1. O **this** dentro de uma função está no contexto **global** (ou seja, `this === global`):

```
const xglobal = function () {
    this.x = 11
}
xglobal()
console.log(global.x) // Retorna:  11
```

Assim como:

```
let comparaComThis = function (param) {
    console.log(this === param)
}
comparaComThis(global) // true!!
```

2. O **this** fora de uma função NÃO está no contexto **global** (ou seja, `this !== global`):

```
console.log(this === global) // false
```

3. E por isso, o **this** a seguir:

```
comparaComThis(this)
```

Não é o mesmo **this** abaixo:

```
let comparaComThis = function (param) {
    console.log(this === param)
}
```

4. O **this** fora de uma função está no contexto **module.exports** (ou seja, `this === module.exports`, em se tratando de Node):

```
console.log(this === module.exports) // true

this.y = 3
global.y = 4

console.log(module.exports.y) // Retorna: 3
console.log(y)                // Retorna: 4
```

5. O **this** dentro de uma função sai do contexto **global** quando utilizo **bind** para ligá-lo a um **objeto**. Fazendo com que o **this** dentro da função aponte para o **objeto** em questão:

```
const obj = {}
comparaComThis = comparaComThis.bind(obj)
comparaComThis(global) // false
comparaComThis(obj) // true
```

6. Por fim, o **this** dentro de uma *função arrow* está no contexto **module.exports**, onde ela foi criada (`this === module.exports`).

```
let comparaComThisArrow = param => console.log(this === param)
comparaComThisArrow(global) // false
comparaComThisArrow(module.exports) // true
comparaComThisArrow(this) // true
```

Este **this** é imutável, pois nem o **bind** é capaz de mudar seu contexto:

```
comparaComThisArrow = comparaComThisArrow.bind(obj)
comparaComThisArrow(obj) // false
comparaComThisArrow(this) // true
```