Alura - Javascript: desenvolvedor poliglota

Exemplos de modulo:

Conceito de importação de arquivos, ontem podem ser importados e exportardo declarando import ou export

```
Modulo A.js{
import function-1 from 'Modulo B'
import function-5 from 'Mudulo C'
}

Mudulo B.js{
export function-1
export function-2
function-3
}

Mudulo C.js{
function-4
export function-5

import function-2 from 'Modulo B'
}
```

Function( )

Quando chamada, os valores a serem passado deve vem esta na ordem como seguinte exemplo, primeiro peso e depois altura.

```
export function calculaIMC() {
    return peso / (altura * altura)
}

------------

import { calculaIMC } from "./oms.js";
```

```
//peso = 70
//altura = 1,75

const imc = calculaIMC(1.75, 70} )

console.log(imc)//122,7
```

como no exemplo os valores foram invertidos e por isso o IMC teve seu resultado calculado errado

para ter o resultado esperado independente da ordem dos valores é importate definir os parametros/propiedade

```
export function calculaIMC(props) {
    return props.peso / (props.altura * props.altura)
}

------------

import { calculaIMC } from "./oms.js";

//peso = 70
//altura = 1,75

const imc = calculaIMC({altura: 1.75, peso: 70} )

console.log(imc)//22,85
```

exemplo em python

```
def calcula_imc(peso, altura):    return peso / (altura * altura)

imc = calcula_imc(peso=70, altura=1,75)

print imc //22,85
```

Destructuring Assignment

A sintaxe de **atribuição via desestruturação (destructuring assignment)** é uma expressão JavaScript que possibilita extrair dados de arrays ou objetos em variáveis distintas.

Vocês constroe um objeto{ }/arry[ ] com as variaveis que serão usadas e pode desestruturalas e usar as variavies que estão dentro do arry/objeto da forma como quiser

```
export function calculaIMC({peso, altura}) {
    return peso / (altura * altura)
}

-----------

import { calculaIMC } from "./oms.js";

const imc = calculaIMC({altura: 1.75, peso: 70} )

console.log(imc)

const configuration = {
    peso: 68,
    altura: 1.  77
}

const {peso: massa, altura} = configuration

console.log(massa)
console.log(altura)
```

Constructor

comparativos de um contrutor Javascrip com Typescrip

```
export class Conta {

    constructor({titular, banco, agencia, numero}) {
        this.titular = titular
        this.banco = banco
        this.agencia = agencia
        this.numero = numero
    }
}
```

```
export class Conta {

     constructor(public titular:string , public banco:string, public agencia:string, publi
```

```
c numero:string) {

}
```

Obejct.assing

O método `Object.assign()` é usado para copiar os valores de todas as propriedades próprias enumeráveis de um ou mais objetos *de origem* para um objeto *destino* Este método irá retornar o objeto *destino.*

`Object.assign(destino, ...origens)`

```
const target = { a: 1, b: 2 };
const source = { b: 4, c: 5 };
const test = { d: 4, c: 5 };

const returnedTarget = Object.assign(target, source, test);

console.log(returnedTarget);
// expected output: Object { a: 1, b: 4, c: 5 }
```

O object.assing juntas os valores da varivael dentro da primeira declarada (destio, origem) reescrevendo os valores do variavel de destinho.

```
export class Conta {

    constructor({titular, banco, agencia, numero}) {
        Object.assign(this, {titular:titular, banco:banco, agencia:agencia, numero:numero}
)
    }
 ----------

import { calculaIMC } from "./oms.js";
import { Conta } from "./model/conta.js";

const imc = calculaIMC({altura: 1.75, peso: 70} )

console.log(imc)

const configuration = {
    peso: 68,
    altura: 1.77
}
```

```
const {peso: massa, altura} = configuration

console.log(massa)
console.log(altura)

const conta = new Conta({
    titular: 'Joana',
    banco: 'Rico',
    agencia: '123',
    numero: '3210'
})

console.log(conta)

const object1 = {nome: 'Joana'}
const object2 = {peso: 56}

Object.assign(object1, object2)
console.log(object1)
```