OVIDADES: (leia também comentários deste post)

O que estudar?

Functions:

Arrow Functions

Callbacks

Funções Anônimas

iife

JavaScript Funcional

-----

Objetos (orientação a objetos)

Modularização

RegEx (Regular Expressions)

JSON

AJAX

NodeJS

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

O que é um Object?

OBJECTS são CONTAINERS de PROPRIEDADES e MÉTODOS.

OBJECTS são VARIÁVEIS COMPOSTAS, assim como os ARRAYS.

Array syntax:

let identificador = [valor, valor, valor]

0 1 2

Object syntax:

let identificador = {nome:valor, nome:valor, nome:valor}

Diferenças:

Para declarar um ARRAY usa-se COLCHETES [ ]

Para declarar um OBJECT usa-se CHAVES { }

\_\_\_\_\_\_\_\_

Nos ARRAYS, os ELEMENTOS (conjunção de: posição e valor) são ACESSADOS por seus ÍNDICES (0, 1, 2).

Nos OBJECTS, as PROPRIEDADES (conjunção de: nome e valor) são ACESSADAS por suas KEYS (nome1, nome2, nome3).

Como fazer este ACESSO (rode este código):

const pessoas = {nome1: "rodolfo", nome2: "rogerin"}

const idades = [20, 35]

// Há 2 maneiras de acessar as PROPRIEDADES de um OBJETO.

// 1) syntax: objectName.propertyName

console.log(`O ${pessoas.nome1} tem 20 anos de idade e o ${pessoas.nome2} têm 35.`)

// 2) syntax: objectName["propertyName"]

console.log(`O ${pessoas["nome1"]} tem 20 anos de idade e o ${pessoas["nome2"]} têm 35.`)

// acessando os ELEMENTOS de um ARRAY através de seus ÍNDICES:

console.log(`O rodolfo tem ${idades[0]} anos de idade e o rogerin têm ${idades[1]}.`)

/\* pessoas é um OBJETO;

nome1 e nome2 são as PROPRIEDADES do OBJETO pessoas;

"rodolfo" e "rogerin" são os VALORES das PROPRIEDADES nome1 e nome2 , respectivamente. \*/

\_\_\_\_\_\_\_\_

Os OBJETOS podem também ter MÉTODOS.

Um MÉTODO é uma FUNÇÃO que é ARMAZENADA como PROPRIEDADE de um OBJETO. Ou seja, MÉTODOS são AÇÕES que podem ser EXECUTADAS em OBJETOS.

Exemplo (rode este código):

const person = {

fName: "Elon",

lName: "Musk",

fullName() {

return `${this.fName} ${this.lName}`

}

}

console.log(person.fullName())

/\* this é uma keyword que refere-se a um OBJETO.

Portanto, this.fName refere-se à propriedade fName DESTE (this) OBJETO (person). \*/

\_\_\_\_\_\_\_\_

O que é this ?

this é uma keyword que refere-se a um OBJETO. this NÃO É uma VARIÁVEL. Você NÃO PODE MUDAR o VALOR de this .

Mas QUAL objeto this refere-se? Isto dependerá de COMO esta keyword foi utilizada ou chamada:

Em um MÉTODO, this refere-se ao OBJETO (exemplo anterior).

Sozinha, this refere-se ao OBJETO GLOBAL.

Em uma FUNÇÃO, this refere-se ao OBJETO GLOBAL.

Em uma FUNÇÃO, em modo estrito (\*strict mode), this é undefined.

Em um EVENTO, this refere-se ao ELEMENTO que RECEBEU o EVENTO.

Métodos como call(), apply() e bind() , podem referir this a QUALQUER OBJETO.

\*O que é Strict Mode?

O comando: "use strict" , indica que um código DEVE SER EXECUTADO em Strict Mode, modo estrito. Este pode ser utilizado no INÍCIO do SCRIPT, agindo GLOBALMENTE, ou DENTRO de uma FUNÇÃO ESPECÍFICA, agindo LOCALMENTE.

Este modo serve para REVELAR MÁS PRÁTICAS de ESCRITA, SYNTAX, ao lançar NOVAS MENSAGENS de ERRO no CONSOLE do NAVEGADOR, que antes passariam desapercebidas. Portanto, este modo auxilia na escrita de um código mais "clean", previnindo que você atribua um valor à uma variável antes de declará-la, por exemplo (rode este código):

// SEM UTILIZAR o Strict Mode:

function serHumano() {

return nome = "rodolfo" // atribuindo um valor à uma variável SEM DECLARÁ-LA.

}

console.log(serHumano()) // "rodolfo" foi exibido normalmente sem mensagens de erro.

console.log('-------')

// UTILIZANDO o Strict Mode:

function pessoa() {

"use strict" // caso APAGUE este "use strict", o código NÃO DARÁ ERRO, mesmo que este exista.

return sobrenome = "da Silva" // atribuindo um valor à uma variável SEM DECLARÁ-LA.

}

console.log(pessoa()) // ReferenceError: sobrenome is not defined

// Para saber mais, Google: "JavaScript Use Strict"