**Funções e os cidadãos de primeira classe**

Até agora nós conseguimos organizar uma coleção de objetos pacientes e calcular o IMC de cada um deles sem precisar copiar e colar o código, mas ainda temos alguns pontos a melhorar no nosso programinha.

A primeira coisa que poderíamos fazer é dar um nome para a nossa lógica que calcula IMC para que fosse mais fácil entender o que ela faz! Para isso usaremos um recurso do Javascript chamado Funções.

**O que são funções?**

Funções são estruturas que nos possibilitam isolar uma determinada lógica e dar um nome a ela, permitindo que a chamemos em vários pontos de nosso código! Elas ajudam muito organizar o nosso código e deixar ele mais legível. Vamos tentar?

No nosso caso, a lógica que queremos isolar é a de calculo de IMC. Dentro do arquivo calcula-imc.jsencontramos essa lógica:

if(paciente.altura != 0){

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

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}else{

console("Não posso dividir por zero!");

}

Para declarar uma função, basta utilizarmos a palavra function seguido do nome da função e um corpo, que seria a lógica que desejamos isolar.Portanto, podemos criar uma função chamada calculaImc com esse corpo (bem no início do arquivo calcula-imc.js):

//arquivo calcula-imc.js

function calculaImc(){

if(paciente.altura != 0){

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

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}else{

console("Não posso dividir por zero!");

}

}

**Observação**: *Existe também uma outra forma de se criar uma função, que é atribuindo-a a uma variável:*

var calculaImc = function(){

// corpo da função

}

*Neste curso, ficaremos com a primeira opção.*

Vamos invocar essa função onde estava a lógica que isolamos, isto é chamar a função dentro do for. Para isso basta digitar o nome da função que definimos seguido de ():

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var paciente = { nome : tdNome.textContent, peso : tdPeso.textContent, altura : tdAltura.textContent};

calculaImc(); invoca a função que declaramos anteriormente

}

Mas note que, na nossa função, nós estamos calculando o imc usando uma variável paciente que não existe, ou seja, não é o pacienteAtual!

Para calcular o imc de um paciente diferente para cada chamada da função, precisamos dizer que nossa função recebe um parâmetro que é um paciente e usá-lo dentro da função:

function calculaImc(paciente){

if(paciente.altura != 0){

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

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

} else{

console("Não posso dividir por zero!");

}

}

Agora, quem definirá quem é esse paciente é o código que invoca essa função. No nosso caso, o paciente será opacienteAtual, portanto,vamos passá-lo como argumento na chamada da função:

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

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

//renomeamos a variaval paciente para pacienteAtual

var pacienteAtual = { nome : tdNome.textContent, peso : tdPeso.textContent, altura : tdAltura.textContent};

calculaImc(pacienteAtual); //passando o pacienteAtual

}

Agora temos um código mais legível com o mesmo comportamento!

Mas note que a nossa função, além de estar calculando o imc(que é o que o nome dela diz), ela *também está modificando a variável* imcTd. Além de não existir dentro da função, estamos fazendo mais coisas do que deveríamos dentro dela! Boa prática é que uma função sabe fazer uma coisa bem!

O que queríamos era somente calcular o IMC, então nossa função seria somente a seguinte:

function calculaImc(paciente){

if(paciente.altura != 0){

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

//sem alteração do tdImc e console.log

} else{

console("Não posso dividir por zero!");

}

}

Mas dessa maneira, nós não estamos mais sendo informados de qual é o IMC de cada paciente!

Seria legal se continuássemos a imprimir o IMC dos pacientes e modificar o tdImc, mas fora da função! Para isso ser possível, precisamos dizer que nossa função irá *retornar* o IMC calculado:

function calculaImc(paciente){

if(paciente.altura != 0){

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

return imc; //retornando imc

} else{

console("Não posso dividir por zero!");

}

}

Deste modo, podemos atribuir o retorno dessa função a uma variável quando a invocarmos dentro do for:

var imc = calculaImc(pacienteAtual)

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

Então, todo nosso código para calcular o imc de nossos pacientes seria o seguinte (conteúdo completo do arquivocalcula-imc.js):

function calculaImc(paciente){

if(paciente.altura != 0){

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

return imc;

} else{

console("Não posso dividir por zero!");

}

}

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

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = { nome : tdNome.textContent, peso : tdPeso.textContent, altura : tdAltura.textContent};

var imc = calculaImc(pacienteAtual);

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}

**Fim do video 5.1**

**Adicionando comportamentos a objetos**

Perceba que a função que criamos, calculaImc, é uma função global. Ou seja, podemos chamar ela em qualquer lugar. Mas faz sentido ela ser? De quem ela calcula o IMC? Do paciente, certo?

Se ela calcula o IMC do paciente, seria legal se esse fosse um comportamento do próprio paciente, e isso é possível! Podemos adicionar esse comportamento ao paciente simplesmente adicionando ao objeto:

function calculaImc(paciente){

if(paciente.altura != 0){

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

return imc;

} else{

console("Não posso dividir por zero!");

}

}

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

//codigo omitido

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: calculaImc //atribuindo a função ao atributos pegaImc

};

//codigo omitido

}

Deste modo, para chamarmos a função do paciente, bastaria utilizarmos o nome que definimos *no objeto*:

var imc = pacienteAtual.pegaImc(pacienteAtual);

console.log(imc);

Mas esse código ainda está meio estranho: estamos chamando a função pegaImc do objeto pacienteAtualpassando o mesmo pacienteAtual como parâmetro! O ideal seria que nossa função soubesse que o paciente que queríamos usar é o mesmo que usamos para chamá-la e, assim, parar de receber um paciente como parâmetro:

function calculaImc(){ //sem parâmetro, mas funciona assim?

if(paciente.altura != 0){

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

return imc;

} else{

console("Não posso dividir por zero!");

}

}

Mas e agora? Nós estamos usando uma variável paciente que não existe dentro da função! O que queremos é usar *esse* paciente, o que usamos para chamar a função. Para isso usaremos a palavra chave this:

function calculaImc(){ //sem parâmetro, mas usando this

if(paciente.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

Agora a chamada de método parece fazer mais sentido, dentro do for:

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

//codigo omitido

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: calculaImc //atribuindo a função ao atributos pegaImc

};

var imc = pacienteAtual.pegaImc(); //sem param!!!

//codigo omitido

}

Deste modo, nosso código final, para montar todos os pacientes, fica:

function calculaImc(){

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

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

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: calculaImc

};

var imc = pacienteAtual.pegaImc();

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}

Mas note que continuamos declarando uma função que pode ser acessada de qualquer lugar, a calculaImc, e só utilizamos ela para colocar no paciente! Alias, chamando essa função diretamento nem vai funcionar pois ela precisa de um objeto paciente.

Para evitar essa declaração, poderíamos colocá-la diretamente no objeto, da seguinte maneira:

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

//codigo omitido

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() { //declarando uma função anônima

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

var imc = pacienteAtual.pegaImc(); //sem param!!!

//codigo omitido

}

Note que agora o nome da função é claramente o nome do atributo, ou seja, pegaImc, e só é acessível através do objeto paciente! Agora você pode apagar a função calculaImc()!

Mostrando uma vez o código completo do arquivo calcula-imc.js:

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

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

var imc = pacienteAtual.pegaImc();

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}

**Fim do video 5.2**

**Passando comportamentos para funções**

Imagine que, além de calcular todos os IMCs de nossos pacientes, em outro ponto do sistema queremos mostrar todos os nomes de nossos pacientes.

Como vamos querer executar um mesmo código com todos os pacientes, podemos criar um novo arquivo imprime-nome.js e copiar *todo o código* para imprimir apenas o nome do pacienteAtual no fim do laço:

//arquivo imprime-nome.js

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

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

var pacienteTr = trsPacientes[posicaoAtual];

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

//só imprimindo nome do pacienteAtual

console.log(pacienteAtual.nome);

}

Para testar o novo arquivo js devemos incluí-lo no arquivo index.html, lá bem no final da página:

<script src="imprime-nome.js"></script>

Uma vez adicionado o script na página index.html você já pode testar no navegador. Os nomes dos pacientes devem aparecer na console do navegador.

**Código repitido**

Repare que nosso código possui dois fors idênticos no momento, um no arquivo calcula-imc.js:

//arquivo calcula-imc.js

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

// monta pacienteAtual

var imc = pacienteAtual.pegaImc();

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}

e outro no arquivo imprime-nome.js:

//arquivo imprime-nome.js

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

// monta pacienteAtual

console.log(pacienteAtual.nome);

}

Estamos copiando e colando código, o que nunca é bom! Para resolver, podemos pensar em criar uma função que executa a parte repetida! Vamos criar um novo arquivo percorre.js com uma função que percorre o array detrs:

//arquivo percorre.js

function percorreArray(trsPacientes){

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

// executa algum código

}

}

E na nossa lógica anterior, bastaria chamar essa função passando o nosso Array como argumento trsPacientes:

percorreArray(trsPacientes);

percorreArray(trsPacientes);

Mas a nossa função que percorre o Array não está executando nenhum código dentro dela! O que eu queria é que, a cada vez que eu chamasse a função percorreArray, eu tivesse de passar um *comportamento* que seria executado em algum ponto dentro dessa função.

Então a ideia é seguinte: a função percorreArray recebe além do array (trsPacientes) um outro argumento que define o código para executar! Muita calma nessa hora :)

Em Javascript, nós envolvemos nossos comportamentos em funções, certo? Então vamos criar uma função que imprime seu imc e outra que imprime o nome de um paciente:

//imprime o imc do paciente

function imprimeEModificaTdDeImc(trAtual){

//monta paciente como visto

var imc = paciente.pegaImc(); // invoca a função no pacienteAtual

imcTd.textContent = imc; // atualiza imc

console.log(imc); // imprime o imc do paciente atual

}

//apenas loga o nome do paciente

function imprimeNome(trAtual){

//monta paciente como visto

console.log(paciente.nome);

}

Legal, agora o que queremos é, na nossa função percorreArray, receber um parâmetro que é esse comportamento, ou seja uma dessas funções:

function percorreArray(trsPacientes, comportamento){

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

// executa algum código

}

}

E para executar esse comportamento, basta invocá-lo!(lembre-se, comportamentos são funções em Javascript)

//arquivo percorre.js

function percorreArray(trsPacientes, comportamento) {

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

comportamento();

}

}

Agora temos nosso for isolado em apenas uma função, que recebe um comportamento para ser utilizado dentro deste loop! Vamos ver como está o nosso código até agora:

Note que, apesar de recebermos um comportamento como argumento na nossa função percorreArray, nós não passamos nenhum na chamada desta.

O que queremos é que: na primeira chamada o comportamento seja imprimeNome e na segunda sejaimprimeEModificaTdDeImc, então vamos passar estas funções como parâmetro:

percorreArray(pacientes, imprimeNome);

percorreArray(pacientes, imprimeEModificaTdDeImc);

Agora, dentro da função percorreArray, estamos chamando as funções imprimeNome(na primeira chamada) eimprimeEModificaTdDeImc(na segunda chamada):

//arquivo percorre.js

function percorreArray(trsPacientes, comportamento) {

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

comportamento(); //imprimeNome ou imprimeEModificaTdDeImc

}

}

Mas lembre que nossas funções imprimeNome e imprimeEModificaTdDeImc, precisam de um paciente para funcionar! Para isso, no nosso for, temos que passar o pacienteAtual como parâmetro para o comportamento:

//arquivo percorre.js

function percorreArray(trsPacientes, comportamento) {

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

var pacienteTrAtual = trsPacientes[posicaoAtual];

comportamento(pacienteTrAtual);

}

}

Agora já podemos testar o código nosso código inteiro, veja cada arquivo!

Arquivo percorre.js que possui a função que faz o loop do array trsPacientes e chamandoa função que recebe pelo segundo argumento (comportamento):

//arquivo percorre.js

function percorreArray(trsPacientes, comportamento) {

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

var pacienteTrAtual = trsPacientes[posicaoAtual];

comportamento(pacienteTrAtual);

}

}

O arquivo imprime-nome.js que busca os trsPacientes, define a função imprimeNome e chamaa funçãopercorreArray:

//arquivo imprime-nome.js

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

percorreArray(trsPacientes, imprimeNome);

function imprimeNome(pacienteTr) {

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

console.log(pacienteAtual.nome);

}

E o arquivo calcula-imc.js que também busca os trsPacientes, define a função imprimeEModificaTdDeImc e chama a função percorreArray:

//calcula-imc.js

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

percorreArray(trsPacientes, imprimeEModificaTdDeImc);

function imprimeEModificaTdDeImc(pacienteTr){

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

var imc = pacienteAtual.pegaImc();

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}

No Javascript é bem comum passarmos funções como parâmetros dessa maneira, o que é diferente de algumas linguagens de programação, vale a pena se acostumar!

Já podemos testar esse código, mas não podemos esquecer de importar o arquivo percorre.js napáginaindex.html. Seguem uma vez todos os imports da página. Importante é que o percorre.jsvem no início pois os outros dois arquivos depende dele:

<script src="percorre.js"></script>

<script src="calcula-imc.js"></script>

<script src="imprime-nome.js"></script>

Funcionou?

**Funções anônimas**

Perceba que no nosso código, nós temos duas funções que só são usadas para passar um comportamento para outra função, a imprimeNome e a imprimeEModificaTdDeImc.

Como nós não temos a intenção de reaproveitar essas funções em outros pontos do nosso código, não precisamos declará-las! Podemos passar uma função diretamente para outra(como a percorreArray) da seguinte maneira:

percorreArray(trsPacientes, function(){

// o que quiser fazer aqui

});

Note que, o que fizemos foi passar a função diretamente, sem declará-la antes e *sem dar um nome a ela*, por isso essas funções são chamadas de *funções anônimas*. Vamos então utilizar funções anônimas no lugar daimprimeNome:

percorreArray(trsPacientes, function(){

//codigo que monta paciente omitido

console.log(pacienteAtual.nome); // imprime o nome do paciente atual

});

Mas note que, desse modo, a função anônima não terá nenhum paciente para acessar, precisamos então receber um como parâmetro, como fazíamos na antiga imprimeNome:

percorreArray(trsPacientes, function(pacienteAtual){

//codigo que monta paciente omitido

console.log(pacienteAtual.nome); // imprime o nome do paciente atual

});

Basta fazer o mesmo para a imprimeEModificaTdDeImc! Assim temos o seguinte resultado:

O percorre.js que não mudou:

//arquivo percorre.js

function percorreArray(trsPacientes, comportamento) {

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

var pacienteTrAtual = trsPacientes[posicaoAtual];

comportamento(pacienteTrAtual);

}

}

O arquivo imprime-nome.js que usa uma função anônima

//arquivo imprime-nome.js

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

//agora usando uma função anonima (função sem nome)

percorreArray(trsPacientes, function(pacienteTr){

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

console.log(pacienteAtual.nome);

}); //muito cuidado pra não esquecer de fechar aqui a chamada da função

O arquivo calcula-imc.js que usa também pode utilizar uma função anônima:

//calcula-imc.js

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

percorreArray(trsPacientes, function (pacienteTr){

var tdNome = pacienteTr.getElementsByClassName("info-nome")[0];

var tdPeso = pacienteTr.getElementsByClassName("info-peso")[0];

var tdAltura = pacienteTr.getElementsByClassName("info-altura")[0];

var pacienteAtual = {

nome : tdNome.textContent,

peso : tdPeso.textContent,

altura : tdAltura.textContent,

pegaImc: function() {

if(this.altura != 0){

var imc = this.peso / (this.altura \* this.altura);

return imc;

} else{

console("Não posso dividir por zero!");

}

}

};

var imc = pacienteAtual.pegaImc();

var tdImc = pacienteTr.getElementsByClassName("info-imc")[0];

tdImc.textContent = imc;

console.log(imc);

}); //muito cuidado pra não esquecer de fechar aqui a chamada da função

Como resultado temos um código mais conciso e elegante!

Para deixar completo, tbm segue o arquivo index.html:

<!doctype html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Meus pacientes</title>

<link rel="stylesheet" type="text/css" href="reset.css">

<link rel="stylesheet" type="text/css" href="alura.css">

</head>

<body>

<header>

<div class="container">

<h1>Meus pacientes</h1>

</div>

</header>

<main class="container">

<section>

<h2>Meus pacientes</h2>

<table>

<tr>

<th>Nome</th>

<th>Peso(kg)</th>

<th>Altura(m)</th>

<th>IMC</th>

</tr>

<tr class="paciente">

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

<td class="info-peso" id="peso-1">57</td>

<td class="info-altura" id="altura-1">1.67</td>

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

</tr>

<tr class="paciente">

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

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

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

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

</tr>

<tr class="paciente">

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

<td class="info-peso" id="peso-2">120</td>

<td class="info-altura" id="altura-2">1.80</td>

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

</tr>

<tr class="paciente">

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

<td class="info-peso" id="peso-2">110</td>

<td class="info-altura" id="altura-2">1.60</td>

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

</tr>

</table>

<button id="calcula-imcs" class="botao ">Calcular Imcs</button>

</section>

<section>

<h2>Adicionar novo paciente</h2>

<form>

<fieldset>

<label for="nome">Nome:</label>

<input id="campo-nome" type="text" placeholder="digite o nome do seu paciente">

</fieldset>

<fieldset class="campo-medio">

<label for="peso">Peso:</label>

<input id="campo-peso" type="text" placeholder="digite o peso do seu paciente">

</fieldset>

<fieldset class="campo-medio">

<label for="altura">Altura:</label>

<input id="campo-altura" type="text" placeholder="digite a altura do seu paciente">

</fieldset>

<button id="adicionar-paciente" class="botao bto-principal">Adicionar</button>

</form>

</section>

</main>

<script src="percorre.js"></script>

<script src="calcula-imc.js"></script>

<script src="imprime-nome.js"></script>

</body>

</html>