**Coleções de dados com Arrays**

A nossa aplicação possui uma lista de pacientes, e agora já estamos calculando o IMC de um paciente!

Imagine então que eu quero ter a altura de todos os meus pacientes na minha aplicação, se eu tivesse 3 pacientes eu teria três variáveis:

var altura1 = 1.57;

var altura2 = 2.00;

var altura3 = 1.64;

Note que, para cada paciente, teríamos uma nova variável. Porém, nós gostaríamos de organizar melhor esse código e não precisar criar uma variável para cada altura.

Para isso, poderíamos criar uma coleção de alturas, também chamado de Array. Em Javascript, podemos criar umArray simplesmente criando uma variável e atribuindo a ela o valor []:

var alturas = [];

Deste modo, estamos criando um Array vazio, caso queiramos adicionar um elemento ao array basta colocá-lo dentro dos colchetes:

var alturas = [1.57, 2.00, 1.64];

Mas como vamos acessar essas alturas agora que elas estão todas em uma mesma variável? Para acessarmos valores dentro de um Array, temos que dizer, entre colchetes, qual a *posição* do Array que queremos acessar. Note que o primeiro elemento, está na posição **0**:

var alturas = [1.57, 2.00, 1.64];

var primeiraAltura = alturas[0];

console.log(primeiraAltura); // 1.57

Então, caso queiramos pegar o *segundo* elemento do Array, utilizaríamos a posição(ou índice) **1**, se quisermos o terceiro, utilizaríamos **2** e assim por diante:

var alturas = [1.57, 2.00, 1.64];

console.log(alturas[0]); // 1.57

console.log(alturas[1]); // 2.00

console.log(alturas[2]); // 1.64

Legal, agora organizamos melhor as alturas dos nossos pacientes, mas seria melhor se utilizássemos o que aprendemos sobre objetos, para colocarmos pacientes inteiros dentro de nosso Array, e não só alturas.

Para isso, criaremos dois pacientes, como já fazíamos anteriormente, e criaremos um Array chamadopacientes contendo estes objetos.

var peso1 = document.getElementById("peso-1").textContent; // pega o peso do paciente

var altura1 = document.getElementById("altura-1").textContent; // pega a altura do paciente

var nome1 = document.getElementById("nome-1").textContent; // pega nome do paciente

var leonardo = {

peso : peso1,

altura : altura1,

nome : nome1

}

var peso2 = document.getElementById("peso-2").textContent; // pega o peso do paciente

var altura2 = document.getElementById("altura-2").textContent; // pega a altura do paciente

var nome2 = document.getElementById("nome-2").textContent; // pega nome do paciente

var paulo = {

peso : peso2,

altura : altura2,

nome : nome2

}

var pacientes = [leonardo, paulo];

Para acessar o leonardo, basta acessar a posição **0** do Array, mas note que agora ele irá retornar um objeto:

//monta pacientes

var pacientes = [leonardo, paulo];

console.log(pacientes[0]) // imprime {altura : (altura do seu primeiro paciente), peso : (peso do primeiro seu paciente), nome: (nome do primeiro seu paciente)}

Para acessarmos a altura desse paciente, basta utilizarmos a *dot notation*, como vimos anteriormente:

//monta pacientes

var pacientes = [leonardo, paulo];

var primeiroPaciente = pacientes[0];

console.log(primeiroPaciente.altura); // imprime altura do primeiro paciente

//pacientes[0].altura; tería o mesmo resutado.

Agora nosso código já está bem mais organizado!

**Percorrendo arrays utilizando os comandos while e for**

Agora nós temos uma coleção de pacientes, mas e se quisermos calcular o IMC de cada paciente? Precisaríamos calcular individualmente para cada item de nosso Array, copiando o código que calcula o IMC para cada um deles:

//monta pacientes

var pacientes = [leonardo, paulo];

var primeiroPaciente = pacientes[0];

if(primeiroPaciente.altura != 0){

var imcDoPrimeiroPaciente = primeiroPaciente.peso / (primeiroPaciente.altura \* primeiroPaciente.altura);

console.log(imcDoPrimeiroPaciente) // 22.31...

}else{

console("Não posso executar uma divisão por 0!");

}

var segundoPaciente = pacientes[1];

if(segundoPaciente.altura != 0){

var imcDosegundoPaciente = segundoPaciente.peso / (segundoPaciente.altura \* segundoPaciente.altura);

console.log(imcDosegundoPaciente) // 25

}else{

console("Não posso executar uma divisão por 0!");

}

Com dois pacientes ainda não é tão ruim, mas imagina se tivéssemos 100 deles? Teríamos de copiar esse código 100 vezes, um para cada paciente!

O que queremos fazer é executar um mesmo código para cada uma das posições de nosso Array, seria legal se pudéssemos fazer isso de forma automática, de algum modo, com que um bloco de código seja executado várias vezes. Ou seja, um Loop! Algo como o seguinte:

enquanto(naoEhOUltimoElementoDoArray){

if(pacienteAtual.altura != 0){

var imcDoPacienteAtual = pacienteAtual.peso / (pacienteAtual.altura \* pacienteAtual.altura);

console.log(imcDoPacienteAtual)

}else{

console("Não posso executar uma divisão por 0!");

}

}

Para fazermos nosso código ser executado para cada item do Array, utilizaremos o comando while. Para utilizá-lo, precisamos dar uma condição que será usada para, enquanto for verdadeira, executar o bloco de código. A condição que queremos é: **se a posição atual não é a última do array**, continua executando o loop. Para verificarmos se a posição atual é a última ou não, precisamos descobrir o tamanho do array, o que pode ser feito acessando o atributo length do array:

//monta pacientes

var pacientes = [leonardo, paulo];

console.log(pacientes.length); // imprime 2

Lembre-se de que as posições do Array começam de **0**, portanto, a última posição é o **tamanho do array menos um (pacientes.length - 1)**. Sendo assim, a condição que precisamos é posicaoAtual <= pacientes.length - 1, e é essa condição que usaremos em nosso comando while:

var posicaoAtual = 0;

while(posicaoAtual <= pacientes.length - 1){

var pacienteAtual = pacientes[posicaoAtual];

if(pacienteAtual.altura != 0){

var imcDoPacienteAtual = pacienteAtual.peso / (pacienteAtual.altura \* pacienteAtual.altura);

console.log(imcDoPacienteAtual);

}else{

console("Não posso executar uma divisão por 0!");

}

}

Mas note que, deste modo, a posicaoAtual sempre valerá 0 e, portanto, a condição nunca será falsa! Entramos em um loop infinito! Para evitar esse problema, devemos somar 1 à posicaoAtual a cada loop:

var posicaoAtual = 0;

while(posicaoAtual <= pacientes.length - 1){

//resto do código

posicaoAtual = posicaoAtual + 1;

}

Incrementar 1 a uma variável é algo bem comum no Javascript, por isso já existe um **atalho**, que faz a mesma coisa. Podemos substituir a linha que soma 1 por posicaoAtual++, que terá o mesmo resultado!

O resultado será:

//monta pacientes

var pacientes = [leonardo, paulo];

var posicaoAtual = 0;

while(posicaoAtual <= pacientes.length - 1){

var pacienteAtual = pacientes[posicaoAtual];

if(pacienteAtual.altura != 0){

var imcDoPacienteAtual = pacienteAtual.peso / (pacienteAtual.altura \* pacienteAtual.altura);

console.log(imcDoPacienteAtual);

}else{

console("Não posso executar uma divisão por 0!");

}

posicaoAtual++;

}

Repare que, agora, não importa quantos pacientes tivermos, iremos imprimir o IMC de todos eles apenas utilizando esse código! Porém, temos alguns códigos que iriam se repetir em qualquer loop que quisermos construir:

* a declaração da variável posicaoAtual (também conhecida como variável de controle)
* a condição do loop
* o incremento ao fim do loop.

Se não quisermos gastar 3 linhas no nosso código, uma para cada item, podemos utilizar o comando for, em vez do while. A diferença do comando for é que ele recebe três argumentos separados por ponto-e-vírgula:

for("declaracao" ; "condicao" ; "incremento"){

//bloco de código

}

Portanto, podemos substituir o while que construímos anteriormente por:

//monta pacientes

var pacientes = [leonardo, paulo];

for(var posicaoAtual = 0; posicaoAtual <= pacientes.length - 1; posicaoAtual++){

var pacienteAtual = pacientes[posicaoAtual];

if(pacienteAtual.altura != 0){

var imcDoPacienteAtual = pacienteAtual.peso / (pacienteAtual.altura \* pacienteAtual.altura);

console.log(imcDoPacienteAtual) // 25

}else{

console("Não posso executar uma divisão por 0!");

}

}

Bacana, já estamos mostrando os IMCs de todos os pacientes do Array! Mas essa coleção sempre terá dois elementos, pois estamos montando apenas dois pacientes e colocando dentro dele, o que não é muito útil, pois poderíamos ter mais pacientes em nossa tabela!

O que queremos agora é montar o nosso Array a partir dos dados da nossa tabela, para isso, vamos analisá-la:

<table>

<tr>

<th>Nome</th>

<th>Peso</th>

<th>Altura</th>

<th>IMC</th>

</tr>

<tr>

<td id="nome-1">Leonardo</td>

<td id="peso-1">50.0</td>

<td id="altura-1">1.60</td>

<td id="imc-1"></td>

</tr>

<tr>

<td id="nome-2">Paulo</td>

<td id="peso-2">100</td>

<td id="altura-2">2.00</td>

<td id="imc-2"></td>

</tr>

</table>

Note que cada paciente é representado por uma tag <tr>, o que poderíamos selecionar utilizando a funçãodocument.getElementsByTagName do seguinte modo:

var nossosTrs = document.getElementsByTagName("tr");

Isso nos devolveria um Array de objetos que representam nossas tags <tr>, mas espere! O primeiro <tr> de nossa tabela não é um paciente!

Para pegarmos somente os pacientes, poderíamos adicionar um atributo class para diferenciá-los:

<table>

<tr>

<th>Nome</th>

<th>Peso</th>

<th>Altura</th>

<th>IMC</th>

</tr>

<tr class="paciente">

<td id="nome-1">Leonardo</td>

<td id="peso-1">50.0</td>

<td id="altura-1">1.60</td>

<td id="imc-1"></td>

</tr>

<tr class="paciente">

<td id="nome-2">Paulo</td>

<td id="peso-2">100</td>

<td id="altura-2">2.00</td>

<td id="imc-2"></td>

</tr>

</table>

Note que nós não usamos id pois ele deve ser único na página, diferente de class.

Agora, para selecionar apenas o elementos que possuem class="paciente", podemos usar a funçãodocument.getElementsByClassName:

var nossosTrs = document.getElementsByClassName("paciente");

Bacana, agora precisamos montar nossos pacientes a partir desses <tr>, para isso precisamos percorrê-los utilizando o comando for que vimos anteriormente:

var nossosTrs = document.getElementsByClassName("paciente");

for(var posicaoDoTrAtual = 0; posicaoDoTrAtual <= nossosTrs.length - 1; posicaoDoTrAtual++){

var trAtual = nossosTrs[posicaoDoTrAtual];

//monta e adiciona paciente no array 'pacientes'

}

Agora só precisamos montar os pacientes a partir do trAtual, ou seja, pegar as informações(<td>) de dentro dotrAtual e criar um objeto a partir delas!

Para pegar um <td> dentro do trAtual, podemos chamar a função getElementsByTagName não em documentcomo antes, mas no trAtual, que é o elemento que queremos usar como raíz:

var nossosTrs = document.getElementsByClassName("paciente");

for(var posicaoDoTrAtual = 0; posicaoDoTrAtual <= nossosTrs.length - 1; posicaoDoTrAtual++){

var trAtual = nossosTrs[posicaoDoTrAtual];

var informacoes = trAtual.getElementsByTagName("td"); // pega todos os tds dentro do tr

}

Mas repare que apesar de termos pego todos os <td> dentro do trAtual, nós não temos ideia de qual é o peso, qual é o nome, etc.

Precisamos dar um jeito de buscar especificamente o <td> que precisamos, vamos então dar classes a eles:

<table>

<tr>

<th>Nome</th>

<th>Peso</th>

<th>Altura</th>

<th>IMC</th>

</tr>

<tr class="paciente">

<td class="info-nome">Leonardo</td>

<td class="info-peso">50.0</td>

<td class="info-altura">1.60</td>

<td class="info-imc"></td>

</tr>

<tr class="paciente">

<td class="info-nome">Paulo</td>

<td class="info-peso">100</td>

<td class="info-altura">2.00</td>

<td class="info-imc"></td>

</tr>

</table>

Agora podemos pegar o <td> com class="info-nome" facilmente utilizando otrAtual.getElementsByClassName("info-nome"). Mas lembre que o getElementsByClassName sempre retorna umArray, pois pode existir mais de um elemento com a mesma class, portanto pegaremos a primeira posição desseArray:

var nossosTrs = document.getElementsByClassName("paciente");

for(var posicaoDoTrAtual = 0; posicaoDoTrAtual <= nossosTrs.length - 1; posicaoDoTrAtual++){

var trAtual = nossosTrs[posicaoDoTrAtual];

var nomeTd = trAtual.getElementsByClassName("info-nome")[0]; // pega tr do nome do paciente atual

var pesoTd = trAtual.getElementsByClassName("info-peso")[0]; // pega tr do peso do paciente atual

var alturaTd = trAtual.getElementsByClassName("info-altura")[0]; // pega tr da altura do paciente atual

}

Agora podemos criar nosso objeto paciente:

var nossosTrs = document.getElementsByClassName("paciente");

for(var posicaoDoTrAtual = 0; posicaoDoTrAtual <= nossosTrs.length - 1; posicaoDoTrAtual++){

var trAtual = nossosTrs[posicaoDoTrAtual];

var nomeTd = trAtual.getElementsByClassName("info-nome")[0]; // pega tr do nome do paciente atual

var pesoTd = trAtual.getElementsByClassName("info-peso")[0]; // pega tr do peso do paciente atual

var alturaTd = trAtual.getElementsByClassName("info-altura")[0]; // pega tr do altura do paciente atual

var paciente = {

nome : nomeTd.textContent,

peso : pesoTd.textContent,

altura : alturaTd.textContent

}

}

Pronto! Agora nossos pacientes serão gerados dinamicamente para cada <tr> na página! Basta agora pegar o IMC do paciente que montamos e modificar o <td> de IMC do <tr> atual:

var nossosTrs = document.getElementsByClassName("paciente");

for(var posicaoDoTrAtual = 0; posicaoDoTrAtual <= nossosTrs.length - 1; posicaoDoTrAtual++){

var trAtual = nossosTrs[posicaoDoTrAtual];

var nomeTd = trAtual.getElementsByClassName("info-nome")[0]; // pega tr do nome do paciente atual

var pesoTd = trAtual.getElementsByClassName("info-peso")[0]; // pega tr do peso do paciente atual

var alturaTd = trAtual.getElementsByClassName("info-altura")[0]; // pega tr do altura do paciente atual

var imcTd = trAtual.getElementsByClassName("info-imc")[0]; // pega tr do imc do paciente atual

var paciente = {

nome : nomeTd.textContent,

peso : pesoTd.textContent,

altura : alturaTd.textContent

}

if(paciente.altura != 0){

var imcDoPaciente = paciente.peso / (paciente.altura \* paciente.altura);

imcTd.textContent = imcDoPaciente;

console.log(imcDoPaciente); // 25

}else{

console("Não posso executar uma divisão por 0!");

}

}