Programação na Internet

Turma i52d

Aula 8
Calling Convention

Function

3 formas de criar funções:

1. function <**name**>([args]) { <body> } – named function
2. function ([args]) { <body> } – anonymous function
3. ([args]) => { <body> } - anonymous function

Chamada a função ===> name()

??? Se a função for anónima como chamamos ???

R: pode ser chamada através de uma referência para essa função.

R: chamando de imediato após a sua declaração!

Function

Propriedades implícitas:

- name – nome da função

Parâmetros implícitos:

- arguments – semelhante a um array com os argumentos da função.
- this – objecto sobre o qual é chamado o método (target)
  - **this != do objecto a que pertence o método.**
  - **this depende da forma como a função é chamada:**
    - **target.name() ====> this é target**
    - **name() ====> this é undefined**

```
function Point(x, y) {
    this.x = x
    this.y = y
    this.print = function () { console.log('Point') }// : ${this.x},
${this.y}`) }
}

const p1 = new Point(5, 7)
p1.print() // p1 => this

const zas = p1.print
zas()      // this ?????
```

- R: p1 ???? Mantém a referencia ???

```
function Point(x, y) {
    this.x = x
    this.y = y
    this.print = function () { console.log(`Point : ${this.x}, ${this.y}`) }
}

const p1 = new Point(5, 7)
p1.print() // p1 => this

const zas = p1.print
zas()      // ....
```

this é o target do método

```
C:\Windows\System32\cmd.exe
D:\ISEL\pg7 pi - 2020-2021 - 1º sem\aulas-52D\aula08-calling-convention>node app01-functions.js
Ola
I am anonynous
Point : 5, 7
D:\ISEL\pg7 pi - 2020-2021 - 1º sem\aulas-52D\aula08-calling-convention\app01-functions.js:18
    this.print = function () { console.log(`Point : ${this.x}, ${this.y}`) }
                                                          ^


TypeError: Cannot read property 'x' of undefined
```

Function

3 formas de criar funções:

1. function <**name**>([args]) { <body> } – named function
2. function ([args]) { <body> } – anonymous function
3. ([args]) => { <body> } - anonymous function

this:

1. function <**name**>([args]) { <body> } – this é o target (e.g. target.name())
2. function ([args]) { <body> } – this é o target
3. ([args]) => { <body> } – this é capturado do **closure**

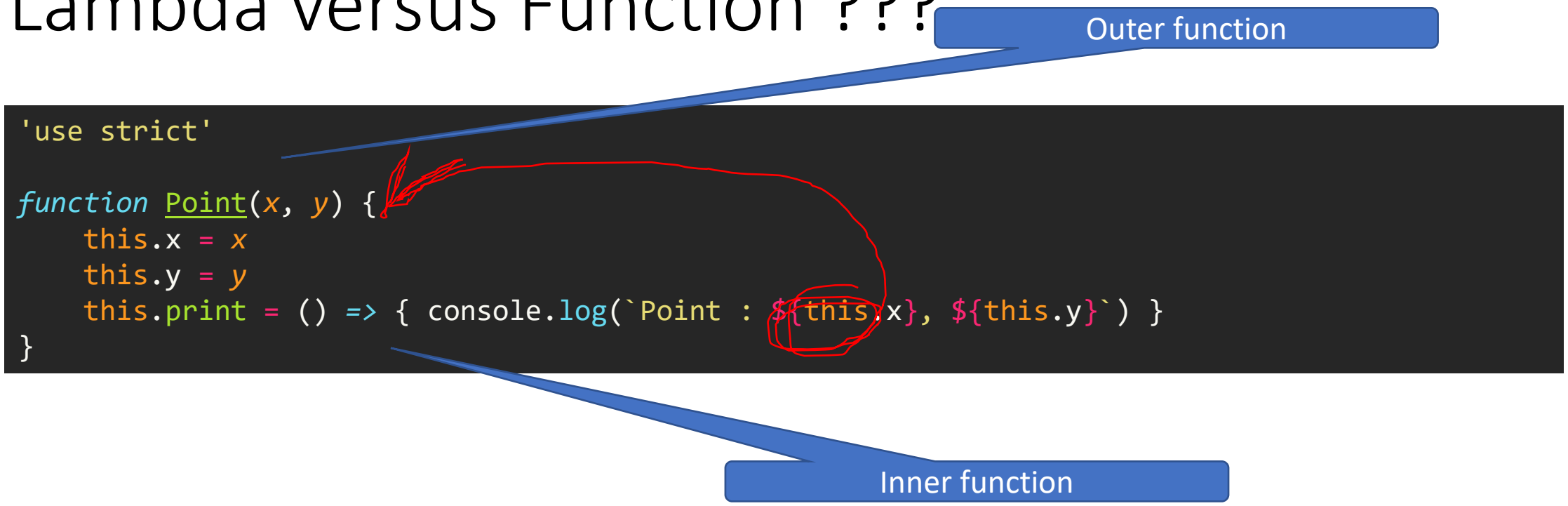
Closure

```
function outer(label) {
    return function inner() {
        console.log(label)
    }
}
```

A variável label usada em inner é capturada do context de execução da função outer.

Variáveis da função externa capturadas pela função interna.

Lambda versus Function ???



this

this:

1. function <**name**>([args]) { <body> } – this é o target (e.g. target.name())
2. function ([args]) { <body> } – this é o target
3. ([args]) => { <body> } – this é capturado do **closure**
4. name.apply(target, array) ou name.call(target, [var args])

Alternativa

```
function Point(x, y) {
    this.print = () => { console.log(`Point : ${x}, ${y}`) }
}

const p1 = new Point(5, 7)
p1.print()
p1.x = 11  //  Adiciona uma nova propriedade x a p1
p1.print()
```

- x e y da outer function (enclosing) são capturados pelo método print
- Os valores de x e y são imutáveis e inacessíveis for a do objecto.

Quiz

Porque é que retorna sempre array vazio?

```
messages.map(item => item.message).filter(msg => msg.lenght < 50)
```

lenght está mal escrito, logo **msg.lenght** é **undefined** o que torna o resultado do predicado sempre **false**.

Quiz

```
function Point(x, y) {
    this.x = x
    this.y = y
    this.print = function () { console.log(`Point : ${x}, ${y}`) }
}

const p1 = new Point(5, 7)
p1.print() // Point: 5, 7
p1.x = 11
p1.print() // Point: 11, 7 ????
```