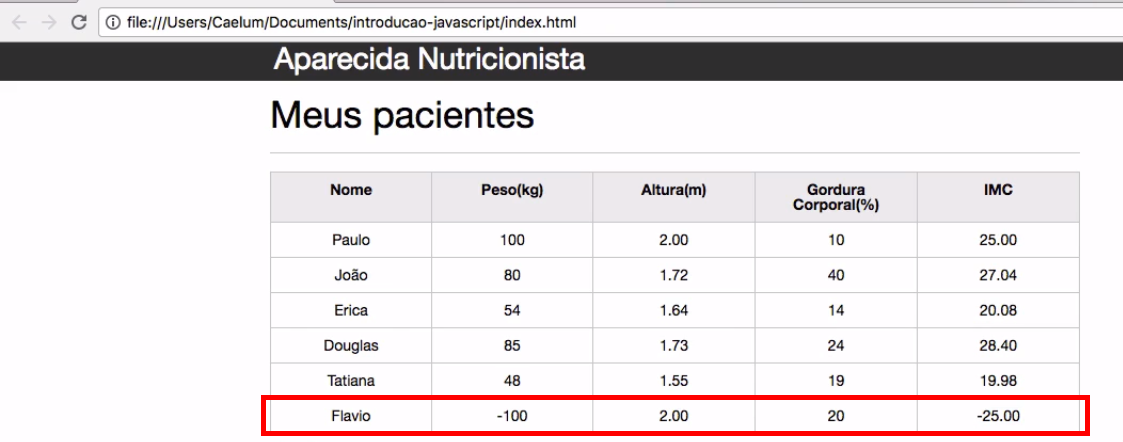
**01 Validando os dados do formulário**

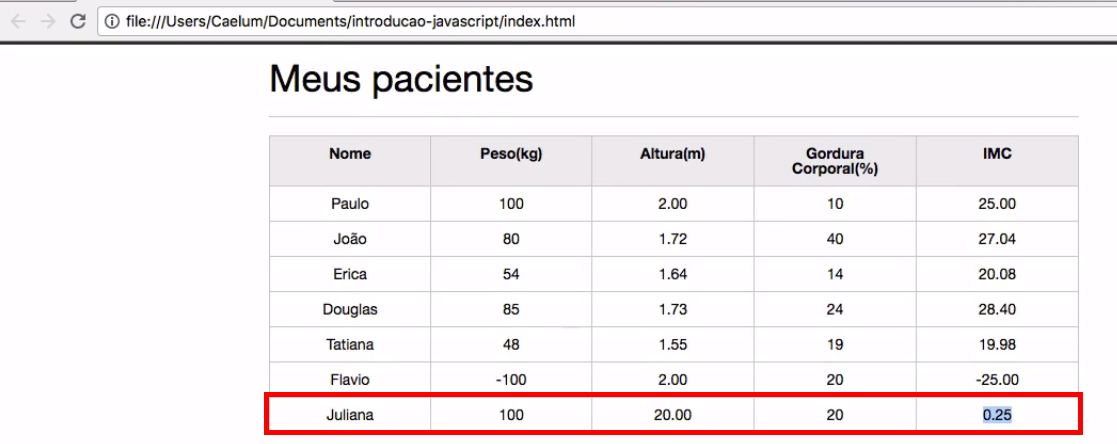
**Transcrição**

O último capítulo foi dedicado à organização do nosso código, principalmente da parte referente ao formulário, que já está adicionando pacientes.

Tivemos um grande trabalho para criar o código de validação do peso e da altura, porém, quando inserimos os dados de um paciente pelo formulário, se preenchermos com um valor negativo no campo de "Peso", o IMC será calculado com um valor negativo.



Não gostaríamos que isso acontecesse. Da mesma forma, se adicionássemos um valor absurdo de altura, como 20metros, o cálculo do IMC também será realizado.



Atualmente as validações implementadas são feitas apenas com pacientes que já estavam na tabela. O ideal é reaproveitar o código de validação, para também validar os dados do formulário antes de adicionarmos o paciente na tabela. Caso os dados sejam inválidos, exibiremos uma mensagem informando qual deles contém um valor inapropriado.

Como trata-se de uma validação dos dados do formulário, devemos fazê-la no form.js. Nós queremos fazer a validação do paciente antes de adicioná-lo à tabela, caso o valor seja válido. A seguir, criaremos a função chamada validaPaciente(), que receberá como parâmetro a variável paciente, cujos valores estarão preenchidos com os dados do formulário graças à função obtemPacienteDoFormulario(form):

function obtemPacienteDoFormulario(form){

var paciente = {

nome: form.nome.value,

peso: form.peso.value,

altura: form.altura.value,

gordura: form.gordura.value,

imc: calculaImc(form.peso.value, form.altura.value)

}

return paciente;

}

A reutilização do código se mostra útil, por termos o objeto paciente montado e pronto para ser utilizado no validaPaciente(paciente). Ela será responsável por testar as propriedades com peso e altura para identificar se os valores são válidos. Começaremos criando um if:

function validaPaciente(paciente){

if(paciente.altura < 3.0 && paciente.altura >= 0){

return true;

}

}

Criamos a condição de que se a altura for menor que 3 e maior ou igual a 0, o valor será válido. Porém, esta validação dos dados já foi realizada em calcula-imc.js:

var tdImc = paciente.querySelector(".info-imc");

var pesoEhValido = true;

var alturaEhValida = true;

if(peso <= 0 || peso >= 1000){

console.log("Peso inválido!");

pesoEhValido = false;

tdImc.textContent = "Peso inválido!";

paciente.classList.add("paciente-invalido");

}

if(altura <= 0 || altura >= 3.00){

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida!";

paciente.classList.add("paciente-invalido");

}

If(alturaEhValida && pesoEhValido){

var imc = calculaImc(peso,altura);

tdImc.textContent = imc;

}

Queremos reaproveitar esta lógica para facilitar inclusive a manutenção do código. Para não corrermos o risco de rescrevermos um código já existente, extrairemos a lógica de verificação dos dados para uma função específica, validaPeso(). Ela receberá peso como parâmetro e retornará true se o valor estiver entre 0 e 1000. Se não estiver, o retorno será false:

function validaPeso(peso){

if (peso >= 0 && peso <= 1000) {

return true;

} else {

return false;

}

}

Do mesmo jeito, vamos criar a função validaAltura():

function validaAltura(altura) {

if (altura >= 0 && altura <= 3.0) {

return true;

} else {

return false;

}

}

Em vez de inicializarmos as variáveis pesoEhValido e alturaEhValida com true, chamaremos as funções de validação.

var pesoEhValido = validaPeso(peso);

var alturaEhValida = validaAltura(altura);

Teremos que fazer ajustes no if também. No entanto, como estamos testando se o peso **não** é válido, para identificar se ele é negativo ou maior que uma tonelada, então vamos utilizar o **operador de negação** (!) no calcula-imc.js. O ! será responsável por inverter os valores da condição avaliada:

if (!pesoEhValido) {

console.log("Peso inválido!");

pesoEhValido = false;

tdImc.textContent = "Peso inválido!";

paciente.classList.add("paciente-invalido");

}

Vamos testar se o código continua funcionando corretamente, alterando no **HTML** o valor do peso do paciente para um valor inválido, por exemplo, 10000.



Em seguida, repetiremos o processo para a função validaAltura(altura), que receberá a altura e fará um teste semelhante dentro do if:

if (!alturaEhValida) {

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida!";

paciente.classList.add("paciente-invalido");

}

Testando com um valor inválido de altura, veremos que tudo funciona corretamente. Agora, com essas duas funções criadas, poderemos chamá-las na função validaPaciente(paciente), em form.js. Vamos testar se o peso e a altura são válidos utilizando as funções que acabamos de criar. Por ora, testaremos apenas com o peso:

function validaPaciente(paciente) {

if (validaPeso(paciente.peso)) {

return true;

} else {

return false;

}

}

O próximo passo será adicionar a função validaPaciente(paciente) no momento do clique do botão após a montagem do objeto paciente. Incluiremos uma condição, na qual vamos imprimir uma mensagem de aviso no console se o valor do validaPaciente() for false.

var botaoAdicionar = document.querySelector("#adicionar-paciente");

botaoAdicionar.addEventListener("click", function(event) {

event.preventDefault();

var form = document.querySelector("#form-adiciona");

var paciente = obtemPacienteDoFormulario(form);

if (!validaPaciente(paciente)) {

console.log("Paciente inválido");

}

var pacienteTr = montaTr(paciente);

var tabela = document.querySelector("#tabela-pacientes");

tabela.appendChild(pacienteTr);

form.reset();

});

Além disso, caso o paciente tenha algum valor inválido, queremos que ele não seja adicionado na tabela. Então devemos sair da função caso a condição do if seja válida. Ou seja, se o código entrar no if, exibiremos a mensagem de paciente inválido no console e **retornaremos** a função, adicionando-se o return vazio:

var form = document.querySelector("#form-adiciona");

var paciente = obtemPacienteDoFormulario(form);

var pacienteTr = montaTr(paciente);

if (!validaPaciente(paciente)) {

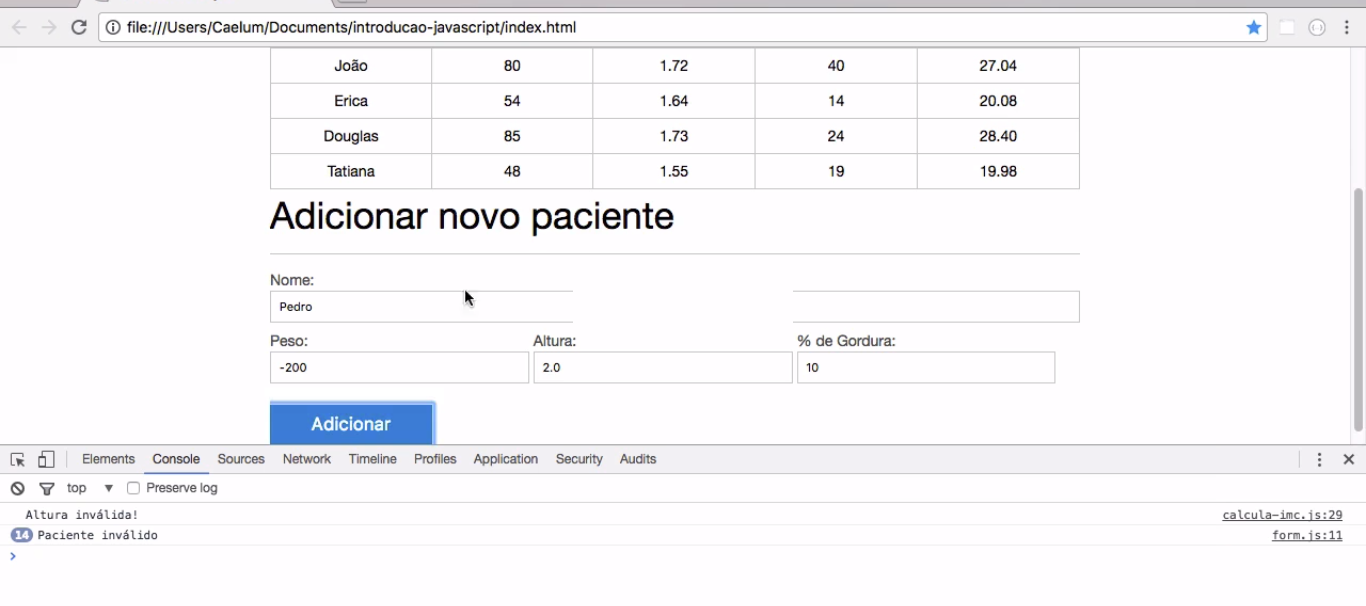
console.log("Paciente inválido");

return;

}

//...

Após o return sairemos da função sem executar o código abaixo dele, responsável por adicionar o paciente na tabela. Em seguida, tentaremos cadastrar um paciente com um valor de peso inválido. Ao clicarmos no botão "Adicionar", o paciente não será adicionado na tabela. Quando abrirmos o console, veremos a mensagem sobre o paciente inválido.



Resolvemos o problema do peso, mas além de mostrarmos uma mensagem no console, seria interessante exibirmos uma mensagem no HTML informando para o usuário qual dado está com problema. Queremos também validar a altura do paciente, o que faremos mais adiante.