**02Por que aprender JavaScript?**

**Transcrição**

Seja bem-vindo ao curso **JavaScript: Programando na linguagem da web**, eu sou o [Douglas Quintanilha](https://twitter.com/dquintanilhas) e serei seu instrutor no curso que te dará uma introdução a essa linguagem tão importante atualmente.

Você pode estar se perguntando: "por que se comenta tanto sobre JavaScript hoje em dia? Por que essa linguagem ganhou tanta popularidade, e por que eu, desenvolvedor, tenho que aprendê-la?"

Para iniciarmos nossa conversa sobre JavaScript, vale avisar que ela é a **linguagem de programação nativa da Web**. Qualquer navegador que você esteja utilizando, incluindo o Google Chrome, com certeza dá suporte ao JS, afinal ele é utilizado para dar dinamismo a nossas páginas. Por exemplo, quando clicamos na ferramenta de busca do site da [Alura](https://www.alura.com.br/), o fato de que seja detectado o evento de clique no ícone da lupa e, em seguida, ocorra uma pequena animação para exibir a caixa de busca, é realizável com JS.

Talvez você tenha conhecimento maior sobre HTML e CSS e garanta que seja possível fazer o mesmo usando o transform, já que também podemos criar algumas animações apenas com CSS puro. Porém, certas ações como algumas validações específicas nos formulários, como aquelas realizadas nos campos de "nome" e "e-mail", não são possibilitadas unicamente com HTML. Esse tipo de dinamismo e interatividade que temos atualmente nos formulários e nas páginas Web vieram com o JavaScript.

Graças ao JS, aumentamos a usabilidade do usuário nas páginas web. Um exemplo é no fórum do [GUJ](http://www.guj.com.br/): quando clicamos em alguma categoria no menu, ele filtrará os tópicos do fórum. Isso só é possível com JavaScript.

É uma linguagem bastante poderosa que permite que o usuário consiga **interagir** com a página. Isto já seria um excelente motivo para despertar seu interesse pela linguagem, mas além de ser dominante nos cenários dos navegadores, o JavaScript recentemente também se expandiu para o lado do servidor com o lançamento do **Node.js**, um ambiente que permite criarmos um servidor Web completo utilizando JavaScript. Desta forma, a linguagem ganhou grande popularidade e caiu no gosto dos desenvolvedores, tornando-se parte do coletivo imaginário de qualquer programador moderno.

O Javascript não "ganhou terreno" apenas no servidor, sendo possível programarmos um Arduíno com a biblioteca [***Johnny-Five***](http://johnny-five.io/), criarmos um aplicativo para Desktop utilizando o framework [***Electron***](https://electron.atom.io/), e até mesmo encontramos um banco de dados como o [***MongoDB***](https://www.mongodb.com/), que utiliza a sintaxe do JavaScript em suas *queries*.

O JavaScript está presente em diversas áreas do desenvolvimento, ampliando as habilidades de desenvolvedor para os mais diversos ramos da tecnologia, e é claro, controlando e manipulando qualquer página Web!

Estudaremos sua sintaxe e veremos como a linguagem funciona, quais são suas nuances, a partir de um pequeno projeto para explorar o JavaScript em sua principal área de atuação, no Front-end, e na manipulação de páginas Web.

**03Por que estudar Javascript?**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23556/next)

O Javascript é uma das linguagens mais importantes de nosso tempo porque:

Parte superior do formulário

* Alternativa correta



é uma linguagem básica de programação, que por ser simples deve ser a linguagem inicial de todos os desenvolvedores.

* Alternativa correta



é uma linguagem que, além de ser a **padrão** nos navegadores atuais, vem **expandindo** para outras áreas como servidores, banco de dados e até plataformas de Internet of Things.

Correta

* Alternativa correta



é uma linguagem semelhante a linguagem Java, que é muito popular, e que deve ser aprendida para aumentar o seu conhecimento em Java.

Parte inferior do formulário

O Javascript cresceu bastante e hoje em dia pode ser encontrado nas mais diversas plataformas de desenvolvimento. Dominá-la é essencial se você deseja desenvolver sites para a Web, já que ela é a linguagem que funciona em todos os navegadores e responsável por dar interatividade e dinamismo as páginas web.

**Olá Mundo com JavaScript**

**Transcrição**

Para iniciarmos nosso aprendizado em JavaScript, construiremos o projeto do site **Aparecida Nutrição**.

O primeiro passo será baixar o projeto inicial do treinamento, disponível nos [exercícios](https://s3.amazonaws.com/caelum-online-public/introducao-javascript/stages/introducao-javascript-inicial.zip). Após descompactá-lo, acesse a pasta introducao-javascript do seu computador, e depois abra o arquivo no editor de textos. Nós vamos abri-lo também no browser, para nos familiarizarmos com o site.



A nutricionista Aparecida fazia controle dos clientes com fichas cadastrais, preenchidas manualmente, nas quais eram armazenados dados como "Nome", "Peso", "Gordura Corporal" e outros. Com a criação do site, ela conseguirá armazenar as informações, a serem exibidas em uma tabela, em que novos cadastros serão adicionados dinamicamente por meio do JavaScript, e poderão ser filtrados posteriormente. Ela também não precisará mais fazer cálculos manualmente, como os de IMC e da porcentagem de gordura corporal dos clientes. Todos serão realizados utilizando-se o JavaScript. Criaremos um site que realmente facilitará o trabalho da Aparecida!

Começaremos criando a função que exibirá a mensagem "Hello World", ou "Olá Mundo" no JS.

No curso, utilizaremos o editor de textos [**Atom**](https://atom.io/), mas existem diversos outros como o [**Sublime Text 3**](https://www.sublimetext.com/) e o [**Visual Studio Code**](https://code.visualstudio.com/). Escolha de acordo com sua preferência.

Adicionaremos a funcionalidade e o dinamismo na página de pacientes, e para isto precisaremos primeiro aprender como utilizarmos o JavaScript corretamente nas páginas. Uma primeira tentativa consiste em utilizarmos as funções dentro do corpo de nossa página.

No início do arquivo index.html, logo após a tag **<body>**, exibiremos um alerta com JavaScript, com a função alert:

<head>

<meta charset="UTF-8">

<title>Aparecida Nutrição</title>

<link rel="icon" href="favicon.ico" type="image/x-icon">

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

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

</head>

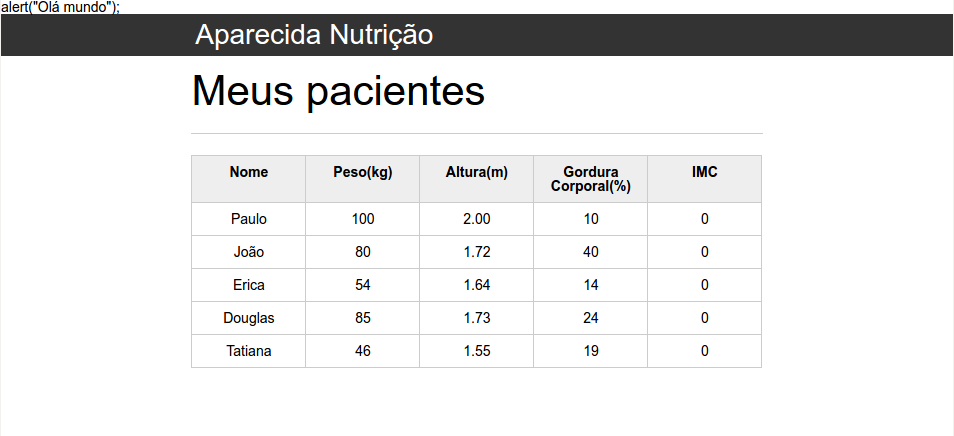
<body>

alert("Olá mundo");

<!-- Restante do index.html -->

</body>COPIAR CÓDIGO

Ao abrirmos a página index.html no navegador, veremos o seguinte resultado:



O script está sendo exibido como texto, ou seja, o navegador não fez nenhum tratamento especial para interpretar o código. Precisaremos adicionar a tag <script>, mas o JavaScript não será carregado. Iremos então adicioná-lo dentro da tag <head>:

<head>

<meta charset="UTF-8">

<title>Aparecida Nutrição</title>

<link rel="icon" href="favicon.ico" type="image/x-icon">

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

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

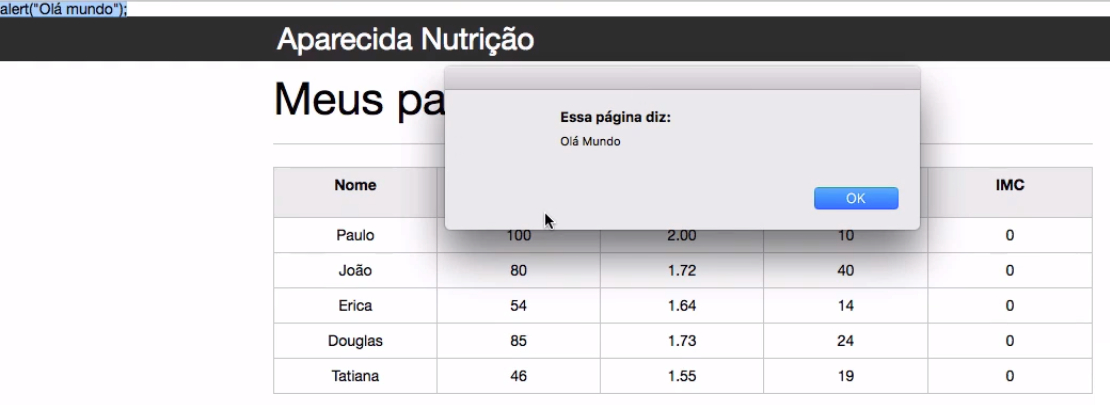
<script>

alert("Olá mundo");

</script>

</head> COPIAR CÓDIGO

Agora, quando recarregarmos a página no navegador, nosso trecho de script será interpretado como um código, e não como um conteúdo qualquer da página.



O alerta com a mensagem foi exibido!

A função alert é uma das opções para se exibir mensagens com JavaScript, porém, ela não é a única. Outra função útil para isto é o console.log, considerado "mais agradável" para a exibição de uma mensagem por não envolver um pop-up. A mensagem será exibida no **console do navegador**.

Vamos testá-la:

<head>

<meta charset="UTF-8">

<title>Aparecida Nutrição</title>

<link rel="icon" href="favicon.ico" type="image/x-icon">

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

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

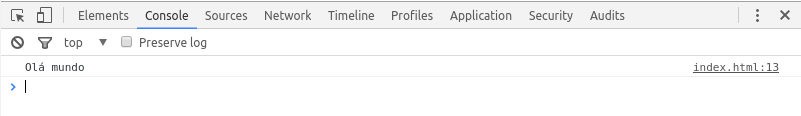
<script>

console.log("Oi Mundo");

</script>

</head> COPIAR CÓDIGO

No Linux ou no Windows, acessamos o console do navegador pressionando a tecla "F12"; no MacOS basta pressionar "COMMAND + ALT + J". Ao abrirmos o console, veremos a mensagem como gostaríamos:

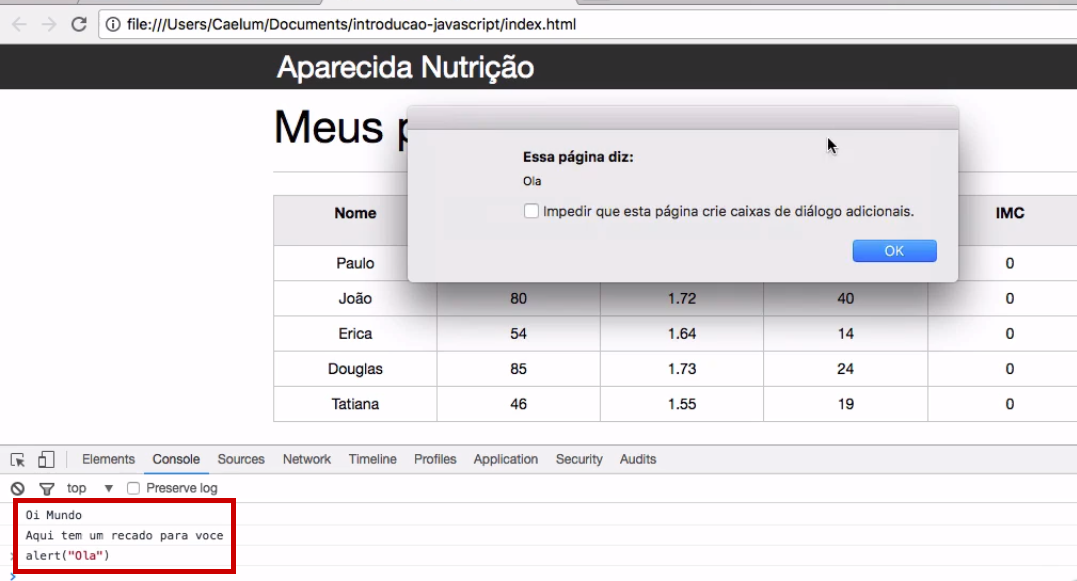


Além disso, o console do navegador também aceita funções JavaScript. Poderemos escrever novamente console.log:

console.log("Aqui tem um recado para você");COPIAR CÓDIGO

Ou até mesmo um alerta:

alert("Olá");COPIAR CÓDIGO



Com o **alert** e o **console.log**, damos os primeiros passos com o JavaScript, entendendo que para seu bom funcionamento, o código deve estar entre as tags <script>. Caso contrário, os navegadores não irão entendê-lo.

**05Interpretando scripts no navegador**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23557/next)

Sabemos que o navegador entende apenas HTML e que se colocarmos scripts diretamente no corpo da mensagem eles serão interpretados como texto. Como resolvemos isso?

Parte superior do formulário

* Alternativa correta



Não há como resolver, já que o navegador interpreta tudo como HTML e não como script.

* Alternativa correta



Através da tag <javascript></javascript>. Tudo que estiver dentro dela será interpretado como script e não texto.

* Alternativa correta



Através da tag <script></script>. Tudo que estiver dentro dela será interpretado como script e não texto.

Correto, a tag script possibilita que tudo que estiver dentro dela seja interpretado com um script de Javascript.

* Alternativa correta



Através da tag <head></head>. Tudo que estiver dentro dela será interpretado como script e não texto.

Parte inferior do formulário

A tag **<script>** é responsável por nos ajudar a resolver este problema. Tudo que declaramos dentro da tag <script> é interpretado como Javascript quando o navegador passar por ali.

No caso abaixo:

<script>

alert("Olá mundo");

</script>COPIAR CÓDIGO

Quando o navegador passar pela tag abaixo ele vai interprar o texto "alert("Olá mundo");" e irá executar a função **alert()**, que é responsável por exibir uma pop up com a mensagem Olá mundo.

**06Uma mensagem no navegador**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23575/next)

Ao abrir o HTML abaixo em um navegador, o que irá acontecer ?

<html>

<head>

</head>

<body>

<h1>Minha página com Javascript</h1>

<script>

Olá mundo!

</script>

</body>

</html>COPIAR CÓDIGO

Parte superior do formulário

* Alternativa correta



Será exibido um erro no console e nada na tela.

* Alternativa correta



Será exibido o <h1> com letras grandes **e** a mensagem Olá mundo com letras menores logo abaixo.

* Alternativa correta



Será exibido uma mensagem Olá mundo no console e o conteúdo do <h1> na tela.

* Alternativa correta



Será exibido **apenas** o texto do <h1> e um erro no console do desenvolvedor.

Sabemos que o navegador vai lendo de cima para baixo o conteúdo do HTML e interpretando o que ele lê de acordo com as tags da página. Neste caso, ele irá ler a tag <h1> e exibirá o seu conteúdo o tamanho da fonte maior, característico de um <h1>.

Parte inferior do formulário

No entanto, quando ler a tag <script>, o navegador entende que todo texto que está dentro da tag <script> deve ser lido como um Javascript, logo ele vai tentar interpretar a frase "Olá Mundo" como se fosse uma função Javascript válido. O que sabemos que não é, e então ele exibirá um erro de *Unexpected Identifier*, que nada mais é o que ele dizendo através do console de desenvolvedor que ele não entende "Olá mundo" como uma de suas funções.

Saber identificar e interpretar os erros do console é uma habilidade de todo bom desenvolvedor Javascript, então é bom começarmos a praticá-la desde já.

Se desejamos que uma mensagem seja exibida no console, devemos sempre chamar a função **console.log()**, que é a função que recebe como parâmetro o conteúdo da mensagem que queremos exibir no console.

**07Query Selector**

**Transcrição**

Agora que temos o domínio das funções iniciais do JavaScript, alert() e console.log(), usadas para exibir mensagens ao usuário, vamos começar a manipular os dados do site da Aparecida. A página deixará de ser estática e ganhará mais interatividade. O índice de massa corporal (IMC) será calculado automaticamente, e conseguiremos adicionar um novo paciente sem alterarmos o HTML. Isso dará dinamismo e deixará a página mais moderna.

Para começarmos a manipular a página, se quiséssemos alterar o título "Meus pacientes" para "Meus clientes", como isso poderia ser feito com JavaScript? Precisaríamos ter acesso ao código do arquivo HTML. Tudo o que estiver contido na tag **<script>** será interpretado como JS e, o que está fora, como HTML. Teremos que levar as funcionalidades criadas em HTML para o mundo JS.

Primeiramente, vamos conhecer o **DOM** (Document Object Model), representação do HTML para o nosso JavaScript, acessível por uma palava do JavaScript chamada document.

No console do navegador, quando digitamos document e pressionamos "ENTER" em seguida, veremos **tudo** o que está na página HTML. O document será a ponte entre o JavaScript e o HTML, e tudo que for alterado nele será alterado na exibição para o usuário.

Vamos experimentar adicionar o document na tag <script>, lembrando que não iremos usá-lo entre "" (aspas), pois ele funcionará como uma varíavel, e as aspas são usadas quando trabalhamos com uma *string*, e não é uma palavra ou frase que queremos imprimir.

<head>

<meta charset="UTF-8">

<title>Aparecida Nutrição</title>

<link rel="icon" href="favicon.ico" type="image/x-icon">

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

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

<script>

console.log("Oi Mundo");

console.log(document);

</script>

</head> COPIAR CÓDIGO

Se voltarmos ao navegador e atualizarmos a página, veremos que no JavaScript teremos acesso ao document. Conseguiremos ver o seguinte código no navegador:

#document

<!DOCTYPE html>

<html lang="pt-br">

<head>...</head>

<body>...</body>

</html>COPIAR CÓDIGO

Mas se não quisermos manipular o DOM inteiro, e sim apenas um pedaço, por exemplo, o texto dentro da tag <h1>, localizada acima do fechamento do <header>,

<header>

<div class="container">

<h1>Aparecida Nutrição</h1>

</div>

</header>

<main>

<section class="container">

<h2>Meus paciente</h2>

<table>

<thead>

//...COPIAR CÓDIGO

como faríamos para modificar apenas o texto "Aparecida Nutrição", que é um pedaço do document?

Vamos encontrar uma forma de **pesquisar** somente a tag <h1>. Para isto, usaremos o método querySelector(), passando como parâmetro o que queremos encontrar - neste caso, entre aspas, pois queremos o termo exato. No console, iremos digitar:

document.querySelector("h1");COPIAR CÓDIGO

Após executarmos o método, ele retornará o conteúdo da tag:

document.querySelector("h1");

<h1>Aparecida Nutrição</h1>COPIAR CÓDIGO

Assim, será selecionado o primeiro h1 da página, justamente aquele que queremos modificar. Então podemos passar este código para o navegador e imprimir o h1 no console do navegador para verificarmos se ele realmente foi selecionado, utilizando o console.log() novamente dentro da tag <script>:

<script>

console.log(document.querySelector("h1"));

</script>COPIAR CÓDIGO

Mas ao atualizarmos a página, o retorno será null. Por que isso acontece? Se usamos o querySelector() no console, conseguimos que o h1 seja retornado, porém, isto não ocorre no código. Qual é a diferença?

O browser, ao carregar a página HTML, vai lendo linha por linha, de cima para baixo. Quando ele chega na tag <script>, ele tenta buscar um h1 na página, porém, isto não está carregado em sua memória. A tag <h1> está **abaixo** do código JavaScript e ainda não foi interpretado pelo navegador, logo, ele não poderá ser selecionado.

Até agora estamos escrevendo HTML e JavaScript no mesmo arquivo, o que pode se tornar confuso conforme nosso código for crescendo. Para evitarmos isso, poderemos escrevê-los em arquivos separados.

O mesmo não ocorre quando executamos o código no console do navegador, pois a página já estará totalmente carregada, e o document estará completo.

Por conta de situações como esta, é uma boa prática colocar a tag <script> no fim do HTML, mais precisamente, como último elemento de <body> após o fechamento de <main>:

<!-- ... -->

</section>

</main>

<script>

console.log(document.querySelector("h1"));

</script>

</body>COPIAR CÓDIGO

Desta vez, ao recarregarmos a página, o **h1** é impresso no console do navegador, sendo selecionado corretamente.

Agora que conseguimos selecionar o h1, o que é preciso fazer para alterarmos o texto? Primeiramente, em vez de imprimir, salvaremos a parte selecionada, no caso o h1, dentro da variável titulo. Para isso, usaremos a palavra var:

<!-- ... -->

</section>

</main>

<script>

var titulo = document.querySelector("h1");

console.log(titulo);

</script>

</body>COPIAR CÓDIGO

No console, a tag <h1> continuará sendo exibida.

<h1>Aparecida Nutrição</h1>COPIAR CÓDIGO

Porém, nosso real objetivo é pegar o conteúdo textual Aparecida Nutrição.

Algumas tags, como h1, h2, p e span, possuem um **conteúdo de texto**. Nesses casos, o JavaScript possui uma propriedade que nos permite acessá-lo: textContent. Vamos testar e imprimir o conteúdo de texto da variável titulo, que representa o h1:

<!-- ... -->

<script>

var titulo = document.querySelector("h1");

console.log(titulo);

console.log(titulo.textContent);

</script>COPIAR CÓDIGO

De volta ao navegador, veremos uma diferença no que será impresso pelos dois console.log()s:

<h1>Aparecida Nutrição</h1>

Aparecida NutriçãoCOPIAR CÓDIGO

Somente o texto **Aparecida Nutrição** será impresso no segundo console.log(). Então conseguiremos acessar e exibir o conteúdo de texto da tag. E para alterá-lo, basta usar o textContent e passar um novo valor para o titulo, igualando-o a um novo texto:

<script>

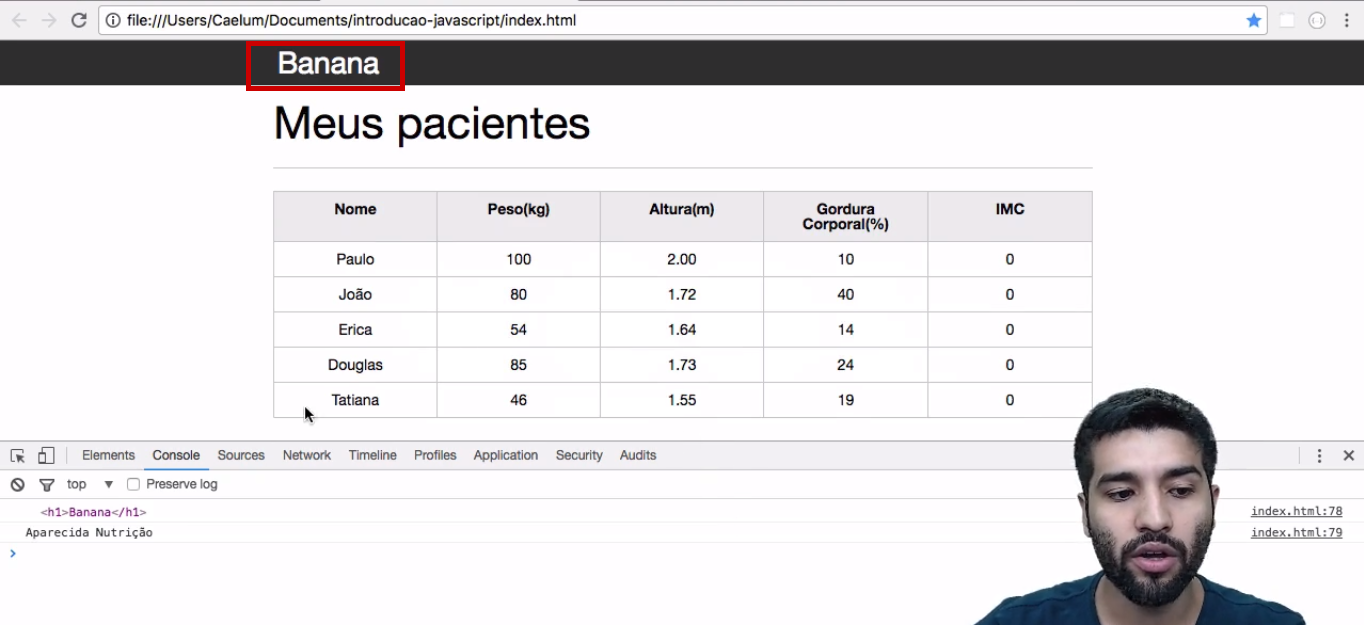
var titulo = document.querySelector("h1");

console.log(titulo);

console.log(titulo.textContent);

titulo.textContent = "Banana";

</script>COPIAR CÓDIGO



Observe que o título da página foi trocado. Se quisermos alterá-lo novamente, por exemplo, para "Aparecida Nutricionista", basta modificar a propriedade textContent.

<script>

var titulo = document.querySelector("h1");

titulo.textContent = "Aparecida Nutricionista";

</script>COPIAR CÓDIGO

Assim, nós alteramos o conteúdo de texto.

**08Boas Práticas**

**Transcrição**

Quando programamos com o JavaScript, é provável que estejamos trabalhando com outras pessoas, então, o que acontece se outro desenvolvedor decidir deixar de utilizar a tag <h1>, substituindo-a por <h2>?

<header>

<div class="container">

<h2>Aparecida Nutrição</h2>

</div>

</header>COPIAR CÓDIGO

Ao abrirmos a página no navegador, será exibida a seguinte mensagem no console:

Uncaught TypeError: Cannot set property 'textContent' of nullCOPIAR CÓDIGO

A mensagem indica que a propriedade textContent é nula e apontará a linha na qual ocorreu o erro, no caso, na tag <script>.

<!-- ... -->

</section>

</main>

<script>

var titulo = document.querySelector("h1");

titulo.textContent = "Aparecida Nutricionista";

</script>COPIAR CÓDIGO

O h1 não foi encontrado porque ele deixou de existir na nossa página e, sendo assim, impossibilitado de ser selecionado. O problema de se buscar uma tag no DOM é que o código JavaScript fica muito atrelado ao HTML. Caso o desenvolvedor decida fazer alterações no HTML, o código deixará de funcionar. Por isso, é uma boa prática não buscarmos por uma tag HTML específica, e usar outras opções que a função querySelector() nos disponibiliza.

Além das tags HTML, o querySelector() nos permite buscar por um elemento por meio da **classe**, do **id**, ou seja, dos **seletores CSS**. Vamos testar esse recurso adicionando uma classe dentro do <h1>:

<header>

<div class="container">

<h2 class="titulo">Aparecida Nutrição</h2>

</div>

</header>COPIAR CÓDIGO

No código JS inserido na parte de baixo do arquivo, em vez de realizarmos a busca no document.querySelector() pela tag <h1>, usaremos o seletor CSS que retornaria o conteúdo na mesma tag. Iremos utilizar o seletor CSS para classe: . (ponto).

<script>

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";

</script>COPIAR CÓDIGO

Agora, serão retornados os elementos da classe .titulo, e **Aparecida Nutricionista** reaparece na página. Se outro desenvolvedor modificar novamente a tag h1 no código HTML, nosso código não será prejudicado e o JavaScript continuará sendo executado, pois deixamos as responsabilidades desacopladas.

Trata-se de uma boa prática separarmos o código JavaScript do HTML, no entanto, o código ainda está completo no index.html, atuando independentemente. O trecho JS está salvo na tag <script>. É interessante desacoplarmos os códigos em arquivos diferentes, com diferentes extensões, .js, .html - já temos o código CSS separado em um arquivo .css.

A seguir, criaremos o arquivo principal.js na pasta **/js**, com todo o código JavaScript. O próximo passo será mover o código da tag <script> para dentro desse arquivo:

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";COPIAR CÓDIGO

A tag **<script>** continuará na página index.html, porém, ela não ficará vazia, e apontará para o arquivo JavaScript externo que criamos, por meio do atributo src (referente ao termo *source*). Como o arquivo principal.js está na pasta js, especificaremos o caminho completo no atributo:

<!-- ... -->

</section>

</main>

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

</body>COPIAR CÓDIGO

Quando recarregarmos a página, o arquivo continuará funcionando corretamente. Vale reforçar que é recomendado deixarmos o código HTML separado do JavaScript.

Nesta aula, nós abordamos vários temas sobre a linguagem JavaScript, vimos que ela ganhou relevância por estar sendo usada de diferentes formas, no navegador, banco de dados, placas Arduíno. É importante que desenvolvedores tenham, pelo menos, um conhecimento básico de como funciona a linguagem.

Vimos algumas funções como console() e alert(), que nos permitem exibir mensagens ao usuário, e conhecemos a utilidade do console de desenvolvedor, presente nos navegadores. Podemos executar códigos JavaScript no console, com isso, conseguimos testar algumas coisas diretamente no navegador.

Vimos também como realizar a busca por algum elemento do HTML, no código JavaScript. Para isto, utilizamos a variável document, que contém todo o conteúdo HTML da página. Quando manipulamos o document, conseguimos manipular o que será exibido ao usuário. Essa manipulação foi realizada por meio da seleção de trechos da tela, feita com o querySelector(). Vimos que este método busca por nome de tags - o que pode trazer problemas -, assim como seletores de CSS.

Na aula, mostramos como criar uma variável no JavaScript e alterar um conteúdo de texto utilizando a propriedade textContent.

Aguardo você nas aulas seguintes, espero que tenha gostado da introdução do curso.

**09Estrutura do curso**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24390/next)

Além dos **vídeos com a transcrição**, neste curso temos alguns tipos de exercícios para você consolidar seus conhecimentos:

Os exercícios de **múltipla escolha** são para você rever os conceitos dos vídeos, para você relembrar os detalhes de cada função ou conceito e fixar bem os conhecimentos.

Os exercícios **Mãos na Massa** são para você implementar o que fizemos em vídeo, com um texto te guiando para que você não esqueça nenhum detalhe.

E no **primeiro exercício** Mão na Massa de cada capítulo também temos o link para **Download** do **projeto completo até aquele ponto**, para caso você precise pular alguma coisa ou comparar com o código que você mesmo fez!

**10Console do desenvolvedor**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23588/next)

Marque as alternativas abaixo que contêm características ou funcionalidades do **console de desenvolvedor** do navegador:

Parte superior do formulário

* Alternativa correta



Consegue alterar nossos arquivos **.js** com ele.

* Alternativa correta



Executar funções Javascript diretamente no console.

Correta!

* Alternativa correta



Exibir erros de códigos Javascript carregados pelo navegador.

Correta!

* Alternativa correta



Pode ser aberto com a tecla de atalho **CTRL + SHIFT + J** no Windows/Linux e **CMD + ALT + J** no Mac.

Correta!

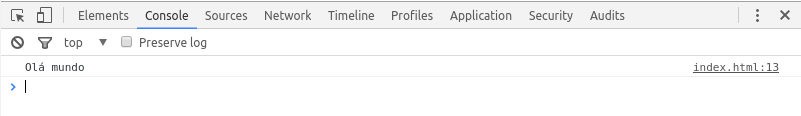
* Alternativa correta



Ver os resultados de funções que imprimem no console, como a **console.log()**.

Correta!

Parte inferior do formulário



O console do desenvolvedor é uma ferramenta muito poderosa e versátil para o desenvolvedor Javascript. Com ele conseguimos rapidamente ser notificados de **possíveis erros em nosso código**, podemos **executar funções**, observar a **saída** de algumas funções e **executar pequenos testes** das funcionalidades do Javascript de modo rápido.

É uma boa prática sempre que estivermos testando nosso site ficar com o console aberto, para ficar monitorando possíveis erros em nosso código.

Como utilizar o console é algo bastante corriqueiro, podemos nos aproveitar dos atalhos **CTRL + SHIFT + J** (Win/Linux) ou **CMD + ALT + J** (Google Chrome no Mac. No Safari, o atalho é **CONTROL + ALT + C**, com o menu "Develop" ativado) para ter acesso ao console de modo rápido.

**11Importando um arquivo externo**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23591/next)

Qual código abaixo importa corretamente um arquivo .js chamado contador.js que se encontra na mesma pasta de nosso arquivo HTML?

Parte superior do formulário

* Alternativa correta



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

* Alternativa correta



<script font="contador.js"></script>

* Alternativa correta



<script src="contador.js"/>

* Alternativa correta



<javascript src="contador.js"></javascript>

* Alternativa correta



<script href="contador.js"></script>

Parte inferior do formulário

Sabemos que é importante fazer a separação de mundos HTML e Javascript, por isso criamos arquivos externos para guardar nosso código. Só que ao separar o Javascript em um arquivo externo, precisamos lembrar de importa-lo no HTML e para isso precisamos utilizar a tag <script> assim:

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

**12Mensagem de boas vindas**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23602/next)

Qual opção abaixo substitui o hasheado (XXXX) por um código que exibe uma pop up com a mensagem de boas-vindas?

XXXX mensagem = "Seja bem vindo ao meu site!"

XXXX(mensagem);COPIAR CÓDIGO

Parte superior do formulário

* Alternativa correta



int e console.log respectivamente.

* Alternativa correta



var e alert respectivamente.

* Alternativa correta



var e console.log respectivamente.

* Alternativa correta



string e alert respectivamente.

* Alternativa correta



var e show respectivamente.

Parte inferior do formulário

Queremos declarar a variável mensagem, por isso devemos utilizar a palavra **var** para criar esta entidade:

var mensagem = "Seja bem vindo ao meu site!"COPIAR CÓDIGO

e logo em seguida desejamos exibir uma pop-up, logo é necessário utilizar a função **alert** que é responsável por isso:

alert(mensagem);COPIAR CÓDIGO

**13Um tal de document**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23605/next)

O que contém a variável document do JavaScript ?

Parte superior do formulário

* Alternativa correta



O código HTML que está escrito no nosso arquivo.html. Alterando os valores do document, estamos alterando o HTML, e por consequência o nosso código fonte do arquivo HTML.

* Alternativa correta



Ela é a representação do DOM ou **D**ocument **O**bject **M**odel, que é a representação da nossa página HTML para o navegador, que é utilizada pelo Javascript para manipular a página.

* Alternativa correta



A lista de todos os estilos dos elementos da página, que nos permite ter acesso aos dados de estilo de cada uma das tags do HTML.

Parte inferior do formulário

O **document** é uma variável muito importante do Javascript. Ela contém o DOM ou **D**ocument **O**bject **M**odel, que é como o navegador enxerga o HTML utilizado por ele para renderizar a página.

O navegador, ao ler o seu arquivo HTML, cria uma *cópia* em memória daquele HTML e a partir dessa *cópia* ele vai desenhando a sua página, colocando as tags e aplicando os estilos. Esta *cópia* é o que chamamos de **DOM** uma representação em memória do HTML do seu arquivo, que o navegador usa para desenhar a página, e a variável **document** é quem contêm o DOM.

Por isso, quando dizemos que com o Javascript nós estamos **manipulando** o DOM, estamos manipulando o que o navegador renderizou. Então ao trocar algum texto do DOM, o navegador *imediatamente* desenha novamente aquele texto, pois o DOM é o que o navegador usa para desenhar o seu site.

Outra caracteristica interessante, é que como **manipulamos o DOM** , quando trocamos um texto de um <h1> ou um estilo de um elemento, na verdade estamos alterando a representação **em memória** , o que faz com que o arquivo fonte que contêm seu HTML fique intacto!

**14Apenas um pedacinho...**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23607/next)

Sabemos que o **document** contém a representação de todo o HTML da nossa página, porém na maioria dos casos estamos interessados em apenas uma parte do HTML, e não nele todo.

Qual **função** podemos utilizar na variável **document** para especificar que queremos trazer apenas uma parte do HTML ?

Parte superior do formulário

* Alternativa correta



document.querySelector();

Correto!

* Alternativa correta



document.getElement();

* Alternativa correta



document.query()

* Alternativa correta



document.queryselector();

* Alternativa correta



document.searchSelector();

Parte inferior do formulário

Quando desejamos pegar apenas parte do HTML, devemos utilizar a função **querySelector()**. A função querySelector faz a busca através de seletores CSS para trazer o HTML que você está interessado. Por exemplo, no caso abaixo:

<h1 class="titulo"> Meu primeiro site</h1>COPIAR CÓDIGO

Podemos buscar este <h1> passando o seletor CSS correto para a função querySelector. Utilizando o seletor de **tag**, podemos buscar assim:

document.querySelector("h1");COPIAR CÓDIGO

E utilizando um seletor CSS de classe, ficaria assim:

document.querySelector(".titulo");COPIAR CÓDIGO

Neste caso, ambas as funções retornariam o h1. Ter domínio dos seletores CSS é importante se você deseja trabalhar bem com o Javascript!

**15Mãos na massa: Download Inicial e Setup**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24388/next)

**Download Inicial e Setup**

**Baixando o projeto**

Para dar ínicio ao projeto, faça o [download](https://s3.amazonaws.com/caelum-online-public/introducao-javascript/stages/introducao-javascript-inicial.zip) do website da Aparecida Nutrição.

**O editor de textos ideal**

Com o projeto dezipado, precisaremos de um editor de textos confiável para trabalhar em nosso projeto. Eu recomendo o [Atom](https://atom.io/), por ser open-source, possuir um sistema diverso de plugins e **suporte** para todas as plataformas de desenvolvimento (Windows, Linux e MacOS).

Outras conhecidas são o [Sublime Text](https://www.sublimetext.com/) ou o [Visual Studio Code](https://code.visualstudio.com/), que também são bons e oferecem suporte a todas as plataformas.

Só peço que evite editores como o Notepad padrão do Windows, Gedit ou o Notepad++ , pois esses não têm as facilidades como autocomplete, visualização de pastas e highlight específicos para as tecnologias da Web.

Recomendo também abrir a pasta do projeto pelo editor de textos, para que seus arquivos fiquem de fácil acesso ao lado esquerdo da tela. Em geral conseguimos isto indo no menu *File -> Open Folder -> selecione a pasta baixada*.

**A plataforma do Javascript: O navegador!**

Depois de aberto o projeto no seu editor de textos, precisamos escolher a plataforma que rodará nosso Javascript. Vimos que o Javascript é aceito em diversos locais hoje em dia, porém como o nosso foco é desenvolver um site para a Web, vamos utilizar o ambiente inicial do Javascript, o navegador!

Como navegador eu recomendo **fortemente** que você utilize o [Google Chrome](https://www.google.com.br/chrome/browser/desktop/) pois é o que eu utilizei ao longo dos vídeos e assim fica mais fácil de você acompanhar e comparar.

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

O setup inicial e a escolha das ferramentas corretas é um passo importante do início de qualquer projeto!

**16Mãos na massa: O primeiro Javascript**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24389/next)

**O primeiro arquivo .js**

Para dar início ao nosso projeto, vamos começar criando nosso primeiro arquivo Javascript, o principal.js, que vai conter o código inicial que faremos neste primeiro capítulo. Para **manter uma boa organização de código** crie o arquivo principal.js dentro da pasta **/js**. Sua estrutra de pastas deve ficar assim:

introducao-javascript

├── css

│ ├── index.css

│ └── reset.css

├── js

│ └── principal.js

├── img

├── favicon.ico

└── index.htmlCOPIAR CÓDIGO

**Importando o arquivo**

Claro que após criar um novo arquivo Javascript, não pode esquecer de importá-lo ao final da tag <body> em nosso HTML, utilizando a tag <script>:

// Restante do código HTML

</section>

</main>

<!-- Importação dos Javascripts AQUI -->

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

</body>

</html>COPIAR CÓDIGO

Repare que colocamos no atributo src o caminho relativo para nosso arquivo principal.js, ou seja a partir da pasta atual, vamos descer para a pasta js/ e em seguida linkar com arquivo.

**Trocando o título da página**

Para dar início aos nossos conhecimentos com o Javascript, vamos trocar o título da nossa página de Aparecida Nutrição para Aparecida Nutricionista, para praticar. Em nosso principal.js vamos:

1- O primeiro passo é selecionar o elemento da página que queremos alterar, neste caso o :

<h1>Aparecida Nutrição</h1>COPIAR CÓDIGO

Sabemos que é possível selecionar um elemento através de sua tag, e como só temos este h1 na página poderíamos facilmente utilizá-lo, porém também vimos que não é uma boa prática utilizar um seletor de tag quando estamos selecionando um elemento do HTML, pois este h1 poderia facilmente ser trocado por um h2 e nosso código quebraria.

Vamos adicionar uma classe no h1 para que a seleção do elemento sempre seja feita atráves desta classe:

<h1 class="titulo">Aparecida Nutrição</h1>COPIAR CÓDIGO

2- Agora que o h1 já tem uma classe, podemos utilizar a **função seletora** do Javascript que é responsável de trazer um elemento do mundo do HTML para o mundo do Javascript que é a querySelector(). Sabemos que queremos buscar na nossa página pelo h1, então vamos chamar a querySelector() na variável que representa a nossa página para o Javascript , o document:

document.querySelector(".titulo");COPIAR CÓDIGO

Como queremos manipular este elemento selecionado, vamos colocá-lo em uma variável de nome semântico para poder ter acesso a ele depois:

var titulo = document.querySelector(".titulo");COPIAR CÓDIGO

3- Por último, como queremos trocar o texto do h1, vamos acessar a propriedade textContent do elemento selecionado, que é a propriedade que tem o contéudo de texto daquela tag. Como queremos **alterar** o contéudo de texto daquela tag, vamos **trocar** o valor da propriedade textContent para um novo texto, que é o nosso novo título Aparecida Nutricionista:

titulo.textContent = "Aparecida Nutricionista"COPIAR CÓDIGO

Ao recarregar a página você deve ver que assim que a página é carregada o novo título já aparece no topo!

**16Mãos na massa: O primeiro Javascript**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24389/next)

**O primeiro arquivo .js**

Para dar início ao nosso projeto, vamos começar criando nosso primeiro arquivo Javascript, o principal.js, que vai conter o código inicial que faremos neste primeiro capítulo. Para **manter uma boa organização de código** crie o arquivo principal.js dentro da pasta **/js**. Sua estrutra de pastas deve ficar assim:

introducao-javascript

├── css

│ ├── index.css

│ └── reset.css

├── js

│ └── principal.js

├── img

├── favicon.ico

└── index.htmlCOPIAR CÓDIGO

**Importando o arquivo**

Claro que após criar um novo arquivo Javascript, não pode esquecer de importá-lo ao final da tag <body> em nosso HTML, utilizando a tag <script>:

// Restante do código HTML

</section>

</main>

<!-- Importação dos Javascripts AQUI -->

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

</body>

</html>COPIAR CÓDIGO

Repare que colocamos no atributo src o caminho relativo para nosso arquivo principal.js, ou seja a partir da pasta atual, vamos descer para a pasta js/ e em seguida linkar com arquivo.

**Trocando o título da página**

Para dar início aos nossos conhecimentos com o Javascript, vamos trocar o título da nossa página de Aparecida Nutrição para Aparecida Nutricionista, para praticar. Em nosso principal.js vamos:

1- O primeiro passo é selecionar o elemento da página que queremos alterar, neste caso o :

<h1>Aparecida Nutrição</h1>COPIAR CÓDIGO

Sabemos que é possível selecionar um elemento através de sua tag, e como só temos este h1 na página poderíamos facilmente utilizá-lo, porém também vimos que não é uma boa prática utilizar um seletor de tag quando estamos selecionando um elemento do HTML, pois este h1 poderia facilmente ser trocado por um h2 e nosso código quebraria.

Vamos adicionar uma classe no h1 para que a seleção do elemento sempre seja feita atráves desta classe:

<h1 class="titulo">Aparecida Nutrição</h1>COPIAR CÓDIGO

2- Agora que o h1 já tem uma classe, podemos utilizar a **função seletora** do Javascript que é responsável de trazer um elemento do mundo do HTML para o mundo do Javascript que é a querySelector(). Sabemos que queremos buscar na nossa página pelo h1, então vamos chamar a querySelector() na variável que representa a nossa página para o Javascript , o document:

document.querySelector(".titulo");COPIAR CÓDIGO

Como queremos manipular este elemento selecionado, vamos colocá-lo em uma variável de nome semântico para poder ter acesso a ele depois:

var titulo = document.querySelector(".titulo");COPIAR CÓDIGO

3- Por último, como queremos trocar o texto do h1, vamos acessar a propriedade textContent do elemento selecionado, que é a propriedade que tem o contéudo de texto daquela tag. Como queremos **alterar** o contéudo de texto daquela tag, vamos **trocar** o valor da propriedade textContent para um novo texto, que é o nosso novo título Aparecida Nutricionista:

titulo.textContent = "Aparecida Nutricionista"COPIAR CÓDIGO

Ao recarregar a página você deve ver que assim que a página é carregada o novo título já aparece no topo!

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

Apesar deste capítulo inicial ter sido curto, aprendemos pontos importantes como:

* Separar a organização de nosso código em arquivos .js
* Importar arquivos .js sempre ao final do body
* A representação do HTML pela variável document
* A função que faz busca querySelector()
* Preferir selecionar por #id ou .classe

No próximo avançaremos mais em nossos conhecimentos de Javascript.

**01Buscando dados do paciente**

**Transcrição**

Vamos continuar com o projeto do site da Aparecida! Nós vimos como selecionar um elemento da página com a função querySelector(), assim como é possível alterar um texto com o textContent. Se selecionarmos um elemento da página, obter e reescrevermos seu conteúdo textual, poderemos realizar o cálculo automático do IMC de cada paciente cadastrado e facilitar a vida da Aparecida.

Para calcularmos o IMC, trabalharemos com valores do **peso** e da **altura** de um paciente da tabela, pois o cálculo é feito desta forma:

IMC = peso / altura x altura;COPIAR CÓDIGO

Tendo os dados, precisaremos de uma forma para levá-los ao mundo do JavaScript, calcular o IMC e imprimir o resultado na devida coluna. Esta será nossa missão, vamos começar?

##Extraindo o peso e a altura do paciente

Primeiramente, vamos focar no primeiro paciente, o Paulo. No index.html, observaremos os seguintes dados HTML:

<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-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

Veremos que cada paciente está contido em uma linha da tabela (<tr>) e que em cada coluna (<td>) há um dado. Já sabemos como selecionar um paciente, por meio do querySelector():

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";

var paciente = document.querySelector(".paciente");COPIAR CÓDIGO

Mas todos os pacientes possuem a classe paciente. Se queremos buscar o Paulo, devemos atribuir algo mais específico, para isto, iremos definir um id para o primeiro paciente (primeiro-paciente):

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

De volta ao arquivo JavaScript, em querySelector utilizaremos o seletor de id, a cerquilha (#, também conhecida como "tralha" ou "hashtag"):

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";

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

console.log(paciente)COPIAR CÓDIGO

Vamos executar o console.log() da variável paciente, e esperamos conseguir executar a tag <tr>, que contém o Paulo. Faremos uma segunda busca no navegador, porém, desta vez vamos buscar em paciente, em vez de pesquisarmos no documento inteiro. O retorno será o seguinte:

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

É retornado um <tr>, no entanto, nosso interesse é no peso e na altura do Paulo, que estão dentro de tags <td>. Mais especificamente, estamos interessados no conteúdo de texto da <td>, e se observarmos o código, veremos que elas podem realizar a busca utilizando as classes info-peso e info-altura.

Nós já usamos o querySelector() para buscar dentro do document, será que conseguimos realizá-la também dentro de paciente?

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";

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

var tdPeso = paciente.querySelector(".info-peso");

console.log(paciente); // tr

console.log(tdPeso); // td que tem o pesoCOPIAR CÓDIGO

Aqui, buscamos em paciente um elemento que tenha a classe .info-peso, usando o seletor de classe (.). Isso vai nos retornar a **<td>** que contém este dado do paciente. Observe que este dado é salvo na variável tdPeso.

No console, veremos o seguinte resultado:

< tr class="paciente" id="primeiro-paciente">...</tr>

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

Novamente, exibe-se a linha com o <td>, e só queremos o conteúdo de texto, ou seja, o valor referente ao peso do paciente. Para termos acesso ao seu conteúdo de texto, adicionaremos o textContent no principal.js:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

console.log(paciente); // tr

console.log(tdPeso); // td que tem o peso

console.log(peso); //Obter 100COPIAR CÓDIGO

No console do navegador, será exibido o valor 100.

< tr class="paciente" id="primeiro-paciente">...</tr>

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

100COPIAR CÓDIGO

No entanto, para o cálculo do IMC, além do peso, precisaremos também da altura. Então vamos acessá-la utilizando a mesma lógica e criando a variável tdAltura logo abaixo da variável peso:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;COPIAR CÓDIGO

Fizemos uma busca com o querySelector() na variável paciente. Nosso objetivo é encontrar a td com a classe .info-altura. Também criamos a variável altura, que é o conteúdo de texto do td, e por isso utilizamos o textContent.

Para inspecionarmos se o que foi feito deu certo, adicionaremos o console.log() para imprimir o tdAltura após as variáveis recém adicionadas:

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

console.log(paciente); // tr

console.log(tdAltura);

console.log(altura); COPIAR CÓDIGO

No console, obteremos:

<tr class="paciente" id="primeiro-paciente">...</tr>

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

2.00COPIAR CÓDIGO

Veremos o valor 2.00 impresso no console, e obteremos os valores do peso e da altura, então seguiremos para o cálculo do IMC, a partir da seguinte fórmula:

IMC = peso / altura x alturaCOPIAR CÓDIGO

Isto é, o valor da massa (peso) dividido pela altura multiplicada por ela mesma.

Se a replicarmos no JavaScript utilizando os operadores matemáticos da linguagem de programação, faremos o seguinte cálculo em uma nova variável, chamada de imc:

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

var imc = peso / (altura \* altura);COPIAR CÓDIGO

Removeremos o console.log(), que se tornou desnecessário no código - observe que usamos o operador \* (asterisco) na multiplicação.

No caso do paciente Paulo, o resultado do IMC deveria ser:

IMC = 100 / (2.00 x 2.00)

IMC = 100 / 4.00

IMC = 25COPIAR CÓDIGO

Anotaremos a lógica do cálculo com comentário.

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

var imc = peso / altura \* altura; // 100 / 2.0 x 2.0 = 100 / 4 =>>>>>>> 25

console.log(imc);COPIAR CÓDIGO

O resultado esperado para a variável **imc** é **25**, porém, o console exibirá 100 como resultado.

Como isso é possível? Será que JavaScript não sabe fazer esta multiplicação básica?

Na verdade, a multiplicação está sendo feita, no entanto precisaremos identificar o problema no código!

**02Acessando elementos dentro de outro elemento**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23601/next)

Temos o seguinte fragmento HTML:

<table>

<thead>

<tr>

<th>Nome</th>

<th>Peso(kg)</th>

<th>Altura(m)</th>

<th>Gordura Corporal(%)</th>

<th>IMC</th>

</tr>

</thead>

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>

</tbody>

</table>COPIAR CÓDIGO

Em seguida, temos o código que busca a tr que guarda os dados de um paciente. Usaremos um seletor de ID para a função document.querySelector:

var trPaciente = document.querySelector('#primeiro-paciente');COPIAR CÓDIGO

No entanto, queremos extrair o peso e a altura que representamos pelos elementos do tipo td dentro da tr que acessamos no mundo JavaScript através da variável trPaciente.

Qual das opções abaixo acessa corretamente a partir de trPaciente o peso e a altura?

Parte superior do formulário

* Alternativa correta



var tdPeso = trPaciente.querySelector('.info-peso');

var tdAltura = trPaciente.querySelector('.info-altura');

Correta! Selecionamos corretamente com o seletor de classe o ponto .

* Alternativa correta



var tdPeso = trPaciente.querySelector('info-peso');

var tdAltura = trPaciente.querySelector('info-altura');

* Alternativa correta



var tdPeso = trPaciente.querySelector('#info-peso');

var tdAltura = trPaciente.querySelector('#info-altura');

Parte inferior do formulário

Com JavaScript podemos obter um elemento pai, no caso trPaciente e dele realizar novas consultas para obter elementos filhos, como foi o caso das TD's de peso e altura.

**03O prato do dia**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24289/next)

Mariana é dona de um restaurante que troca os pratos semanalmente. Os pratos são exibidos para seus clientes em seu site como uma tabela, como a abaixo:

<table>

<thead>

<tr>

<th>Salada</th>

<th>Prato Principal</th>

<th>Acompanhamento</th>

</tr>

</thead>

<tbody>

<tr class="prato-do-dia" id="sexta">

<td class='salada'>Alface e Tomate</td>

<td class='principal'>Frango Grelhado</td>

<td class='acompanhamento'>Arroz e Feijão preto</td>

</tr>

...Outros dias da semana

<tr class="prato-do-dia" id="segunda">

<td class='salada'>Salada de Batata</td>

<td class='principal'>Macarrão com molho branco</td>

<td class='acompanhamento'>Milho, ervilha e brócolis</td>

</tr>

</tbody>

</table>COPIAR CÓDIGO

Se Mariana deseja exibir o prato de cada dia em uma outra área de sua página, da forma abaixo:

Prato do dia!

Alface e Tomate

Frango Grelhado

Arroz e Feijão PretoCOPIAR CÓDIGO

Qual código abaixo extrai as informações de **um dia da tabela**, por exemplo a sexta-feira, e imprime **no console** no formato que Mariana deseja?

Parte superior do formulário

* Alternativa correta



var salada = document.querySelector(".salada").textContent;

var principal = document.querySelector(".principal").textContent;

var acompanhamento = document.querySelector(".acompanhamento").textContent;

console.log("Prato do dia!");

console.log(salada);

console.log(principal);

console.log(acompanhamento);

* Alternativa correta



var pratoDoDia = document.querySelector(".prato-do-dia");

var salada = pratoDoDia.querySelector(".salada").textContent;

var principal = pratoDoDia.querySelector(".principal").textContent;

var acompanhamento = pratoDoDia.querySelector(".acompanhamento").textContent;

console.log("Prato do dia!");

console.log(salada);

console.log(principal);

console.log(acompanhamento);

* Alternativa correta



var pratoDoDia = document.querySelector("#sexta");

var salada = pratoDoDia.querySelector(".salada").textContent;

var principal = pratoDoDia.querySelector(".principal").textContent;

var acompanhamento = pratoDoDia.querySelector(".acompanhamento").textContent;

console.log("Prato do dia!");

console.log(salada);

console.log(principal);

console.log(acompanhamento);

A melhor estratégia neste caso, como queremos buscar os pratos de um dia específico é primeiro buscar a <tr> que contêm os pratos deste dia, fazendo o uso da função document.querySelector:

var pratoDoDia = document.querySelector("#sexta");COPIAR CÓDIGO

Parte inferior do formulário

Em seguida, vamos buscar **dentro desta <tr>** por os pratos específicos, então vamos utilizar a função querySelector **na própria <tr> \**, afinal a função querySelector() é especialista \**em buscar**, seja na página inteira quando utilizamos-a buscando no document (document.querySelector) ou quando queremos fazer uma busca em elemento específico que foi pré selecionado , como é o caso agora:

var salada = pratoDoDia.querySelector(".salada").textContent;

var principal = pratoDoDia.querySelector(".principal").textContent;

var acompanhamento = pratoDoDia.querySelector(".acompanhamento").textContent;COPIAR CÓDIGO

Já aproveitamos e emendamos na função seletora a propriedade textContent , afinal estamos interessados é no nome do prato que está escrito dentro da tag html, e não na tag em si!

Em seguida, basta apenas imprimirmos como foi solicitado:

console.log("Prato do dia!");

console.log(salada);

console.log(principal);

console.log(acompanhamento);

**04Calculando e exibindo o IMC corretamente**

**Transcrição**

O JavaScript sabe como multiplicar, mas não entendeu a mensagem que tentamos passar para ele! No código, o cálculo do IMC foi representado da seguinte forma:

var imc = peso / altura \* altura;COPIAR CÓDIGO

Nós conseguimos entender que a altura deve ser multiplicada por ela mesma antes da divisão. Porém, esquecemos de especificar para o JavaScript o que deve ser feito, logo, o código será executado da esquerda para direita, de forma que o primeiro cálculo seja a divisão do peso pela altura.

Peso / altura

100 / 2 = 50COPIAR CÓDIGO

O passo seguinte, foi multiplicar o resultado da divisão e multiplicá-lo pela altura.

50 x 2 = 100COPIAR CÓDIGO

Desta forma, chegamos ao resultado inesperado 100, e este não é o valor do IMC.

Em JavaScript, existem os operadores aritméticos básicos de todas as linguagens de programação: +, -, \*, / e %. Temos ainda outros caracteres para nos ajudar a representar uma conta matemática, os parênteses ().

Para corrigirmos o cálculo, adicionaremos os parênteses na conta, para indicar em qual ordem as operações deverão ser executadas. De volta ao arquivo principal.js:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

console.log(imc);COPIAR CÓDIGO

O resultado será exibido apenas no console e, no caso, o valor do IMC será 25.

Falta atualizarmos a tabela após o cálculo do IMC. Então selecionaremos a <td> do paciente, cujo valor é 0:

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

Queremos que este valor seja alterado para o resultado do cálculo:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

// selecionando a coluna do IMC

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

var imc = peso / (altura \* altura);COPIAR CÓDIGO

Nós já vimos como selecionar o tdAltura, para acessarmos o IMC, e também como usarmos o querySelector(). Então basta atribuirmos o conteúdo do IMC calculado ao tdImc, adicionando textContent:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

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

tdImc.textContent = imc;COPIAR CÓDIGO

Se tudo der certo, quando executarmos o código no navegador, a página será carregada com o primeiro valor do IMC preenchido:



O IMC do Paulo é exibido na sua coluna graças ao script JavaScript.

**05Desmembrando uma instrução em variáveis**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23598/next)

Aprendemos que o cálculo do índice de massa corporal (IMC) é feito dividindo o nosso peso pela altura vezes a altura, o que pode ser representado pela seguinte fórmula na linguagem JavaScript:

alert(78 / (1.71 \* 1.71));COPIAR CÓDIGO

No entanto, para melhorarmos a legibilidade e manutenção deste código, podemos utilizar variáveis.

Quais opções abaixo possuem o mesmo resultado da instrução anterior com o auxílio de variáveis?

Parte superior do formulário

* Alternativa correta



var peso = 78;

var altura = 1.71;

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

alert(imc);

* Alternativa correta



var peso = 1.71;

var altura = 78;

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

alert(imc);

* Alternativa correta



var peso = 78;

var altura = 1.71;

var imc = peso / altura \* altura;

alert(imc);

Faltou o parênteses em altura \* altura. Sem ele, o peso será dividido pela altura primeiro e o resultado será multiplicado pela altura, resultando em um valor totalmente não esperado.

Parte inferior do formulário

Veja que uma simples instrução quando desmembrada em variáveis necessita da atenção do desenvolvedor.

**07Operadores Lógicos**

**Transcrição**

Conseguimos calcular o IMC do paciente Paulo, mas o que acontece se alguém, no momento de digitar o código HTML, colocar um número negativo no campo de peso? Ou se incluir um número absurdo, como 10 mil quilos? Ou, sem querer, adicionar uma altura de 200?

IMC = 10000 / 2.00 x 2.00

IMC = 10000 / 4.00

IMC = 2500COPIAR CÓDIGO

O resultado será um valor impossível. Às vezes, na pressa de digitar, a pessoa pode acabar colocando um número a mais ou um sinal a menos, e dentre tantas outras informações, acaba não notando esses erros no HTML.

Valores improváveis não deveriam ser aceitos pela tabela. Como já possuímos os dados do peso e da altura no código JavaScript, é possível **validar** as informações. Desta forma, o IMC do paciente só será calculado se o peso e a altura forem válidos, e sempre teremos um IMC que reflita um valor real. Se o peso for muito alto ou a altura muito baixa, por exemplo, será exibida uma mensagem de aviso informando que os dados do paciente são inválidos e devem ser verificados.

Atualmente, o principal.js está assim:

var titulo = document.querySelector(".titulo");

titulo.textContent = "Aparecida Nutricionista";

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

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

tdImc.textContent = imc;COPIAR CÓDIGO

Antes de calcularmos o IMC, poderemos validar os dados, começando pelo peso. Com um if, verificaremos se o peso é menor ou igual a 0, ou seja, **negativo**. Se este for o caso, uma mensagem de alerta será exibida no console. Iremos adicionar o if logo abaixo da variável tdImc:

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

if (peso <= 0) {

console.log("Peso inválido");

}COPIAR CÓDIGO

Se colocarmos um valor inválido para o peso no index.html...

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

Além disso, 1000 quilos também é um valor muito alto para um ser humano. Então, vamos verificá-lo também, adicionando outro if:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

if (peso <= 0) {

console.log("Peso inválido");

}

if (peso > 1000) {

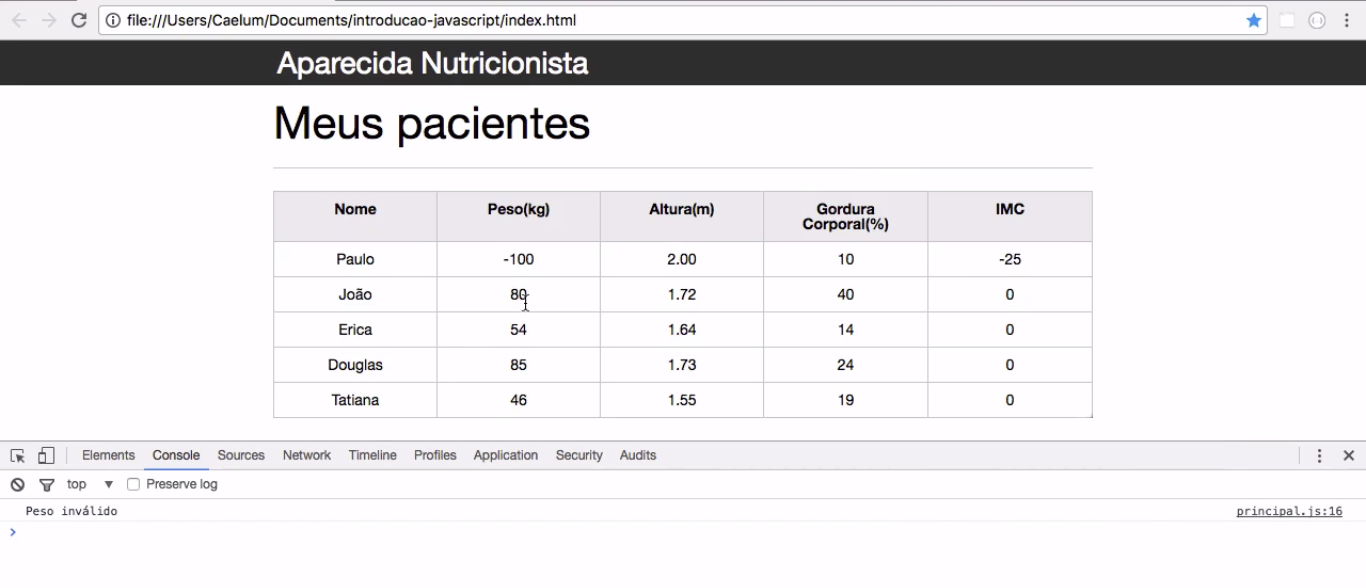
console.log("Peso inválido");

}

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

tdImc.textContent = imc;COPIAR CÓDIGO

No console, será exibida a mensagem Peso inválido.



No entanto, imagine o que aconteceria se para cada validação de peso, fosse necessário adicionar mais um **if**? O código acabaria ficando imenso! Porém, é possível juntar os dois tipos de verificações em um único **if**, sendo que cada um deles retornará true ou false para as verificações, ou seja, um resultado **booleano**. Tentaremos concatenar as verificações:

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

if (peso < 0 peso > 1000) {

console.log("Peso inválido");

}COPIAR CÓDIGO

Porém, como faremos para juntar as duas condições, sendo que o resultado deverá se enquadrar em uma ou outra condição? A palavra "ou" não existe no JavaScript, mas existe um operador lógico nas linguagens de programação que poderá ser utilizado: ||, duas barras verticais que recebem o nome de *pipes*:

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

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

console.log("Peso inválido");

}COPIAR CÓDIGO

Se a primeira condição, peso < 0, for verdadeira, ou a segunda condição, peso > 1000, for verdadeira, o código no **if** será executado. Ao testarmos, veremos que o funcionamento é o mesmo, sem a necessidade de criarmos um if para cada caso.

Seria interessante também verificarmos o valor de altura, considerando inválida aquela que for menor que 0 ou maior que 3:

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

console.log("Altura inválida");

}COPIAR CÓDIGO

Se testarmos rodar o código com um valor fora de uma das condições, a mensagem Altura inválida será exibida no console. Mas o cálculo do IMC ainda será realizado, mesmo nos casos em que trabalharmos com alturas e/ou pesos inválidos, o que não é o que queremos. Neste casos, gostaríamos que o IMC não fosse calculado.

##Cálculo do IMC somente com peso e altura válidos

Para que o IMC não seja calculado quando o peso e a altura forem inválidos, poderemos criar duas variáveis, uma validação para cada caso. Vamos inicializá-las com true:

var pesoEhValido = true;

var alturaEhValida = true;

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

console.log("Peso inválido");

}

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

console.log("Altura inválida!");

}COPIAR CÓDIGO

Poderemos testar inserindo um valor de altura inválido. No console, veremos a mensagem "Altura inválida!". Ou seja, estamos presumindo que ambos serão válidos. Mas caso não sejam, seus valores deverão mudar para false dentro dos ifs:

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

console.log("Peso inválido!");

pesoEhValido = false;

}

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

console.log("Altura inválida!");

alturaEhValida = false;

}COPIAR CÓDIGO

Os valores das variáveis começarão como verdadeiras, mas se algo der errado, serão setadas como false. Após as alterações, o código ficará assim:

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

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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

var pesoEhValido = true;

var alturaEhValida = true;

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

console.log("Peso inválido!");

pesoEhValido = false;

}

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

console.log("Altura inválida!");

alturaEhValida = false;

}

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

tdImc.textContent = imc;COPIAR CÓDIGO

Agora, antes de calcular o IMC, faremos uma pequena verificação das duas variáveis: se a primeira (alturaEhValida) for verdadeira **E** a segunda (pesoEhValido) também, o IMC será calculado.

Para verificarmos se uma condição **E\*\* outra são verdadeiras, utilizaremos o operador lógico representado por dois & (**&&\*\*):

var pesoEhValido = true;

var alturaEhValida = true;

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

console.log("Peso inválido!");

pesoEhValido = false;

}

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

console.log("Altura inválida!");

alturaEhValida = false;

}

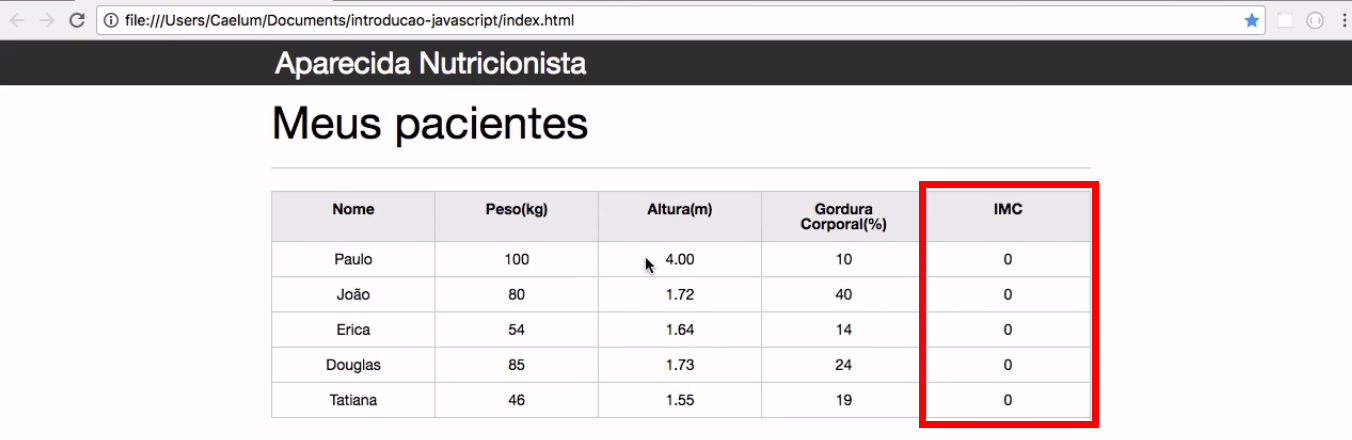
if (alturaEhValida && pesoEhValido) {

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

tdImc.textContent = imc;

}COPIAR CÓDIGO

Ao testarmos, iremos verificar se o peso ou a altura são inválidos e, neste caso, o IMC não será calculado. Na tela, veremos impresso o valor 0 na coluna da tabela.



Os usuários geralmente não ficam com o console do navegador aberto, então criaremos uma mensagem mais visível para eles. Na hora de verificar o peso e a altura, se algum deles for inválido, deixaremos uma mensagem mais significativa alterando-se o textContent:

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

console.log("Peso inválido!");

pesoEhValido = false;

tdImc.textContent = "Peso inválido!";

}

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

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida!";

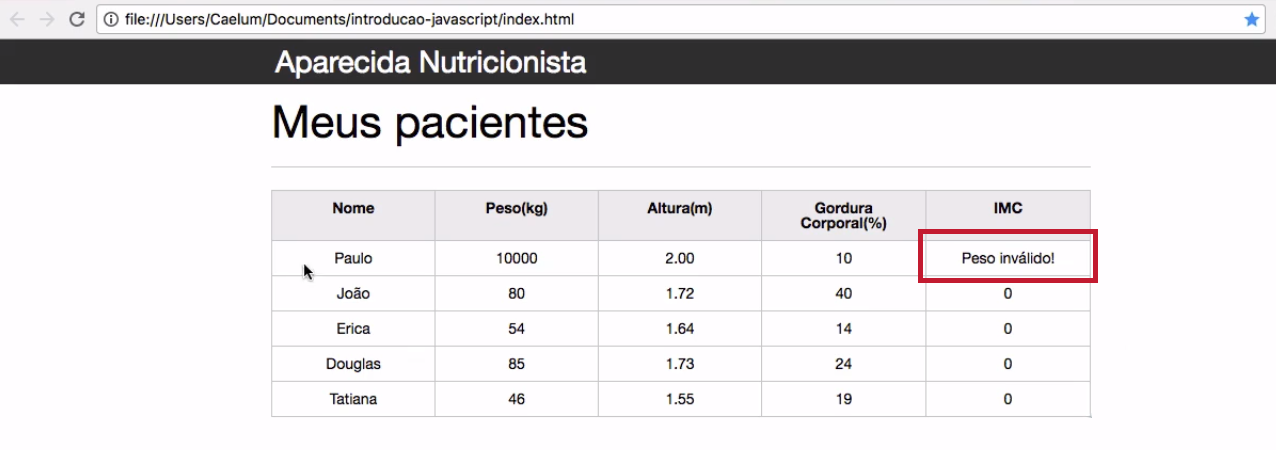
}

if (alturaEhValida && pesoEhValido){

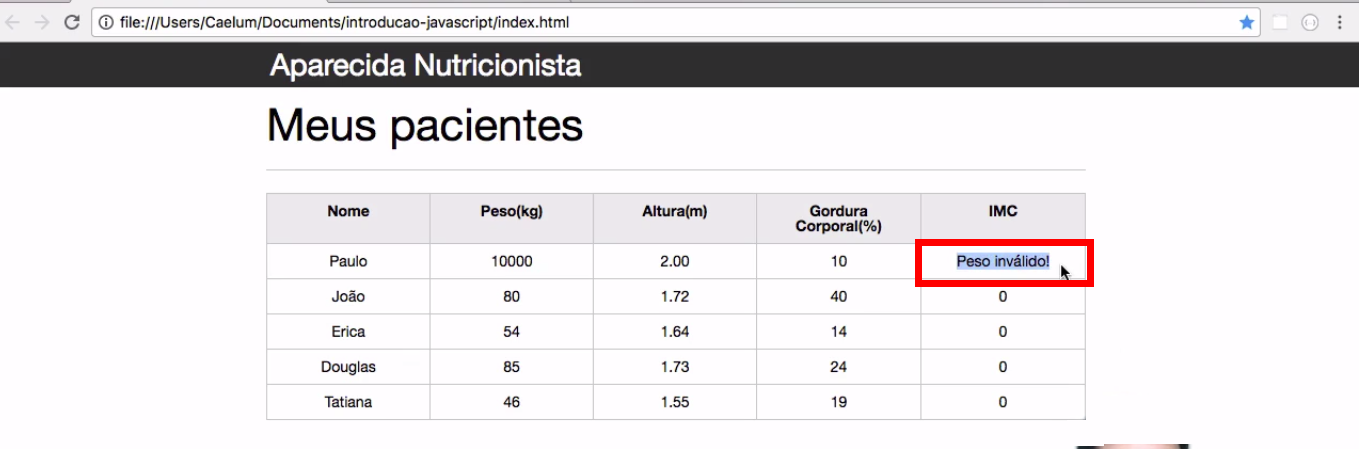
var imc = peso / (altura \* altura);

tdImc.textContent = imc;

}COPIAR CÓDIGO



Observe que alteramos o textContent para "Peso inválido" e "Altura inválida". Se colocarmos um peso ou uma altura inválidos no HTML e testar novamente a nossa página, a mensagem aparecerá conforme o esperado.



É possível refazermos o teste alterando o valor da altura no index.html:

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

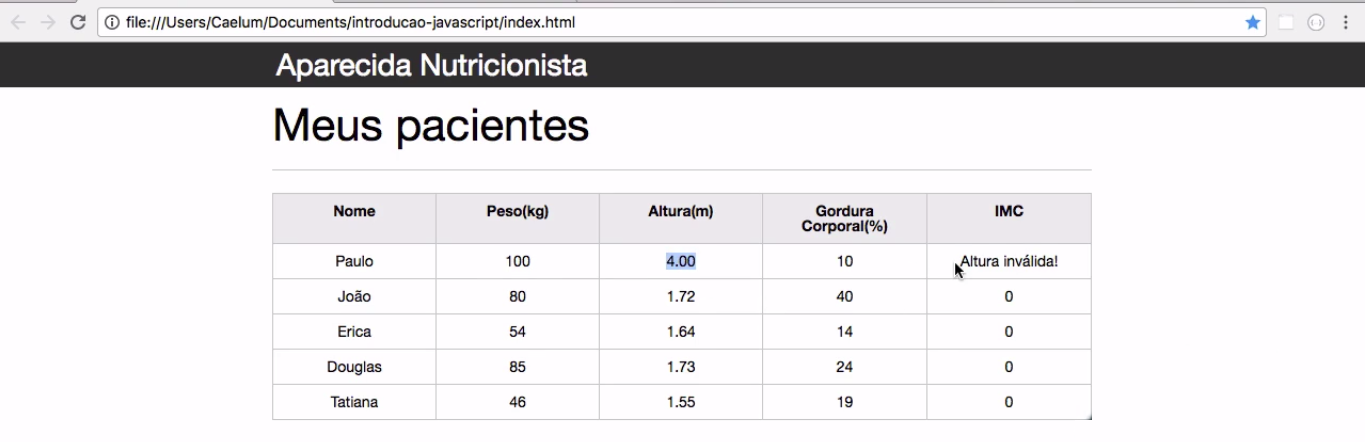
<td class="info-altura">4.00</td>

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>COPIAR CÓDIGO

Quando colocamos uma altura de 4.00, na tabela será exibida a mensagem de Altura Inválida!.



Há uma pequena falha nas nossas condições se o peso for igual a 1000. O IMC continuará sendo calculado. Faremos um ajuste no if para que isso ocorra, agora a condição será de >=:

//...

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

console.log("Peso inválido!");

pesoEhValido = false;

tdImc.textContent = "Peso inválido!";

}

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

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida!";

}

if (alturaEhValida && pesoEhValido){

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

tdImc.textContent = imc;

}COPIAR CÓDIGO

Tomamos o mesmo cuidado com os valores de peso. Observe que como não queremos que o valor 0 seja considerado válido para altura e peso, iremos alterar a condição para <=. Detectamos mais um erro que poderia ocorrer no código!

Com esta última validação, chegamos ao fim da aula. Vimos vários assuntos, como as lógicas com o if e sobre como calcular o IMC adicionando os parênteses. Mostramos o uso adequado dos operadores matemáticos e o operador lógico de **OU** (||), utilizado no if. Vimos também o &&, usado com o significado de **E** na condição.

**08Validando e exibindo resultado**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23604/next)

Aprendemos a obter um elemento do Mundo HTML e trazê-lo para o mundo JavaScript para podermos acessá-lo aplicando as mais diversas lógicas de programação.

Por exemplo, vamos revisar o código que extrai de uma página o peso e a altura para calcular o IMC. Primeiro, temos um fragmento do HTML:

<table>

<thead>

<tr>

<th>Nome</th>

<th>Peso(kg)</th>

<th>Altura(m)</th>

<th>Gordura Corporal(%)</th>

<th>IMC</th>

</tr>

</thead>

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>

</tbody>

</table>COPIAR CÓDIGO

Segundo, o código JavaScript que manipula esse HTML:

var trPaciente = document.querySelector('#primeiro-paciente');

var tdPeso = trPaciente.querySelector('.info-peso');

var tdAltura = trPaciente.querySelector('.info-altura');

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

var imc = peso / ( altura \* altura);COPIAR CÓDIGO

No entanto, não podemos simplesmente realizar o cálculo às cegas. Precisamos primeiro verificar se o peso e altura trazidos do mundo HTML são válidos. Não queremos magoar ninguém calculando um IMC indevido, certo? Aprendemos em lógica de programação a trabalharmos com a instrução if que espera receber true ou false. Sendo assim, podemos realizar críticas em nosso código da seguinte forma:

var trPaciente = document.querySelector('#primeiro-paciente');

var tdPeso = trPaciente.querySelector('.info-peso');

var tdAltura = trPaciente.querySelector('.info-altura');

var tdImc = trPaciente.querySelector('.info-imc');

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

var pesoEhValido = true; // assumindo de boa fé que o peso é válido

var alturaEhValida = true; // assumindo de boa fé que a altura é válida

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

pesoEhValido = false;

}

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

alturaEhValida = false;

}

if(pesoEhValido && alturaEhValida) {

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

}COPIAR CÓDIGO

Apesar do código acima ser funcional, em nenhum momento ele está exibindo o IMC calculado na página, muito menos as mensagens de erro caso peso ou altura sejam inválidos. Qual das opções abaixo exibe o IMC , inclusive as mensagens de erro?

Parte superior do formulário

* Alternativa correta



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

pesoEhValido = false;

tdPeso.textContent = 'peso inválido';

}

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

alturaEhValida = false;

tdAltura.textContent = 'altura inválida';

}

if(pesoEhValido && alturaEhValida) {

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

tdImc.textContent = imc;

}

* Alternativa correta



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

pesoEhValido = false;

tdPeso.value = 'peso inválido';

}

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

alturaEhValida = false;

tdAltura.value = 'altura inválida';

}

if(pesoEhValido && alturaEhValida) {

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

tdImc.value = imc;

}

* Alternativa correta



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

pesoEhValido = false;

peso.textContent = 'peso inválido';

}

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

alturaEhValida = false;

altura.textContent = 'altura inválida';

}

if(pesoEhValido && alturaEhValida) {

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

imc.textContent = imc;

}

Parte inferior do formulário

Veja que já somos capazes de capturar informações na página e ainda executar um monte de críticas para verificar a integridade dos dados antes de executamos nossa lógica com eles.

**09Informações escapando**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24295/next)

No sistema de RH do banco Finanção, as informações dos funcionários de uma determinada área são acessíveis somente aos gerentes *daquela* área. Um código Javascript que verifica se um funcionário do setor de TI poderia ou não ter acesso aos dados funciona como consta abaixo:

var setorTI = true;

var gerente = true;

if(setorTI == true || gerente == true){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

} COPIAR CÓDIGO

Com o tempo, os funcionários observaram que a validação continha **falhas**, pois pessoas que não deveriam ter acesso aos dados do setor TI estavam conseguindo mesmo assim.

Com base em seus conhecimentos de Javascript e Lógica de programação, marque todas as opções abaixo que indicam quais funcionários têm acesso às informações do setor de TI.

Parte superior do formulário

* Alternativa correta



Todos os funcionários do setor de TI.

Correta!

* Alternativa correta



Todos os funcionários do banco.

* Alternativa correta



Qualquer funcionário que fosse gerente no banco.

Correta!

* Alternativa correta



Apenas os gerentes que **não** sejam do setor de TI.

* Alternativa correta



Apenas os gerentes pertencentes ao setor de TI.

Parte inferior do formulário

Como a condição lógica testada é um OU ( representado pelas barrinhas || no Javascript ), caso uma condição OU outra seja atendida, o acesso às informações é liberado, bastando-se cumprir uma delas.

A primeira condição é setorTI == true, ou seja , se o funcionário for do setor de TI, ele já tem acesso, mesmo não sendo gerente.

A segunda condição é gerente == true, ou seja, se o funcionário for gerente, o acesso é permitido, mesmo não sendo do setor de TI.

No próximo exercício faremos a correção deste código para que o acesso seja

dado apenas aos gerentes deste setor.

**10Restrigindo o acesso**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24298/next)

Vimos que o banco Finanção está com uma falha na hora de verificar quem tem acesso a quais informações.

Qual alteração você faria no código abaixo para que apenas **gerentes** que são do **departamento de TI** tenham acesso aos dados de seus funcionários ?

var setorTI = true;

var gerente = true;

if(setorTI == true || gerente == true){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

}COPIAR CÓDIGO

Parte superior do formulário

* Alternativa correta



var setorTI = true;

var gerente = false;

if(setorTI == true , gerente == true ){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

}

O Javascript não suporta esta sintaxe com virgula em suas expressões.

* Alternativa correta



var setorTI = true;

var gerente = false;

if(setorTI == true && gerente == true){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

}

* Alternativa correta



var setorTI = true;

var gerente = false;

if(setorTI == true AND gerente == true ){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

}

Apesar da ideia ser correta, o operador AND não existe no Javascript. Se queremos testar a condição lógida de **E** devemos utilizar o operador &&.

Parte inferior do formulário

Basta apenas trocar a condição lógica de OU (||) para E (&&) assim , só quem vai ter o acesso permitido são as pessoas que são do setor de TI **E** também são gerentes.

Logo, o código final :

var setorTI = true;

var gerente = true;

if(setorTI == true && gerente == true){

console.log("Acesso permitido");

}else{

console.log("Acesso negado")

}

**12Mãos na Massa: Validando os dados do usuário**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24411/next)

Já estamos conseguindo calcular o IMC, porém falta validar os dados que vêm da tabela. Vamos lá:

1- O primeiro passo é fazer uma verificação do peso, vamos estabelecer que um peso é inválido se ele for menor ou igual a 0 quilos **OU** maior que 1000 quilos. Para isto, vamos utilizar um *if* para checar o peso, e a condição lógica de **OU** (||):

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

console.log("Peso inválido");

}COPIAR CÓDIGO

2- Não basta apenas exibir no console a validação. Vamos escrever na coluna do IMC também o erro de peso inválido:

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

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

}COPIAR CÓDIGO

3- Vamos repetir a mesma lógica para a altura, só que considerando inválido uma altura menor ou igual a 0 metros ou maior ou igual a 3 metros.

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

}COPIAR CÓDIGO

4- Agora não basta verificar se a altura ou o peso estão errados, só devemos calcular o IMC se ambos passarem na validação. Para isto, vamos criar duas variáveis, **pesoEhValido** e **alturaEhValida**, iniciar seus valores como **true**. Caso as validações falhem, trocaremos este valor para **false**:

var alturaEhValida = true;

var pesoEhValido = true;

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

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}COPIAR CÓDIGO

5- Agora vamos fazer uma lógica para apenas calcular o IMC se o peso **E\*\* a altura forem válidos. Vamos utilizar o operador lógico de \*\*E** (&&):

var alturaEhValida = true;

var pesoEhValido = true;

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

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

if(altura <= 0 || altura >= 3){

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}

if(pesoEhValido && alturaEhValida){

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

tdImc.textContent = imc;

}COPIAR CÓDIGO

O seu código final deve estar assim:

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

var tdAltura = paciente.querySelector(".info-altura");

var tdPeso = paciente.querySelector(".info-peso");

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

var altura = tdAltura.textContent;

var peso = tdPeso.textContent;

var alturaEhValida = true;

var pesoEhValido = true;

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

console.log("Peso inválido!");

tdPeso.textContent = "Peso inválido!";

pesoEhValido = false;

}

if (altura <= 0 || altura >= 3) {

console.log("Altura inválida!");

tdAltura.textContent = "Altura inválida!";

alturaEhValida = false;

}

if (alturaEhValida && pesoEhValido) {

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

tdImc.textContent = imc;

} else {

tdImc.textContent = "Altura e/ou peso inválidos!"

}COPIAR CÓDIGO

Agora estamos validando antes de realizar o cálculo do IMC!

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

Chegamos ao fim de mais um capítulo, e neste aqui utilizamos um pouco da lógica de programação e das funções do Javascript para validar nosso primeiro paciente.

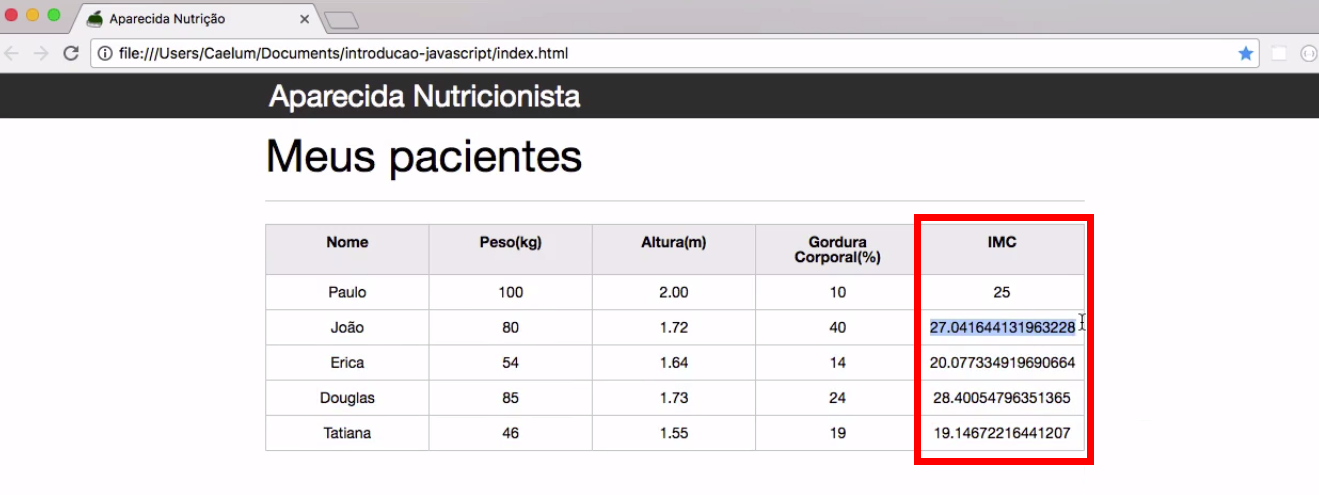
Neste capítulo vimos:

* O operador lógico de OU ( || )
* O operador lógico de E ( && )
* Busca de elementos
* **01Replicando a validação e o cálculo do IMC para todos os pacientes**

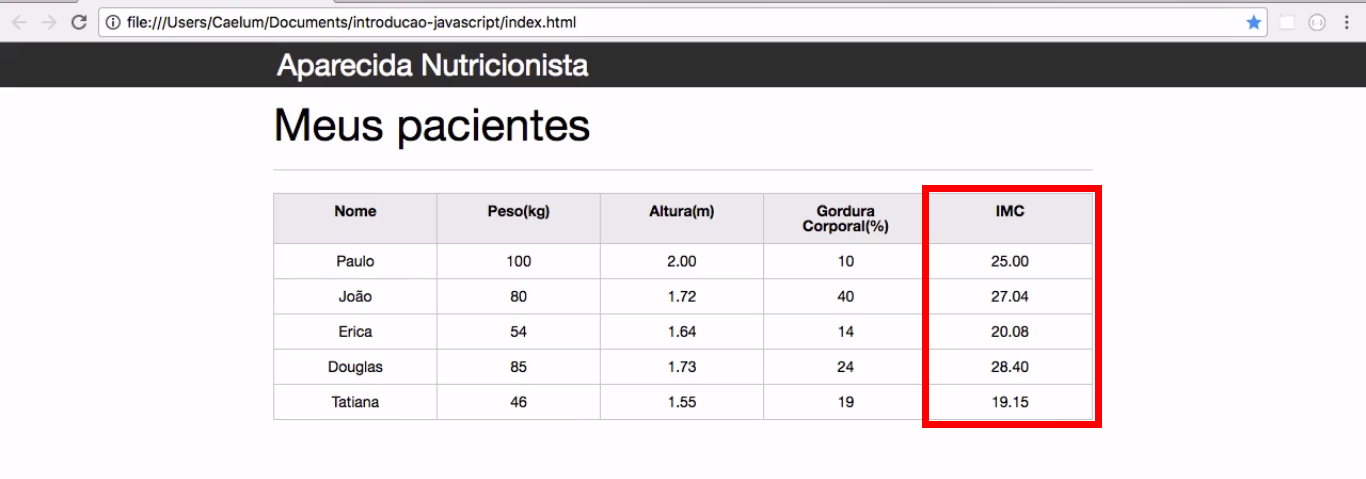
## Transcrição

* Conseguimos realizar a validação do primeiro paciente, calculando o IMC e verificando se a altura e o peso são válidos. A seguir, replicaremos esse código de validação e cálculo do IMC para os demais pacientes da tabela, como no trecho abaixo:
* <tbody id="tabela-pacientes">
* <tr class="paciente" id="primeiro-paciente">
* <td class="info-nome">Paulo</td>
* <td class="info-peso">1000</td>
* <td class="info-altura">2.00</td>
* <td class="info-gordura">10</td>
* <td class="info-imc">0</td>
* </tr>
* //...COPIAR CÓDIGO
* O código responsável por calcular o IMC do paciente e as validações dos dados é, praticamente, todo o arquivo principal.js. Então, bastaria copiarmos esse código, modificar a classe primeiro-paciente e os nomes das variáveis, que já validaríamos outro paciente.
* var paciente2 = document.querySelector("#segundo-paciente");
* var tdPeso2 = paciente.querySelector(".info-peso");
* var peso2 = tdPeso.textContent;
* var tdAltura2 = paciente.querySelector(".info-peso");
* var altura2 = tdAltura.textContent;COPIAR CÓDIGO
* No entanto, a **repetição de código** não é uma boa prática de programação, o ideal é sempre reaproveitar o trabalho já pronto. No caso, poderíamos fazer um loop - conhecido na lógica de programação -, para conseguirmos iterar por cada linha da tabela, ou seja, por cada paciente. Desta forma, selecionaremos os dados de cada um deles e calcularemos os IMCs.
* Então, se quisermos iterar por todos os pacientes, devemos selecioná-los - e não somente um deles, como estamos fazendo atualmente. Se analisarmos o código da tabela no index.html, veremos que **todos os pacientes** possuem a classe paciente:
* <tbody id="tabela-pacientes">
* <tr class="paciente" id="primeiro-paciente">
* <td class="info-nome">Paulo</td>
* <td class="info-peso">-100</td>
* <td class="info-altura">4.00</td>
* <td class="info-gordura">10</td>
* <td class="infor-imc">0</td>
* </tr>
* <tbody id="tabela-pacientes">
* <tr class="paciente" id="paciente">
* <td class="info-nome">João</td>
* <td class="info-peso">80</td>
* <td class="info-altura">1.72</td>
* <td class="info-gordura">40</td>
* <td class="info-imc">0</td>
* </tr>
* <!-- ... -->COPIAR CÓDIGO
* Logo, em vez de buscarmos pelo id primeiro-paciente, vamos buscar pela classe paciente. Testaremos isso:
* var pacientes = document.querySelector(".paciente");
* console.log(pacientes);COPIAR CÓDIGO
* Apesar de termos selecionado por classe, só nos foi retornado o primeiro paciente, a primeira linha com a classe paciente. Isso porque a função querySelector() nos retorna apenas **um elemento**, independentemente do que passarmos para a mesma. Se estivermos interessados em buscar vários elementos com a classe .paciente, devemos utilizar querySelectorAll():
* var pacientes = document.querySelectorAll(".paciente");
* console.log(pacientes);COPIAR CÓDIGO
* Essa função nos retorna um array no console, com todos os elementos que possuem a classe paciente.
* [tr#primeiro-paciente.paciente, tr.paciente, tr.paciente, tr.paciente, tr.paciente]COPIAR CÓDIGO
* Ao abrirmos o array, veremos todos os pacientes:
* NodeList[5]
* 0: tr#primeiro-paciente.paciente
* 1: tr.paciente
* 2: tr.paciente
* 3: tr.paciente
* 4: tr.paciente
* length: 5COPIAR CÓDIGO
* Veremos impresso também a propriedade length que nos informa que temos 5 elementos.

## Iterando sobre os pacientes

* Agora que temos todos os pacientes da tabela, devemos iterar por eles, e para cada um deles executaremos o código responsável por calcular o IMC e validar o peso e a altura.
* Há várias formas de fazermos isso, e uma delas, bastante conhecida por todo mundo que já viu algo de lógica de programação é o loop for. Ele receberá três argumentos: a declaração da variável inicial, até onde queremos que essa variável cresça, e o que queremos fazer no fim de cada iteração. Por exemplo:
* var pacientes = document.querySelectorAll(".paciente");
* for (var i = 0; i < 5; i++) {
* }COPIAR CÓDIGO
* Mas não queremos percorrer um número fixo, e sim percorrer a lista de pacientes. Todo array possui uma propriedade chamada length, que nos informa o seu tamanho. Então vamos usá-la para percorrer até o tamanho da lista de pacientes:
* var pacientes = document.querySelectorAll(".paciente");
* for (var i = 0; i < pacientes.length; i++) {
* console.log()
* }COPIAR CÓDIGO
* Deste modo, evitaremos trabalhar com um tamanho fixo de elementos.
* Cada elemento do array tem um índice, que começa da posição 0, o qual representará o primeiro paciente, o 1 será o segundo, e assim por diante. Para acessarmos o paciente do array, basta passarmos o índice entre colchetes para o mesmo. Esse índice será a variável i, que percorrerá do valor 0 até a última posição do array de pacientes **menos** 1:
* var pacientes = document.querySelectorAll(".paciente");
* for (var i = 0; i < pacientes.length; i++) {
* console.log(pacientes[i]);
* }COPIAR CÓDIGO
* Feito isso, temos acesso ao pacientes[i], e bastará movermos o código que calcula o IMC e faz a validação para dentro do for. Além disso, faremos um ajuste para não precisarmos alterar todo o código. Usaremos um pequeno "truque", criando a variável paciente, que será um atalho para pacientes[i]:
* var pacientes = document.querySelectorAll(".paciente");
* for (var i = 0; i < pacientes.length; i++) {
* var paciente = pacientes[i];
* var tdPeso = paciente.querySelector(".info-peso");
* var peso = tdPeso.textContent;
* var tdAltura = paciente.querySelector(".info-altura");
* var altura = tdAltura.textContent;
* 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";
* }
* if (altura <= 0 || altura >= 3.00) {
* console.log("Altura inválida!");
* alturaEhValida = false;
* tdImc.textContent = "Altura inválida";
* }
* if (alturaEhValida && pesoEhValido) {
* var imc = peso / (altura \* altura);
* tdImc.textContent = imc;
* }
* }COPIAR CÓDIGO
* De volta ao browser, veremos que conseguimos calcular o IMC de todos os pacientes!
* 

## Limitando as casas decimais de um número

* Alguns dos IMCs possuem diversas casas decimais, tornando o número imenso. Podemos melhorar isso utilizando uma função que limita esta quantidade. Essa função é toFixed(), que recebe como parâmetro a quantidade de casas decimais a serem exibidas depois do ponto.
* Vamos utilizá-la no momento em que o IMC é impresso na célula:
* if (alturaEhValida && pesoEhValido) {
* var imc = peso / (altura \* altura);
* tdImc.textContent = imc.toFixed(2);
* }COPIAR CÓDIGO
* Agora a visualização dos IMCs ficou um pouco melhor, sem tantos números nas casas decimais.
* 
* Vimos um loop simples, bastante utilizado na lógica de programação. Com o for, conseguimos validar o IMC e fazer os cálculos para todos os pacientes. Se os dados de algum paciente estiverem incorretos, um aviso será exibido na célula relacionada da tabela.

# 01Replicando a validação e o cálculo do IMC para todos os pacientes

**Transcrição**

Conseguimos realizar a validação do primeiro paciente, calculando o IMC e verificando se a altura e o peso são válidos. A seguir, replicaremos esse código de validação e cálculo do IMC para os demais pacientes da tabela, como no trecho abaixo:

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

<td class="info-peso">1000</td>

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

<td class="info-gordura">10</td>

<td class="info-imc">0</td>

</tr>

//...COPIAR CÓDIGO

O código responsável por calcular o IMC do paciente e as validações dos dados é, praticamente, todo o arquivo principal.js. Então, bastaria copiarmos esse código, modificar a classe primeiro-paciente e os nomes das variáveis, que já validaríamos outro paciente.

var paciente2 = document.querySelector("#segundo-paciente");

var tdPeso2 = paciente.querySelector(".info-peso");

var peso2 = tdPeso.textContent;

var tdAltura2 = paciente.querySelector(".info-peso");

var altura2 = tdAltura.textContent;COPIAR CÓDIGO

No entanto, a **repetição de código** não é uma boa prática de programação, o ideal é sempre reaproveitar o trabalho já pronto. No caso, poderíamos fazer um *loop* - conhecido na lógica de programação -, para conseguirmos iterar por cada linha da tabela, ou seja, por cada paciente. Desta forma, selecionaremos os dados de cada um deles e calcularemos os IMCs.

Então, se quisermos iterar por todos os pacientes, devemos selecioná-los - e não somente um deles, como estamos fazendo atualmente. Se analisarmos o código da tabela no index.html, veremos que **todos os pacientes** possuem a classe paciente:

<tbody id="tabela-pacientes">

<tr class="paciente" id="primeiro-paciente">

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

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

<td class="info-altura">4.00</td>

<td class="info-gordura">10</td>

<td class="infor-imc">0</td>

</tr>

<tbody id="tabela-pacientes">

<tr class="paciente" id="paciente">

<td class="info-nome">João</td>

<td class="info-peso">80</td>

<td class="info-altura">1.72</td>

<td class="info-gordura">40</td>

<td class="info-imc">0</td>

</tr>

<!-- ... -->COPIAR CÓDIGO

Logo, em vez de buscarmos pelo id primeiro-paciente, vamos buscar pela classe paciente. Testaremos isso:

var pacientes = document.querySelector(".paciente");

console.log(pacientes);COPIAR CÓDIGO

Apesar de termos selecionado por classe, só nos foi retornado o primeiro paciente, a primeira linha com a classe paciente. Isso porque a função querySelector() nos retorna apenas **um elemento**, independentemente do que passarmos para a mesma. Se estivermos interessados em buscar vários elementos com a classe .paciente, devemos utilizar querySelectorAll():

var pacientes = document.querySelectorAll(".paciente");

console.log(pacientes);COPIAR CÓDIGO

Essa função nos retorna um array no console, com todos os elementos que possuem a classe paciente.

[tr#primeiro-paciente.paciente, tr.paciente, tr.paciente, tr.paciente, tr.paciente]COPIAR CÓDIGO

Ao abrirmos o array, veremos todos os pacientes:

NodeList[5]

0: tr#primeiro-paciente.paciente

1: tr.paciente

2: tr.paciente

3: tr.paciente

4: tr.paciente

length: 5COPIAR CÓDIGO

Veremos impresso também a propriedade length que nos informa que temos 5 elementos.

**Iterando sobre os pacientes**

Agora que temos todos os pacientes da tabela, devemos iterar por eles, e para cada um deles executaremos o código responsável por calcular o IMC e validar o peso e a altura.

Há várias formas de fazermos isso, e uma delas, bastante conhecida por todo mundo que já viu algo de lógica de programação é o *loop* for. Ele receberá três argumentos: a declaração da variável inicial, até onde queremos que essa variável cresça, e o que queremos fazer no fim de cada iteração. Por exemplo:

var pacientes = document.querySelectorAll(".paciente");

for (var i = 0; i < 5; i++) {

}COPIAR CÓDIGO

Mas não queremos percorrer um número fixo, e sim percorrer a lista de pacientes. Todo *array* possui uma propriedade chamada length, que nos informa o seu tamanho. Então vamos usá-la para percorrer até o tamanho da lista de pacientes:

var pacientes = document.querySelectorAll(".paciente");

for (var i = 0; i < pacientes.length; i++) {

console.log()

}COPIAR CÓDIGO

Deste modo, evitaremos trabalhar com um tamanho fixo de elementos.

Cada elemento do array tem um índice, que começa da posição 0, o qual representará o primeiro paciente, o 1 será o segundo, e assim por diante. Para acessarmos o paciente do array, basta passarmos o índice entre colchetes para o mesmo. Esse índice será a variável i, que percorrerá do valor 0 até a última posição do array de pacientes **menos** 1:

var pacientes = document.querySelectorAll(".paciente");

for (var i = 0; i < pacientes.length; i++) {

console.log(pacientes[i]);

}COPIAR CÓDIGO

Feito isso, temos acesso ao pacientes[i], e bastará movermos o código que calcula o IMC e faz a validação para dentro do for. Além disso, faremos um ajuste para não precisarmos alterar todo o código. Usaremos um pequeno "truque", criando a variável paciente, que será um atalho para pacientes[i]:

var pacientes = document.querySelectorAll(".paciente");

for (var i = 0; i < pacientes.length; i++) {

var paciente = pacientes[i];

var tdPeso = paciente.querySelector(".info-peso");

var peso = tdPeso.textContent;

var tdAltura = paciente.querySelector(".info-altura");

var altura = tdAltura.textContent;

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";

}

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

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida";

}

if (alturaEhValida && pesoEhValido) {

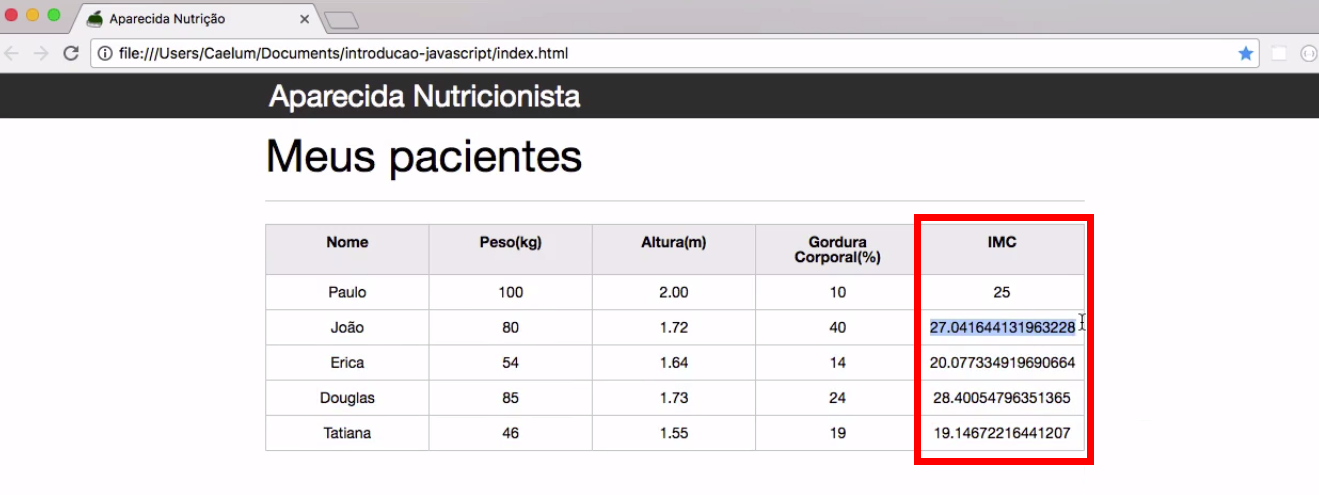
var imc = peso / (altura \* altura);

tdImc.textContent = imc;

}

}COPIAR CÓDIGO

De volta ao browser, veremos que conseguimos calcular o IMC de todos os pacientes!



**Limitando as casas decimais de um número**

Alguns dos IMCs possuem diversas casas decimais, tornando o número imenso. Podemos melhorar isso utilizando uma função que limita esta quantidade. Essa função é toFixed(), que recebe como parâmetro a quantidade de casas decimais a serem exibidas depois do ponto.

Vamos utilizá-la no momento em que o IMC é impresso na célula:

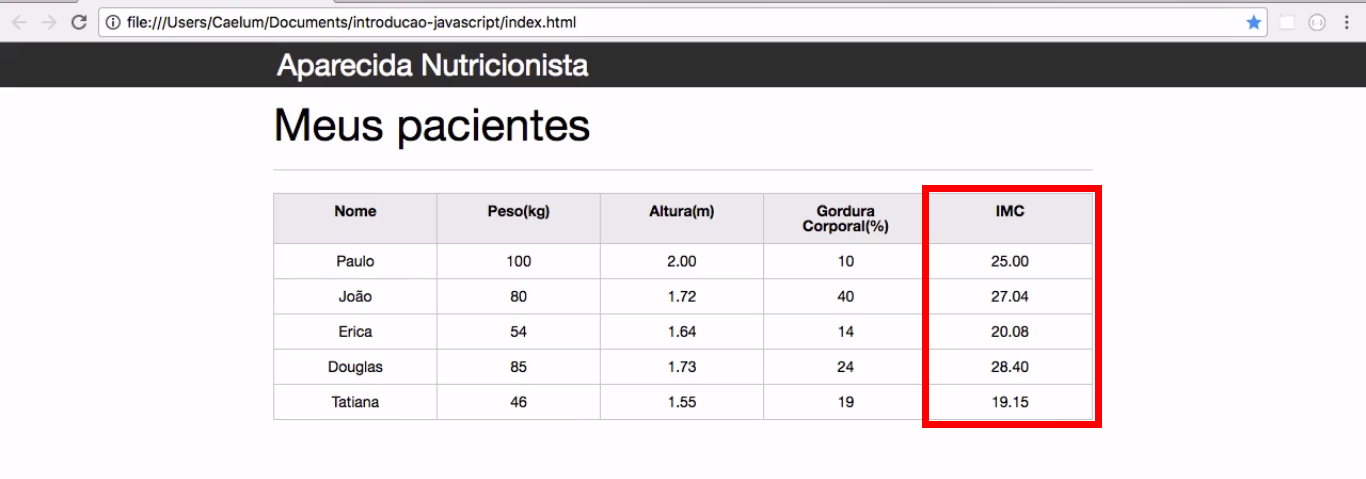
if (alturaEhValida && pesoEhValido) {

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

tdImc.textContent = imc.toFixed(2);

}COPIAR CÓDIGO

Agora a visualização dos IMCs ficou um pouco melhor, sem tantos números nas casas decimais.



Vimos um *loop* simples, bastante utilizado na lógica de programação. Com o for, conseguimos validar o IMC e fazer os cálculos para todos os pacientes. Se os dados de algum paciente estiverem incorretos, um aviso será exibido na célula relacionada da tabela.

**02Estilos com JavaScript**

**Transcrição**

Faremos ajustes para deixar os nossos erros mais identificáveis. Por enquanto, temos somente cinco pacientes, e para identificar um deles cujo valor de peso ou altura seja inválido, vamos procurar pela mensagem exibida na última coluna. Imagine se a tabela tiver 50 ou 100 pacientes cadastrados? Será preciso ficar *scrollando* a página e procurar pelas mensagens!

Por isso, deixaremos uma marcação visual para a nutricionista, indicando as linhas com erros. Faremos com que a linha com peso ou altura inválido exiba um fundo vermelho.

Vamos acessar o arquivo principal.js e analisar o trecho em que realizamos as validações:

var pesoEhValido = true;

var alturaEhValida = true;

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

console.log("Peso inválido!");

pesoEhValido = false;

tdImc.textContent = "Peso inválido";

}

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

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida";

}COPIAR CÓDIGO

Para alterarmos a cor do fundo, selecionaremos a linha (cada tag tr) - o que já fizemos anteriormente e será representada pela variável paciente. Nosso objetivo é modificar o **estilo** do elemento, portanto utilizaremos a propriedade style, com os estilos aplicados. Ou seja, vamos alterar o CSS aplicado. Se adicionarmos style.color, poderíamos alterar a cor da fonte na linha.

var pesoEhValido = true;

var alturaEhValida = true;

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

console.log("Peso inválido!");

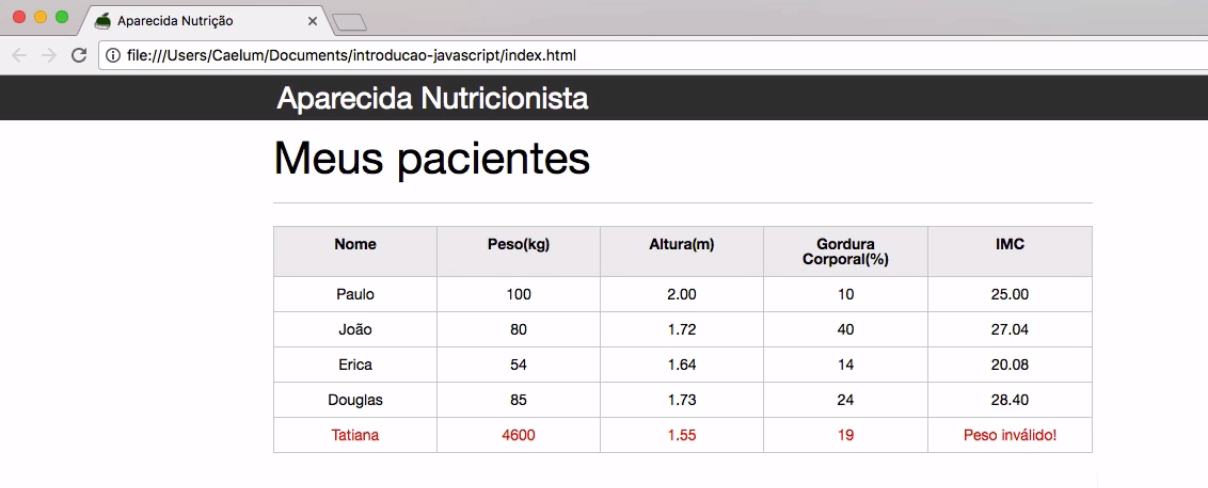
pesoEhValido = false;

tdImc.textContent = "Peso inválido";

paciente.style.color = "red";

}COPIAR CÓDIGO

Quando atualizarmos a página veremos que a cor da fonte foi alterada.



Assim, por conseguirmos alterar a cor da fonte usando a propriedade color, poderemos modificar o fundo da linha ao adicionarmos a propriedade backgroundColor. Porém, se tentarmos escrever da mesma forma como é feito no CSS (background-color), nosso código não funcionará. Quando queremos modificar um estilo com duas palavras, o hífen (-) não é aceito pela linguagem JavaScript. Teremos que utilizar o padrão ***Camel Case*** - no qual as palavras compostas ou frases são iniciadas com maiúsculas e unidas sem espaços.

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

console.log("Peso inválido!");

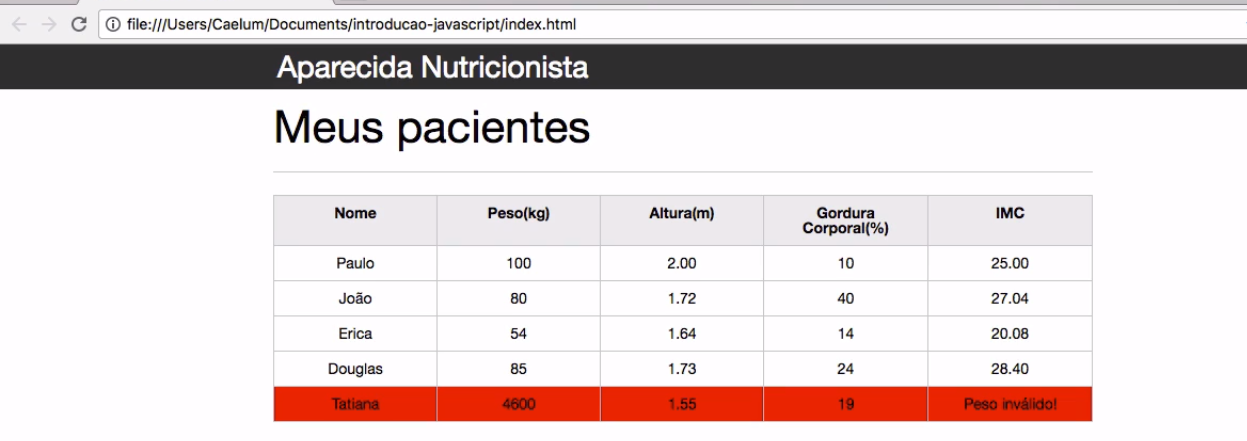
pesoEhValido = false;

tdImc.textContent = "Peso inválido";

paciente.style.backgroundColor = "red";

}COPIAR CÓDIGO

Sempre que houver um erro, ele ficará em evidência na tabela.



Vamos alterar a cor para um tom mais claro: lightcoral.

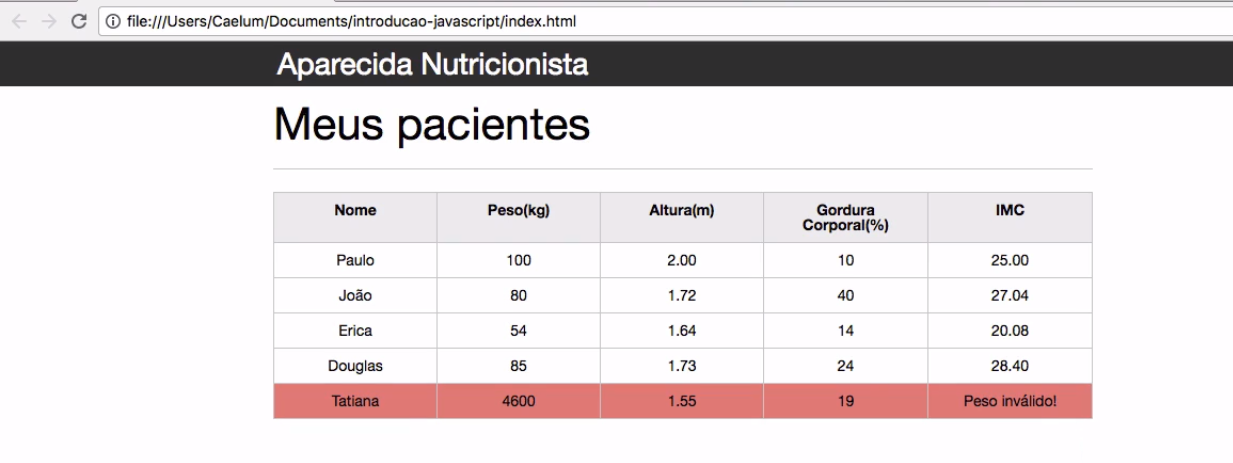
if (altura <= 0 || altura >= 3.00) {

console.log("Altura inválida!");

alturaEhValida = false;

paciente.style.backgroundColor = "lightcoral";

}COPIAR CÓDIGO



Em seguida, vamos usar a propriedade style para sinalizar o problema na altura.

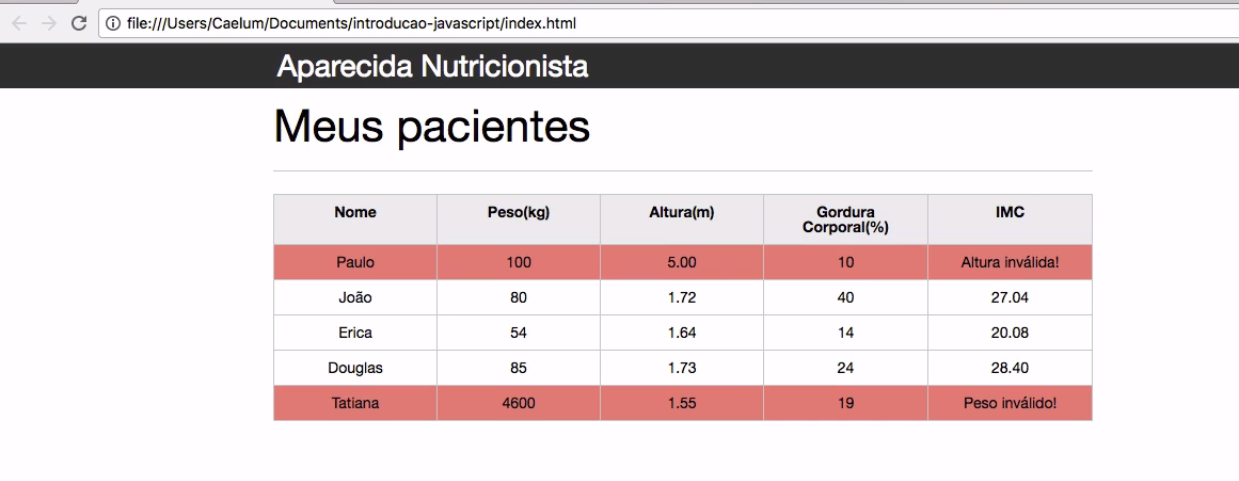
if (altura <= 0 || altura >= 3.00) {

console.log("Altura inválida!");

alturaEhValida = false;

paciente.style.backgroundColor = "lightcoral";

}COPIAR CÓDIGO



Sempre que ocorre um erro, ele ficará fácil de ser identificado na tabela. Conseguiremos fazer a linha ficar com o fundo vermelho quando algum dado for inválido.

No entanto, imagine a situação em que o designer decide que a cor de fundo não será mais vermelha, e sim laranja. Devemos nos lembrar de modificá-la nas duas verificações.

O local correto para definirmos o estilo da página é no arquivo CSS. Novamente, é importante separarmos esses ajuste do arquivo HTML.

É importante mantermos essa separação dos "mundos": o HTML é utilizado para marcar os elementos, o JavaScript para adicionar dinamismo e manipular a página, enquanto o CSS é responsável por estilizar. Logo, o ideal não é modificar o estilo da linha no código JavaScript, então qual é o meio termo entre não modificar o estilo diretamente, mas conseguir alterá-lo, no JavaScript?

É uma boa prática, em vez de adicionarmos um estilo, incluirmos uma classe ao elemento. Então criaremos uma nova classe em index.css com dados do paciente inválido. A classe paciente-invalido irá indicar com quais cores trabalharemos nestes casos:

.paciente-invalido {

background-color: lightcoral;

}COPIAR CÓDIGO

De volta ao JavaScript, assim como temos o style\*\* para acesso aos estilos e o textContent para acessarmos o conteúdo de texto de um elemento, teremos a classList para acessarmos as classes. E usando o método add, adicionaremos a classe desejada, no caso, paciente-invalido:

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");

}COPIAR CÓDIGO

Se o designer quiser modificar a cor de fundo, deverá modificá-la no CSS e não mais no JavaScript. Inspecionando-se uma linha cujo fundo possui alteração de cor, verificaremos no console que a classe paciente-invalido foi adicionada.

O próximo passo será incluir classList.add("paciente-invalido") também na validação da altura:

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

console.log("Altura inválida!");

alturaEhValida = false;

tdImc.textContent = "Altura inválida";

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

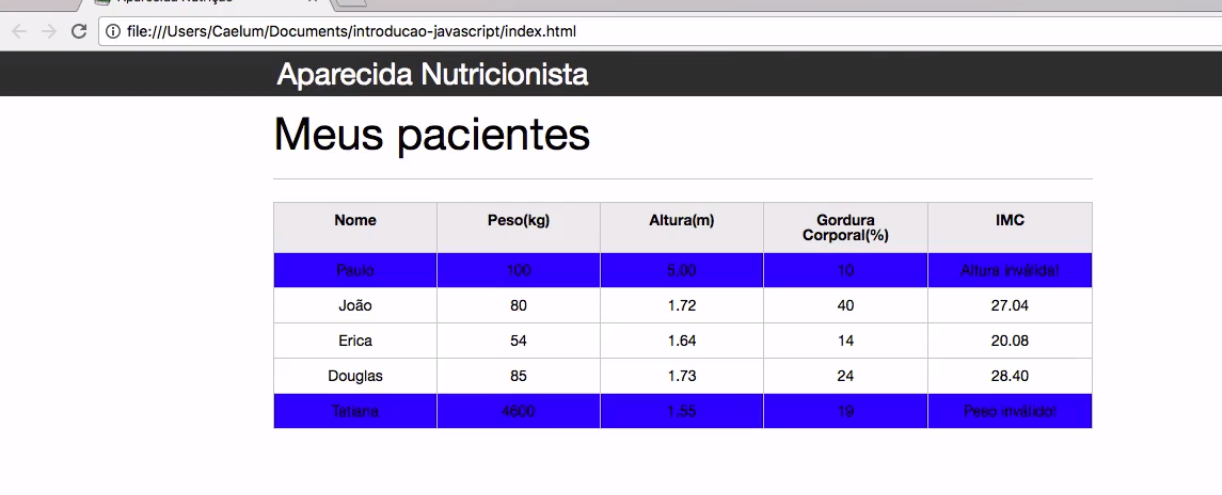
}COPIAR CÓDIGO

Agora, sempre que for necessária uma alteração na parte visual da página, elas serão feitas no arquivo index.css. Caso queiramos que o fundo da linha com o dado inválido fique com a cor azul, por exemplo, devemos alterar a classe .paciente-invalido no código CSS.

.paciente-invalido{

background-color: blue;

}COPIAR CÓDIGO



Não precisaremos localizar todos os pontos onde escrevemos lightcoral no código. Evitaremos o risco de esquecermos a alteração em algum trecho. Ao adicionarmos uma classe no CSS, nosso código fica organizado.

Vamos deixar a cor do fundo da linha com o erro como estava:

.paciente-invalido{

background-color: lightcoral;

}COPIAR CÓDIGO

Encerramos mais uma aula do curso de JavaScript, no qual apresentamos como fazer o cálculo do IMC para todos os pacientes da tabela, em vez de um. Mostramos como selecionar vários elementos do HTML simultaneamente, usando a função querySelectorAll(). Com ela podemos trazer uma lista de objetos selecionados contendo a classe .paciente. Depois, percorremos essa lista com o laço for. Trabalhamos também com a propriedade length, que nos retorna o tamanho do array, além de aprendermos como acessar uma posição especifica do array utilizando o [].

Por fim, conseguimos calcular o IMC dos pacientes e arredondamos os resultados com valores decimais, fazendo uso do método toFixed(). E manipulamos o estilo de um objeto no JS, por meio da propriedade style - que não deve ser utilizada com frequência. Na aula, explicamos a importância de não misturarmos os "mundos" de linguagens diferentes, no caso, quando queremos alterar o visual da página, devemos manipular as classes do arquivo CSS. A nova classe será adicionada com classList.add() no JavaScript. Este tipo de ajuste é uma boa prática que deve ser adotada por facilitar a manutenção do código.

Obrigado por ter chegado até aqui e espero você na próxima aula!

**03Calculando gastos**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23619/next)

Jéssica criou um aplicativo de finanças para ter um melhor controle de seus gastos, visto que ela estava gastando muito com eletrônicos.

Ela queria ver qual a porcentagem de sua renda era destinada a cada categoria de gastos e para isto ela usava um simples cálculo:

var despesaCategoria = 1532;

var rendaTotal = 6255;

var porcentagemCategoria = (despesaCategoria/rendaTotal) \* 100;

console.log(porcentagemCategoria);COPIAR CÓDIGO

Jéssica viu que ela não tinha interesse em saber a porcentagem com tantas casa decimais , como neste exemplo que iria ficar um número muito feio (24.49240607513989). Para o caso dela, **uma** casa decimal seria o suficiente.

Qual alteração no código dela faria com que o número de casas decimais ficasse limitado a uma casa decimal ?

Parte superior do formulário

* Alternativa correta



var despesaCategoria = 1532.0;

var rendaTotal = 6255.0;

var porcentagemCategoria = despesaCategoria/rendaTotal \* 100.0;

console.log(porcentagemCategoria);

* Alternativa correta



var despesaCategoria = 1532;

var rendaTotal = 6255;

var porcentagemCategoria = despesaCategoria/rendaTotal \* 100;

var porcentagemDecimal = porcentagemCategoria.toDecimal(1);

console.log(porcentagemDecimal);

* Alternativa correta



var despesaCategoria = 1532;

var rendaTotal = 6255;

var porcentagemCategoria = despesaCategoria/rendaTotal \* 100;

console.log(porcentagemCategoria.toFixed(1));

Parte inferior do formulário

Podemos controlar o número de casas decimais exibidas através da função **.toFixed()**.

A função toFixed() deve ser usadas em números, logo se queremos transformar o porcentagemCategoria em um número com apenas uma casa decimal, devemos fazer:

porcentagemCategoria.toFixed(1);COPIAR CÓDIGO

O número passado entre parêntenses deve ser o número de casas decimais para formatar o porcentagemCategoria.

**04Seleção falhando**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23665/next)

João está montando um script em Javascript que deve pegar todos os nomes dos produtos de um e-commerce.

O HTML da lista é como o abaixo:

<ul class="lista-produtos">

<li class="produto">Web cam</li>

<li class="produto">Microfone</li>

<li class="produto">Fundo verde</li>

<li class="produto">Notebook</li>

</ul>COPIAR CÓDIGO

E o script do João é o seguinte:

var produtos = document.querySelector("lista-produtos");

for( var i=0 ; i < produtos.length ; i++){

var produto = produtos[i];

var nomeDoProduto = produto.textContent;

console.log(nomeDoProduto);

}COPIAR CÓDIGO

Porém o script não funciona como João espera.

Quais alternativas abaixo apontam para os possíveis erros do código de João ?

Parte superior do formulário

* Alternativa correta



João utilizou a propriedade textContent errada, ele poderia simplesmente imprimir o produto[i].

* Alternativa correta



João está utilizando a função errada, como ele quer selecionar diversos elementos, ele deve usar a função querySelectorAll().

* Alternativa correta



João está usando a propriedade .length que está escrito errado, o correto é .lenght.

* Alternativa correta



João esqueceu de colocar o **.** no seletor de sua função querySelector() e trocar o seletor para apenas "produto". Dessa forma, o seletor correto seria ".produto".

Parte inferior do formulário

João cometeu dois erros ao querer selecionar todos os produtos da lista.

O primeiro foi o uso da função de seleção errada, afinal se ele queria selecionar todos os produtos ele deveria utilizar a função querySelectorAll(), que retornar um array com todo mundo que atende critério de seleção.

O segundo erro foi no parâmetro da função seletora, se ele deseja buscar por todos os produtos que tem **classe** produto, ele deve utilizar o seletor de classe que é **ponto**.

**05Um parágrafo azul**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23671/next)

Qual dos códigos abaixo altera a **cor da letra** do parágrafo abaixo para a cor **azul** ?

<p id="latim"> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. </p>

<script>

var paragrafo = document.querySelector("#latim");

// Código aqui!

</script>COPIAR CÓDIGO

Parte superior do formulário

* Alternativa correta



paragrafo.style.color = "blue";

* Alternativa correta



paragrafo.color = "blue";

* Alternativa correta



paragrafo.css.color = "blue";

* Alternativa correta



paragrafo.styles.color = "blue";

Parte inferior do formulário

Se queremos modificar um **estilo** de um elemento selecionado com o Javascript, devemos primeiro acessar a propriedade **style** , que contêm todos os estilos daquele elemento. Depois que temos a propriedade **.style** em mãos, devemos especificar qual estilo desejamos alterar , que neste caso é a cor do texto ou seja o estilo **color**.

paragrafo.style.color;COPIAR CÓDIGO

Só que não basta pegar o estilo, devemos atribuir a ele um novo valor, como neste caso desejamos alterar para a cor azul, vamos colocá-lo com valor de **blue**:

paragrafo.style.color = "blue";COPIAR CÓDIGO

O interessante é que podemos colocar como valor, qualquer valor que seria válido no CSS, então se quiséssemos poderíamos utilizar o código RGB para representar a cor:

paragrafo.style.color = "rgb(0,0,255)";COPIAR CÓDIGO

Ambos os códigos obtêm o mesmo resultado!

**06Mas e se for o fundo ?**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23679/next)

E se mudarmos de ideia e, em vez da **cor** do parágrafo, quisermos que o **fundo** do último exercício seja azul?

<p id="latim"> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. </p>

<script>

var paragrafo = document.querySelector("#latim");

// Código aqui!

</script>COPIAR CÓDIGO

Qual código abaixo deixa o **fundo** do parágrafo azul?

Parte superior do formulário

* Alternativa correta



paragrafo.style.background\_color = "blue";

* Alternativa correta



paragrafo.style.background-color = "blue";

* Alternativa correta



paragrafo.style.backgroundColor = "blue";

O correto é usar o padrão camel**C**ase quando estamos tentando acessar um estilo que possuí duas palavras no CSS.

Parte inferior do formulário

Já vimos que para alterar o estilo de um elemento devemos acessar a propriedade style, seguido do estilo que será modificado. Porém , quando o nome do estilo possui duas palavras no CSS - como é o caso de background-color - devemos representá-lo com o estilo camelCase, no qual a primeira palavra fica em caixa baixa (minúsculo) e a letra inicial das palavras seguintes ficam em caixa alta (maiúsculo). Desta forma, a propriedade background-color deverá ser escrita da seguinte maneira no código: **backgroundColor**.

Ou seja, para alterar a cor de fundo do parágrafo para azul, utilizaremos o código abaixo:

<script>

var paragrafo = document.querySelector("#latim");

paragrafo.style.backgroundColor = "blue";

</script>COPIAR CÓDIGO

Isto é válido para qualquer propriedade do CSS que tenha duas palavras, como background-color, font-size, font-style, border-radius...etc!

**07Acessando as classes**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23669/next)

Qual propriedade abaixo nos dá acesso à lista das classes de um HTML selecionado pela função querySelectorAll?

Parte superior do formulário

* Alternativa correta



.class

* Alternativa correta



.classes

* Alternativa correta



.addClass

* Alternativa correta



.classArray

* Alternativa correta



.classList

.classList é a única das opções que existe no JavaScript!

Parte inferior do formulário

Quando selecionamos um elemento com as funções de querySelector, elas nos devolvem um objeto que tem algumas propriedades especiais, que falam sobre as características do HTML selecionado. Uma dessas propriedades é a **.classList**, que como o nome indica nos mostras **classes** que aquele HTML tem.

Experimente usar o seguinte código:

<h1 class="titulo principal azul"> Sou o titulo principal! </h1>COPIAR CÓDIGO

E no arquivo JavaScript, peça para imprimir o classList da tag h1:

var titulo = document.querySelector("h1");

console.log(titulo.classList);COPIAR CÓDIGO

Será exibido no seu console, uma estrutura de lista do JavaScript com todas as classes daquele <h1>:

//resultado

["titulo","principal","azul"]

**08Modificando os titulos**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23670/next)

Rômulo deseja que todos os H2's de sua página tenham a classe "titulo", que modifica a fonte dos H2's e deixa eles com um destaque específico.

Qual dos códigos abaixo implementa uma solução em Javascript para que todos os <h2> da página ganhem a classe "titulo"?

Parte superior do formulário

* Alternativa correta



var titulos = document.querySelectorAll("h2");

for(var i = 0 ; i < titulos.length ; i++){

var titulo = titulos[i];

titulo.addClass("titulo");

}

* Alternativa correta



var titulos = document.querySelectorAll("h2");

for(var i = 0 ; i < titulos.length ; i++){

var titulo = titulos[i];

titulo.classList.add("titulo");

}

* Alternativa correta



var titulos = document.querySelectorAll("h2");

for(var i = 0 ; i < titulos.length ; i++){

var titulo = titulos[i];

titulo.classList.add(".titulo");

}

* Alternativa correta



var titulos = document.querySelectorAll("h2");

for(var i = 0 ; i < titulos.length ; i++){

var titulo = titulos[i];

titulo.classList.addClass("titulo");

}

Parte inferior do formulário

A função responsável por adicionar uma classe é a função .add(), que recebe como parâmetro o nome da classe que queremos adicionar!

Só devemos lembrar de chamar esta função depois de acessarmos a propriedade .classList, que contêm as classes do HTML selecionado. Então, para resolver o problema de Rômulo, devemos utilizar o código abaixo:

var titulos = document.querySelectorAll("h2");

for(var i = 0 ; i < titulos.length ; i++){

var titulo = titulos[i];

titulo.classList.add("titulo");

}

**09Mãos na Massa: O IMC de todos os pacientes**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24419/next)

**Começando deste ponto ?**

Começando deste ponto? Você pode fazer o [DOWNLOAD](https://s3.amazonaws.com/caelum-online-public/introducao-javascript/stages/introducao-javascript-capitulo-3.zip) completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

Por enquanto nosso código só calcula e valida o IMC de **um** único paciente. Vamos atacar este problema neste capítulo para que o IMC de todos os pacientes sejam carregados:

1- Primeiro passo então é deixar de selecionar apenas um paciente e passar a selecionar **todos** os pacientes. Faremos isto buscando por todas as <tr> que tem a classe paciente. Claro, devemos também utilizar outra função, a função querySelectorAll() que é capaz de retornar mais de um elemento quando faz a busca no HTML. Faça a substituição no seu código para trocar o nome da variável paciente para pacientes e também faça a substituição da função e do seletor:

var pacientes = document.querySelectorAll(".paciente");

console.log(paciente);COPIAR CÓDIGO

2- Recarregando a página agora, veremos que o console dará um erro, mas é por que trocamos o nome da variável e ainda não adaptamos nosso código para trabalhar com um *array* de pacientes em vez de um único. Para validar as informações e calcular o IMC de cada paciente do array precisamos **percorrer** este array utilizando um clássico **for**:

for(var i = 0; i < pacientes.length ; i++){

var paciente = pacientes[i];

console.log(paciente);

}COPIAR CÓDIGO

3- Agora que conseguimos percorrer a lista de pacientes, basta mover o nosso código que validava e calculava o IMC do paciente para dentro da tabela. O código final deve ficar assim:

var pacientes = document.querySelectorAll(".paciente");

for(var i = 0; i < pacientes.length ; i++){

var paciente = pacientes[i];

var tdPeso = paciente.querySelector(".info-peso");

var tdAltura = paciente.querySelector(".info-altura");

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

var peso = tdPeso.textContent;

var altura = tdAltura.textContent;

var alturaEhValida = true;

var pesoEhValido = true;

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

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

}

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

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

}

if(pesoEhValido && alturaEhValida){

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

tdImc.textContent = imc;

}

}COPIAR CÓDIGO

4- Por último, podemos limitar o número de casas decimas do IMC utilizando a função .toFixed(). Faça a substuição para imprimir o IMC na tabela com apenas duas casas decimais:

// Restante do código

if(pesoEhValido && alturaEhValida){

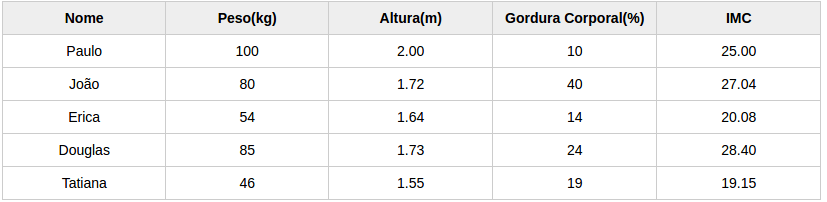
var imc = peso / ( altura \* altura);

//ALTERAÇÃO AQUI...

tdImc.textContent = imc.toFixed(2);

}

} COPIAR CÓDIGO



Agora cada paciente deve estar com IMC sendo cálculado!

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

Estamos calculando o IMC de todos os pacientes, mas no próximo exercício vamos ver um jeito melhor de exibir os erros!

**10Mãos na Massa: Melhorando a visualização do erro**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/24422/next)

Vamos melhorar um pouco a visualização de nossa tabela, para que os erros pintem a linha de vermelho além escrever na coluna de IMC quando ocorrer um erro.

1- Vamos trocar a cor da linha colocando uma nova classe na mesma, que será a classe paciente-invalido. Como primeiro passo crie uma nova classe no seu arquivo index.css como a classe abaixo:

.paciente-invalido{

background-color: lightcoral;

}COPIAR CÓDIGO

Nossa classe troca o background-color do elemento para a cor lightcoral, que é um vermelho claro.

2- Agora que temos uma classe com o estilo que queremos, devemos aplicá-la quando acontecer um erro, e os erros são detectados dentro de cada if. No if de peso, adicione a classe paciente-invalido a linha da tabela que isto ocorreu ( ou seja, na variável paciente):

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

console.log("Peso inválido");

tdImc.textContent = "Peso inválido!";

pesoEhValido = false;

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

}COPIAR CÓDIGO

Estamos alterando o estilo colocando uma classe para facilitar a manutenção de nosso código, afinal se precisarmos alterar a cor, basta alterar o valor da propriedade background-color no CSS.

2- Faça a mesma coisa agora para quando a altura for inválida:

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

console.log("Altura inválida");

tdImc.textContent = "Altura inválida!";

alturaEhValida = false;

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

}COPIAR CÓDIGO

Pronto, agora temos um visual melhor quando ocorrer um erro em nossa tabela!

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

Neste capítulo aprendemos a lidar uma coleção de elementos do HTML e vimos:

* Como selecionar diversos elementos com a função querySelectorAll()
* Relembramos um método clássico de iteração com o **for**
* Vimos como manipular estilos com a propriedade .style
* Vimos que devemos utilizar camelCase quando queremos modificar uma propriedade que tenha duas palavras, como a background-color
* A boa prática de não alterar o estilo diretamente no Javascript e sim modificar as classes dos elementos com a propriedade .classList

# 01Escutando eventos

**Transcrição**

Nossa tabela faz o cálculo do IMC de todos pacientes com os dois tipos de validações, inclusive sinalizando visualmente quando uma linha tem problemas, e assim facilitaremos a vida da Aparecida em relação ao cadastro de pacientes.

Atualmente, para adicionar um novo paciente, ela deve acessar o arquivo index.html e incluir uma nova <tr> na tabela, juntamente com as tags <td>, e modificar os dados da pessoa. Depois disso, uma nova linha surgirá na tabela.

Ou seja, o HTML é alterado manualmente, o que não é prático. Seria muito simples se em vez de adicionarmos o paciente diretamente no HTML, houvesse um formulário no site, o qual ela pudesse preencher com os dados do paciente. E, após clicar-se no botão "Adicionar", o paciente seria adicionado à tabela. É exatamente isso que iremos implementar.

Como esse é um curso focado em JavaScript, disponibilizaremos o formulário pronto, basta adicioná-lo abaixo do fechamento da tag <main>, e antes de <script>:

<!-- ... -->

<section class="container">

<h2 id="titulo-form">Adicionar novo paciente</h2>

<form id="form-adiciona">

<div class="grupo">

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

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

</div>

<div class="grupo">

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

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

</div>

<div class="grupo">

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

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

</div>

<div class="grupo">

<label for="gordura">% de Gordura:</label>

<input id="gordura" type="text" placeholder="digite a porcentagem de gordura do seu paciente" class="campo campo-medio">

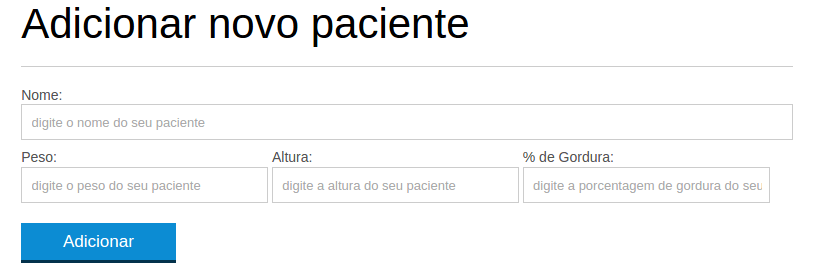
</div>

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

</form>

</section>COPIAR CÓDIGO

O formulário possui os campos de preenchimento dos seguintes dados: "nome", "peso", "altura" e "% de gordura":



Quando a Aparecida preencher esses campos e clicar no botão "Adicionar", uma nova linha na tabela deverá ser criada, com a exibição dos dados recém adicionados através do formulário. Mas para fazê-lo, primeiramente devemos entender algumas funcionalidades do JavaScript.

Como faremos para detectar o exato momento em que o usuário clicará no botão e, então, executar uma ação?

A ação de "perceber o que o usuário está fazendo na página" é o que chamamos de **evento** do browser, que pode ser escutado com o JavaScript. Ações como clicar, duplo clique, *scrollar*, passar o mouse em cima, são exemplos dos tipos de interação que o usuário pode fazer com a página.

Para escutarmos um evento de clique, devemos especificar em qual elemento queremos fazê-lo. Se queremos identificar o momento do clique do botão, usaremos a variável titulo seguida de um "escutador de evento" que escutará as interações dos usuários, addEventListener(). Ela será responsável por adicionar o escutador de evento, como já diz o nome traduzido para o português. Em que tipo de evento estamos interessados? No caso, justamente no evento de click. Queremos que algo aconteça quando o usuário clicar no elemento. Indicaremos o que deve ser feito usando-se a função mostraMensagem(). Dentro dela, adicionaremos console.log("Olá, eu fui clicado!").

titulo.addEventListener("click", mostraMensagem);

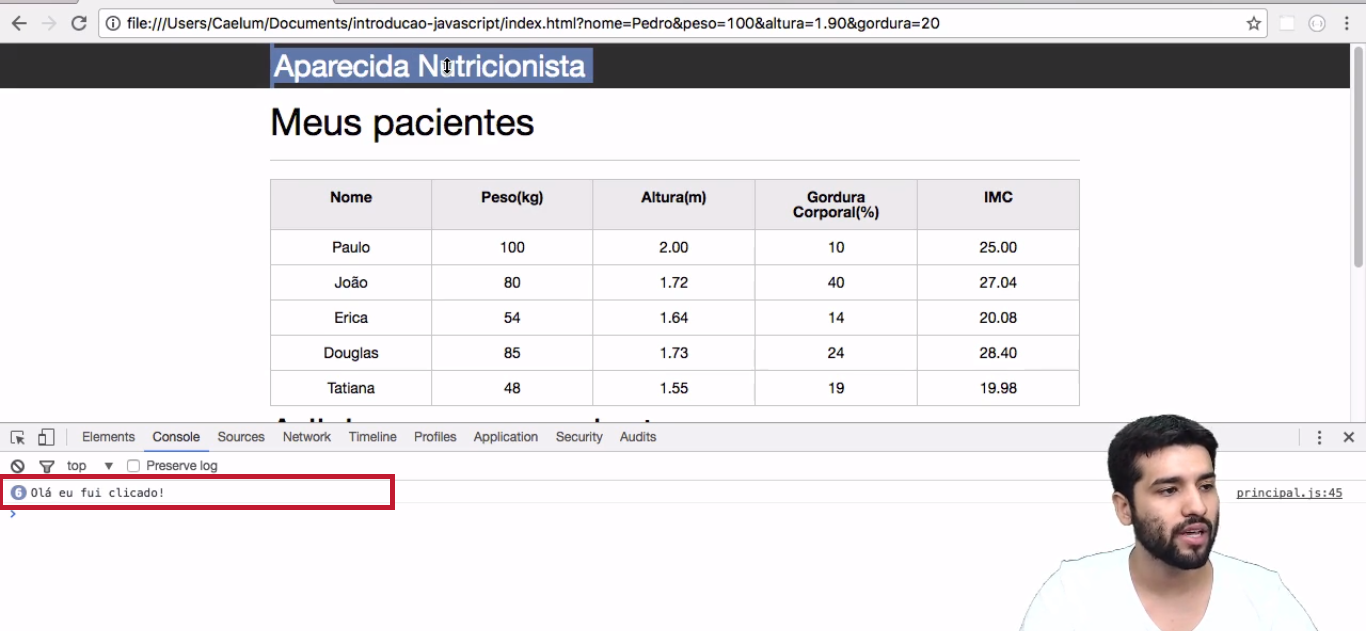
function mostraMensagem(){

console.log("Olá eu fui clicado!");

}COPIAR CÓDIGO

Observe que na variável titulo, chamamos a função mostraMensagem() ao passá-la como segundo parâmetro do addEventListener().

Em seguida, vamos atualizar a página do browser e verificar se as alterações estão funcionando. Clicaremos no título "Aparecida Nutricionista", que é o conteúdo da tag <h1>, e para cada clique a mensagem será exibida no console.



Nós conseguimos adicionar o escutador de eventos no título da página. Quando ele percebe que o usuário interagiu a partir de um clique, ele chama a função mostraMensagem(). Desta forma, atrelamos o evento com um determinado trecho de código HTML.

Uma outra abordagem que poderíamos adotar seria usarmos uma **função anônima**. Em vez de declararmos mostraMensagem como um parâmetro addEventListener(), poderíamos chamar diretamente function():

titulo.addEventListener("click", function (){

console.log("Olha só posso chamar uma função anônima.")

});COPIAR CÓDIGO

Observe que a função ainda possui as chaves ({}), mas não recebe um nome. Funções anônimas são bastante utilizadas, principalmente no caso de eventos. Além disso, chamamos uma função quando interagimos com um evento de clique, que só existirá quando o botão for clicado.

Resumindo, podemos trabalhar tanto com uma função **nomeada**, como a mostraMensagem(), ou uma função **anônima**, que será criada como segundo parâmetro da addEventListener(). Por enquanto, vamos praticar utilizando uma função anônima.

Porém, não queremos adicionar um evento de clique quando clicamos no título, e sim após o clique do usuário no botão "Adicionar" do formulário. A única ação que realizaremos a principio será exibir uma mensagem no console.

Teremos que selecionar o botão no JavaScript! No fim do trecho do código referente ao formulário, encontraremos a tag <button>, acima do fechamento de <form> em index.html:

<!-- ... -->

<div class="grupo">

<label for="gordura">% de Gordura</label>

<input for="gordura" name="gordura" type="text" placehoder="digite a porcentagem de gordura do seu paciente" class="campo campo-medio">

</div>

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

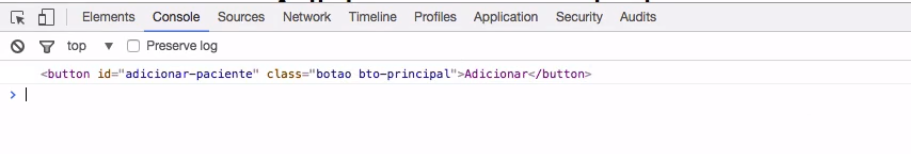
</form>COPIAR CÓDIGO

Em seguida, vamos adicionar o botão no arquivo principal.js. Abaixo do último if de validação dos dados, criaremos a variável botaoAdicionar, e o botão será levado pela função document. Usaremos a função querySelector() para buscar no HTML e, passaremos como parâmetro o seletor #adicionar-paciente:

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

console.log(botaoAdicionar)COPIAR CÓDIGO

Adicionaremos o console.log() para verificarmos se o código está funcionando. No browser, a mensagem será exibida ao atualizarmos a página, indicando que conseguimos pegar o botão como gostaríamos.



Nós também queremos escutar um evento de clique do botão, por isso, adicionaremos novamente addEventListener().

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

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

console.log("Oi, cliquei no botão.");

});COPIAR CÓDIGO

No entanto, se tentarmos clicar no botão "Adicionar" do formulário, a mensagem não será exibida no console. Por que isso acontece?

Por padrão, sempre que clicamos em um botão contido em uma tag <form> do HTML, os seus dados serão enviados para outra página. Como não especificamos uma página para ser o alvo da tag <form>, a única ação realizada é a limpeza dos dados, e a página sendo recarregada em seguida. Ao fazermos isto, além do formulário, o console fica limpo também - por isso, não veremos a mensagem.

O evento de clique está sendo escutado corretamente, porém, como a página é recarregada rapidamente, não conseguiremos ver a mensagem impressa no console. Desta forma, não conseguiremos salvar os dados do paciente na tabela, nem exibir a mensagem.

O que devemos fazer para que a página não seja recarregada e tenha o comportamento esperado?

**02O botão que falhou**

[**PRÓXIMA ATIVIDADE**](https://cursos.alura.com.br/course/javascript-programando-na-linguagem-web/task/23608/next)

Maria está ansiosa para descobrir a razão do seu código não ter funcionado. É um código simples, que exibe um alerta quando um botão é clicado:

<button id="botao">clique-me</button>

<script>

var botao = document.querySelector('#botao');

function botaoHandler() {

alert('Botão clicado');

}

botao.addEventListener('click', botaoHandler());

</script>COPIAR CÓDIGO

Medite sobre o código de Maria e tente descobrir o problema. Assim que tiver chegado a uma conclusão, clique em "Ver Opinião do Instrutor" para ver a resposta do instrutor.

VER OPINIÃO DO INSTRUTOR

**Opinião do instrutor**

O problema está nessa linha:

botao.addEventListener('click', botaoHandler());COPIAR CÓDIGO

Em vez de associar a função botaoHandler para o evento click, ela acabou associando o retorno da função, ao adiciona-la usando parênteses. Como a função não retorna nada, o código que será executado será igual a nada quando o botão for clicado. Para corrigirmos o código, basta **retirarmos os parênteses**:

botao.addEventListener('click', botaoHandler);COPIAR CÓDIGO

Veja que não estamos mais chamando a função, mas passando-a por inteiro para o evento click . Quando o botão for clicado, por debaixo dos panos, o navegador executará botaoHandler() para nós.