STEP Javascript - Homework 27
Arrow functions e High Order Functions - Continuação

No código que segue ao final do doc (fornecido por um de vocês como solução do homework anterior), prosseguimos revendo e desenvolvendo alguns exemplos-teste de como definir e usar arrow functions. O código amplia também o uso de high order functions, além de outras funções além da função map().

Lembrando que as high order functions são funções de objeto e/ou listas (a função map() é uma função built-in de arrays) que geralmente recebem, como argumento de entrada, outra function - geralmente uma arrow funtcion - (e opcionalmente podem returnar uma outra function, o que costumamos chamar de *programação funcional*, mas que só veremos em cursos mais avançados).

Exercício: Baseando-se no código apresentado, crie arrow functions que atuarão sobre a coleção de dados que segue:

```
var notas = [
   {nome: 'João',        { pgto1: 100.08,        pgto2: 20.00,   pagto3: 5.00}, sexo: 'M' },
   {nome: 'Sara',        { pgto1: 10.0,          pgto2: 10.30,   pagto3: 1.00}, sexo: 'F' },
   {nome: ‘Francisco',   { pgto1: 12.00,         pgto2: 30.33,   pagto3: 2.00}, sexo: 'M' },
   {nome: 'Paulo',       { pgto1: 1.02,          pgto2: 2.50,    pagto3: 3.00}, sexo: 'M' },
   {nome: 'Everton',     { pgto1: 15.20,         pgto2: 2.80,    pagto3: 5.00}, sexo: 'M' },
   {nome: 'Paula',       { pgto1: 26.00,         pgto2: 28.00,   pagto3: 1.00}, sexo: 'F' },
   {nome: 'Demétrio',    { pgto1: 5.00,          pgto2: 57.10,   pagto3: 5.00}, sexo: 'M' },
   {nome: 'Márcio',      { pgto1: 8.00,          pgto2: 3.10,    pagto3: 2.00}, sexo: 'F' },
   {nome: 'Carlos',      { pgto1: 5.00,          pgto2: 12.00,   pagto3: 8.00}, sexo: 'F' },
   {nome: 'Janaína',     { pgto1: 10.01,         pgto2: 10.45,   pagto3: 9.00}, sexo: 'F' }
]
```

As funções serão:

O total de pagamentos de cada pessoa
O total de pagamentos de sexo masculino
O total de pagamentos de sexo feminino
O maior pagamento de cada pessoa do sexo masculino
O maior pagamento de cada pessoa do sexo feminino
O maior pagamento entre todas as pessoas de sexo masculino
O maior pagamento entre todas as pessoas de sexo feminino
O maior de todos os pagamentos

(obviamente, você também deve escrever código que usa essas funções, mostrando os resultados na página, classicamente, os “getElementById”. Nas funcionalidades que se

referem a “cada pessoa” : total de pagamentos, maior pagamento, etc - deve ser exibido também o nome da pessoa)

DICA: repetindo esta dica: além do código apresentado ao final do doc, dê uma olhada no seguinte exemplo que seria aplicado a esta coleção de alunos:

let isMasculino = estudante => estudante.sexo === ‘M’
let getAlunosMasculino = notas => (notas.filter(isMasculino))
// IMPORTANTE: esta notação, não vista no código na sala, é a condensação completa
// de uma arroy function. A função filter() é uma high order function (com uma
// funcionalidade parecida com a da função map()) que usa a arrow function
// isMasculino() como argumento de entrada. Além disso, a função na segunda linha
// ela está inteiramente envolvida entre parentesis. Desenvolveremos esta forma na próxima aula

==========
Entregue A PASTA (diretório) com os arquivos .html, .js e outros se houverem. Não entregue .txt, .docx, etc. Faça um zip da pasta e upload deste zip
==========

=== Codigo sobre Arrow Funcions e High Order Functions ===

hm26.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <script defer src="codigo26.js"></script>
    <title>HM 26</title>
</head>
<body>
    <div id="notas"></div> <br> <br>
    <div id="avaliacoes"></div>
</body>
</html>
```

codigo26.js

```
// coleção de dados
```

```
var notas = [
    { nome: 'João', nota: 8, sexo: 'M' },
    { nome: 'Sara', nota: 12, sexo: 'F' },
    { nome: 'Francisco', nota: 16, sexo: 'M' },
    { nome: 'Paulo', nota: 2, sexo: 'M' },
    { nome: 'Everton', nota: 4, sexo: 'M' },
    { nome: 'Paula', nota: 18, sexo: 'F' },
    { nome: 'Demétrio', nota: 5, sexo: 'M' },
    { nome: 'Márcio', nota: 13, sexo: 'F' },
    { nome: 'Carlos', nota: 15, sexo: 'F' },
    { nome: 'Janaína', nota: 9, sexo: 'F' }
];

// variáveis auxiliares globaix
var soma = 0;
var me = 0;
var maior;

// arrow functions "utilitárias"
// const isSexo = (param) => { return param.sexo === sexo };
const getNota = (turma) => { return turma["nota"] }
const somar = (nt) => { soma += nt }
const acharMaior = (nota) => {
    if (nota > maior) {
        maior = nota
    }
}

// outra função auxiliar
function stringSexo(sexo) {
    if (sexo === "M") {
        return "Homens"
    } else if (sexo === "F") {
        return "mulheres"
    } else {
        return "inválido"
    }
```

```
}

// média que é reutilizada outras funções
function media(listaNt, quem) {
    var lista = listaNt.map(getNota);
    soma = 0;
    lista.forEach(somar)
    me = soma / lista.length
    document.getElementById("avaliacoes").innerHTML += ("Média " + quem + ": " +
me) + "<br>"
}

// função autoexecutável que faz a extração das notas da coleção inteira
(function mediaTurma() {
    media(notas, "turma");
})();

function mediaSexo(sexo) {
    const isSexo = (param) => { return param.sexo === sexo };
    listaPorSexo = notas.filter(isSexo)
    media(listaPorSexo, stringSexo(sexo))
}

function maiorNota(lista, quem) {
    var listaMaior = lista.map(getNota);
    maior = listaMaior[0]
    listaMaior.forEach(acharMaior)
    document.getElementById("avaliacoes").innerHTML += ("maior nota" + quem + " :
" + maior) + "<br>"
};

function maiorPorSexo(sexo) {
    const isSexo = (param) => { return param.sexo === sexo };
    listaPorSexo = notas.filter(isSexo)
    maiorNota(listaPorSexo, stringSexo(sexo));
}
```

```
const exibeNotas = (param) => {
    document.getElementById("notas").innerHTML += param["nome"] + ": "
    document.getElementById("notas").innerHTML += param["nota"] + "<br>"
}
notas.forEach(exibeNotas)

// o "main code"
mediaSexo("M")
mediaSexo("F")
maiorPorSexo("M")
maiorPorSexo("F")
```