANDLING

```
Box(2)
  .then(() => {
    throw new Error('ops')
  })
  .then(x => x + 2)
  .then(x => x * 3)
  .unbox()
```

```
Box(2)
  .then(() => {
    throw new Error('ops')
  })
  .then(x => {
    console.log(`${x} + 2`)
    return x + 2
  })
  .then(x => {
    console.log(`${x} * 3`)
    return x * 3
  })
  .unbox()
```

TÚNEL SEM COLATERAL

```
it('## LAZYBOX', () => {
  let x0 = 0
  const b = LazyBox(() => x0 + 2).then(x => x * 3)

  x0 = 2
  const doze = b.unbox()
  assert.equal(doze, 12, 'O valor calculado na caixa deveria ser 12')
})
```