Sumário

[Ex1 – News 2](#_Toc525838725)

[Resolução 2](#_Toc525838726)

[Ex2 – Login 6](#_Toc525838727)

[Resolução 6](#_Toc525838728)

[Ex3 – Transação 7](#_Toc525838729)

[Resolução 7](#_Toc525838730)

[Ex4 – Alunos 11](#_Toc525838731)

[Resolução 11](#_Toc525838732)

[Acessando algumas informações da página 11](#_Toc525838733)

[Selecionando um aluno 12](#_Toc525838734)

[E a lógica? 14](#_Toc525838735)

[Organizando nossos arquivos 15](#_Toc525838736)

[Aplicando a média para todos os alunos 16](#_Toc525838737)

[Melhorando a visualização dos alunos 19](#_Toc525838738)

[Escutando eventos 20](#_Toc525838739)

[Como evitar o comportamento padrão de um evento 21](#_Toc525838740)

[Adicionando um novo aluno na lista 21](#_Toc525838741)

[Separando nosso código 23](#_Toc525838742)

[Criando um objeto 24](#_Toc525838743)

[Validando um formulário 26](#_Toc525838744)

[Retornando uma mensagem de erro 28](#_Toc525838745)

# Ex1 – News

## Resolução

1. ES6

<https://medium.com/@matheusml/o-guia-do-es6-tudo-que-voc%C3%AA-precisa-saber-8c287876325f>

1. Referências e valores



* 1. DOM

<https://tableless.com.br/entendendo-o-dom-document-object-model/>

<https://developer.mozilla.org/pt-BR/docs/DOM/Referencia_do_DOM>

* 1. Let e const

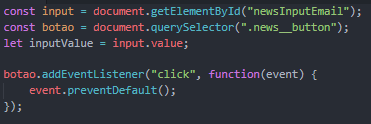
<https://medium.com/skillo-io/quando-usar-const-let-e-var-javascript-es6-3046547db8f3>

**var** — The variable may or may not be reassigned, and the variable may or may not be used outside of its block.

**let** — Is slightly less broad than var. With let, the variable may be reassigned, but the variable will **only be scoped to the block which it is defined.**

**const** — Is the strictest of all the variables. const cannot be reassigned, and is scoped to the block which it is defined. (However, properties can be mutated)

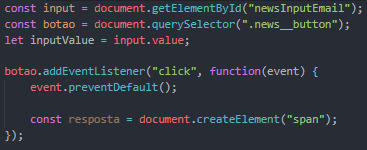
1. Criar uma função ao clicar no botão



* 1. Eventos

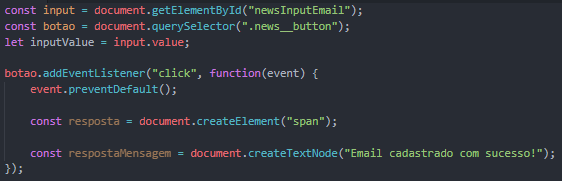
<https://www.devmedia.com.br/trabalhando-com-eventos-em-javascript/28521>

1. Criar um elemento



<span></span>

1. Criar um texto para exibir a mensagem



Email cadastrado com sucesso!

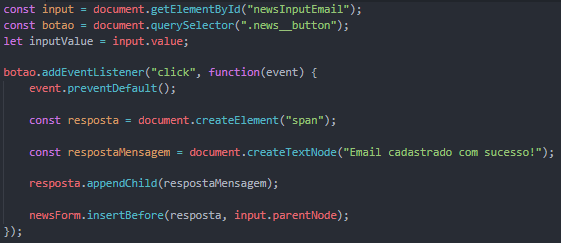
1. Inserir a mensagem dentro da tag span



<span> Email cadastrado com sucesso!</span>

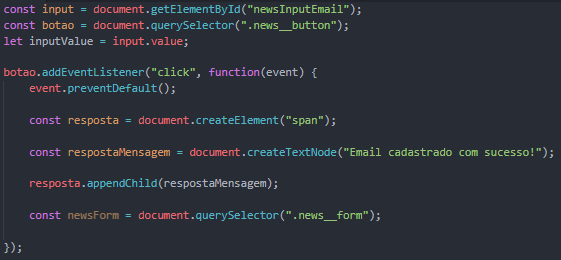


1. Inserir o texto antes de um item estabelecido

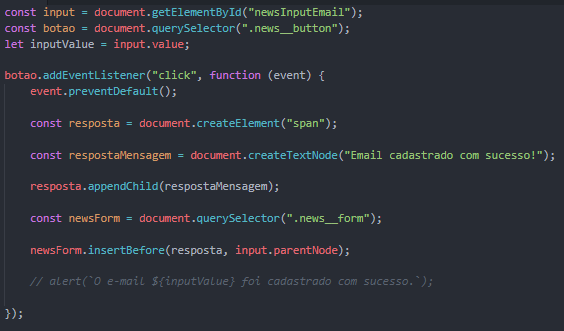


<https://developer.mozilla.org/pt-BR/docs/Web/API/Node/insertBefore>

1. Pegar a referência do formulário



1. Mostrar o e-mail cadastrado



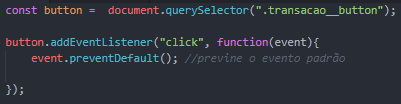
# Ex2 – Login

## Resolução

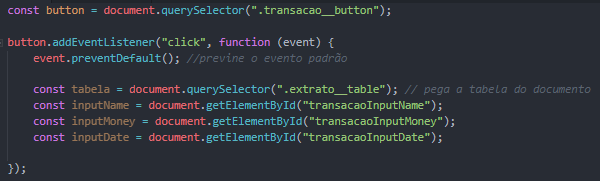
# Ex3 – Transação

## Resolução

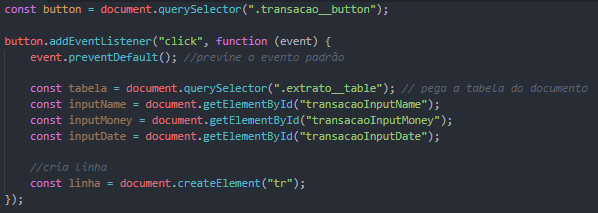
1. Referência do botão



1. Referência da tabela e dos input’s selecionados



1. Cria uma linha



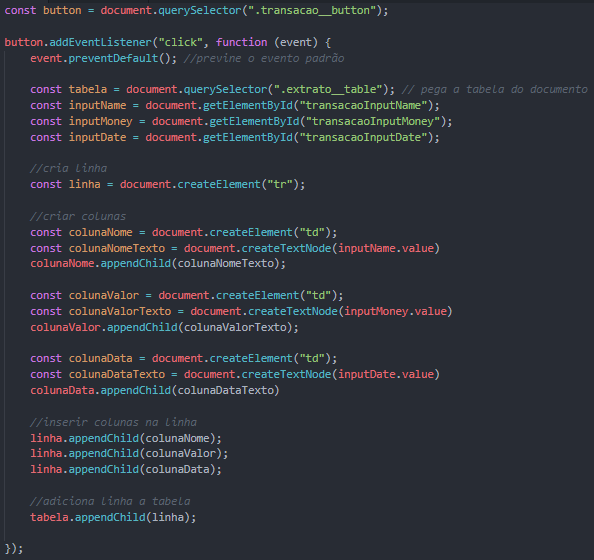
1. Cria as colunas com seus respectivos valores



1. Insere as colunas na linha



1. Adiciona a linha na tabela



# Ex4 – Alunos

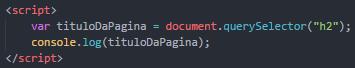
## Resolução

Nesta primeira etapa, vou começar a trabalhar com js em um formulário de cadastro.

O objetivo deste formulário é: calcular a média com base em duas notas e colocar uma cor de fundo para saber se o aluno possui uma média maior do que a exigida pelo curso.

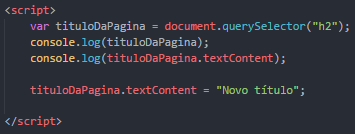
### Acessando algumas informações da página

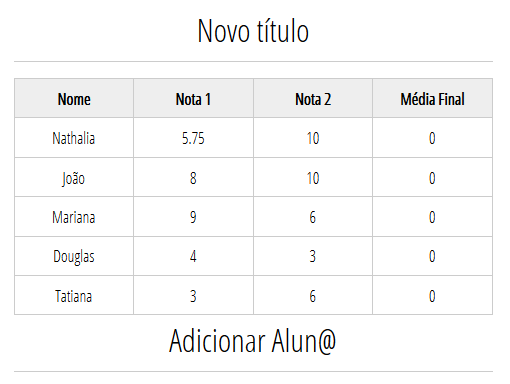
Antes de calcularmos a média, vamos ver como buscamos algumas informações de nossa página.



Nós queremos apenas a informação sobre o conteúdo deste título (“Alun@s”). Assim, podemos alterar o que está escrito.





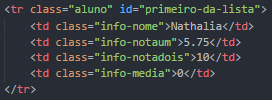


Conseguimos manipular qualquer elemento dentro de uma página html.

### Selecionando um aluno

Uma vez que conseguimos manipular os elementos dentro de uma página HTML, o próximo passo, sendo objetivo, é que queremos buscar as notas do nosso primeiro aluno da lista e calcular a média.

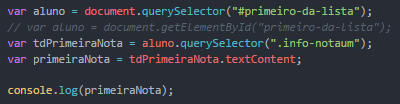
Porém, observamos que os dados de acesso para os alunos, são os mesmos. Vamos realizar a seguinte tarefa. Vamos colocar um identificador para este aluno e realizarmos os cálculos que desejamos.



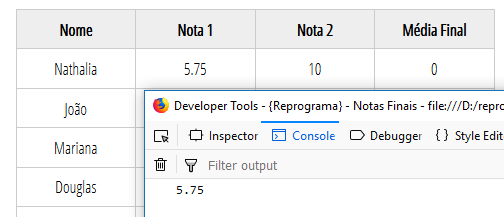
Existem algumas maneiras de eu buscar um determinado item (por id ou por classe).



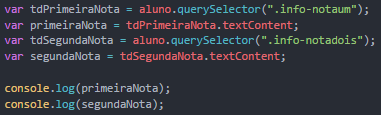
Posso utilizar o querySelector ou até mesmo o getElementById, ou até por classe ou tag.

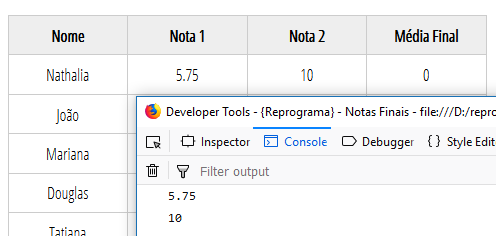


Dessa maneira, conseguimos buscar as notas do aluno que desejamos e realizar o cálculo. Assim como fizemos com a primeira nota, vamos realizar com a segunda nota.

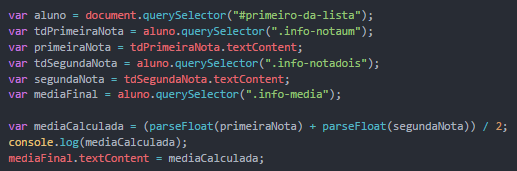


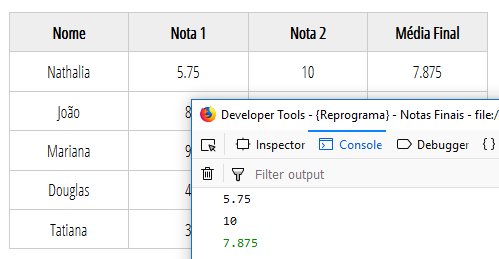
Buscando os dois valores das notas que foram inseridas.





Uma vez que temos os dados das nossas notas, vamos apenas calcular sua média.



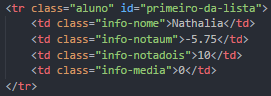


### E a lógica?

Vimos até agora, como calcular a média dada duas notas. Mas aonde entra a lógica? Imagina que, por algum motivo, o professor tenha entrado com uma inválida ou até mesmo negativa (inválida). Ainda assim, a média seria calculada? Além disso, uma nota poderia ser maior do que 10 (dez)?

Vamos fazer uma verificação simples: caso a nota seja menor do que zero ou superior a 10, a média final deverá ser mostrada como inválida. Como faríamos?

Vamos alterar o valor da primeira nota para um valor negativo.

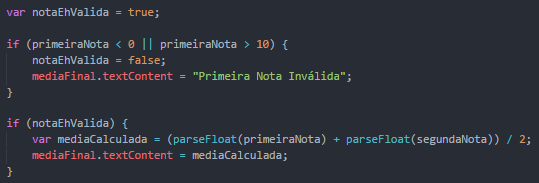


Após alterarmos o valor, vamos realizar a nossa lógica.

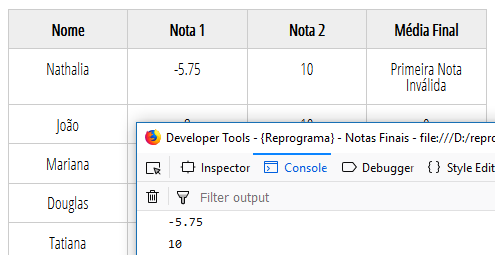
Primeiro vamos declarar uma variável para saber se o valor inserido está de acordo com os critérios.



Após essa verificação, vamos verificar os valores inseridos até então.



Caso o valor da primeira nota entre em uma das duas primeiras decisões, o valor será considerado inválido.

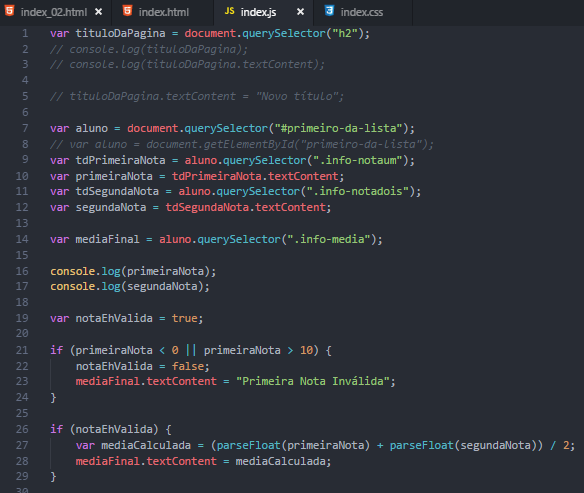


### Organizando nossos arquivos

Vamos começar a organizar os nossos arquivos do nosso projeto. Vamos criar um novo arquivo chamado index.js (por enquanto) dentro da pasta js.



Além disso, vamos colocar o código que escrevemos até agora, dentro do index.js.

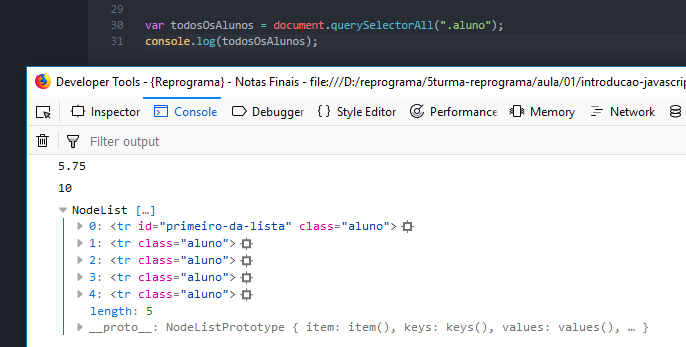




### Aplicando a média para todos os alunos

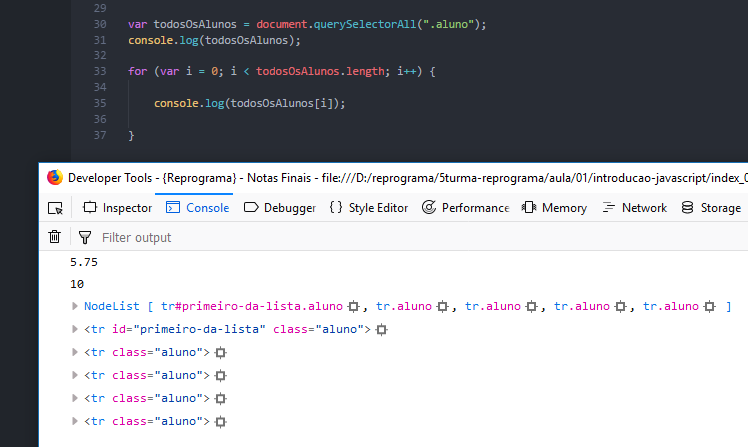
Uma vez que calculamos a média apenas para o primeiro aluno da lista, o que precisamos fazer, é calcular a média para todos os alunos da nossa lista.

O querySelector, ele busca apenas uma referência deste item. Já o querySelectorAll, nos trará toda a listagem de alunos.



Já temos a nossa lista de alunos. Falta apenas percorrermos essa lista de maneira que: para cada aluno dessa lista, eu consiga realizar o cálculo da média.

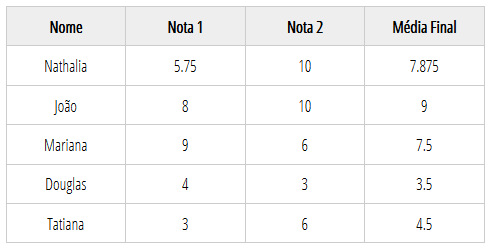
Vamos aos passos:



Uma vez que eu consigo percorrer a minha lista de alunos inteira, basta eu copiar o código que nós já fizemos e colocarmos dentro dessa condição. Ou seja, quando a lista for percorrida, ele fará para item da minha lista, sendo assim, para cada que for incluído.

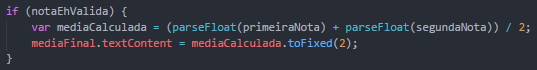
A única diferença será que ao invés de selecionarmos o primeiro aluno pelo id, nós selecionaremos pelo número do índice.

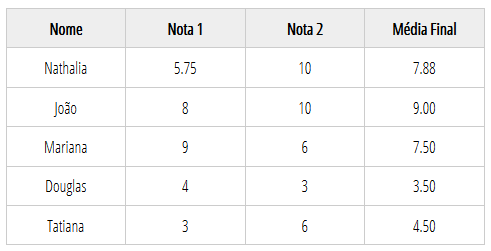




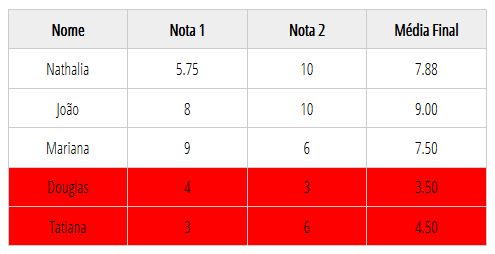
<https://www.w3schools.com/js/tryit.asp?filename=tryjs_number_tofixed>

Apenas para formatar os números que estamos mostrando em nossa lista, vamos utilizar o toFixed.



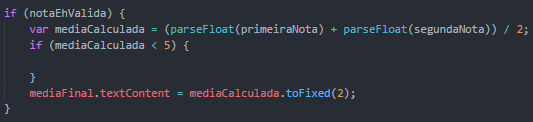


### Melhorando a visualização dos alunos



Nós já vimos que sempre que quisermos manipular os elementos dentro de uma árvore de elementos html, nós temos recursos e uma api que faz isso pra nós. Uma vez que queremos alterar o estilo de um determinado item da nossa lista, o que podemos fazer?

Antes de alterarmos o valor correspondente, precisamos realizar uma outra condição simples. Só ficará vermelho caso a média do aluno seja inferior a 5.



Caso contrário, a cor permanecerá a mesma.

Google:

Javascript change css style

<https://www.w3schools.com/js/js_htmldom_css.asp>

Nós podemos realizar inicialmente de duas maneiras:

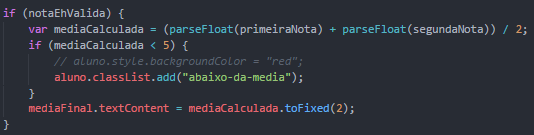


Nós podemos adicionar diretamente o estilo em nossa lista para cada item. Porém, imagina que, nós vamos manipular mais de um item por vez, nós teríamos que adicionar muitas propriedades.

Nós podemos criar um estilo em nosso arquivo css, por exemplo:



Agora, ao invés de adicionarmos diretamente o background em cada linha, adicionamos o estilo como um todo. E caso eu queira adicionar novas propriedades nesse estilo, sua manipulação ficará mais fácil.

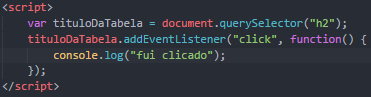


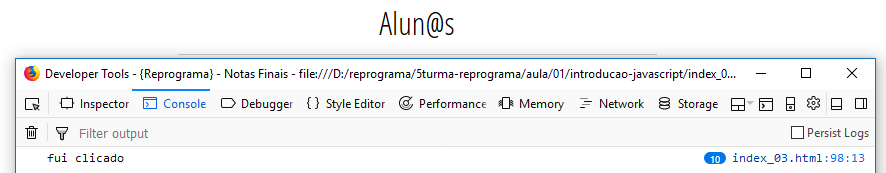
### Escutando eventos

Por quê será que é necessário eu precisar “escutar” um evento da minha página? A ideia é simples. Quando eu clicar no botão para adicionar um novo aluno, eu desejo que alguma ação se realize. Neste caso, eu vou adicionar um novo aluno em nossa tabela.

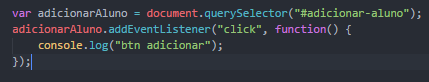
Mas antes disso, vamos ao básico.

Quando eu quero que algo ocorra quando eu clico, por exemplo, no título da página, eu trabalho com o ‘click’.





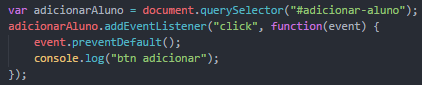
Fizemos o teste no nosso próprio index.html. E tudo funciona perfeitamente. Vamos então ao que de fato, nós precisamos fazer.



Antes de continuarmos, testem o que ocorre quando eu preencho os dados do formulário e clico no botão ‘Adicionar’.

### Como evitar o comportamento padrão de um evento

event.preventDefault();

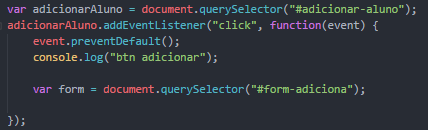


Testem novamente o que ocorre agora quando o botão é clicado.

### Adicionando um novo aluno na lista

<https://developer.mozilla.org/pt-BR/docs/Traversing_an_HTML_table_with_JavaScript_and_DOM_Interfaces>

Através do formulário, nós temos acesso aos seus *inputs.* Através da propriedade *name* que colocamos em nosso HTML, nós podemos chama-los diretamente.



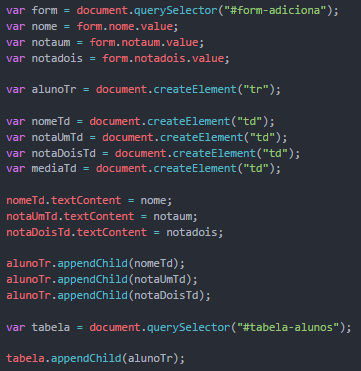
Como nós queremos o valor de entrada do campo input:

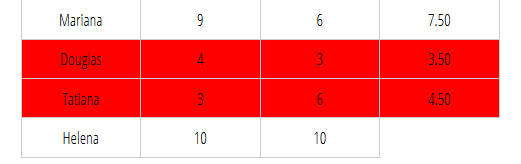


Vamos buscar os 3 dados que desejamos de entrada.



Uma vez que pegamos os dados de entrada do usuário, nós precisamos criar um novo elemento HTML com JavaScript, ou seja, pegar os dados do usuário e inserir mais uma linha em nossa tabela. Como faremos isso?

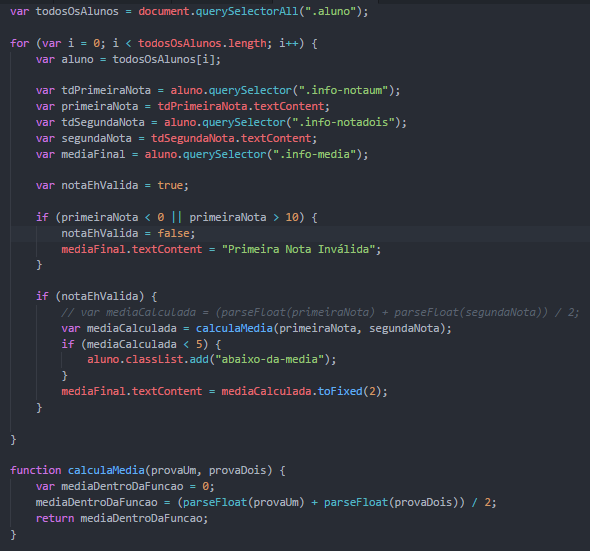




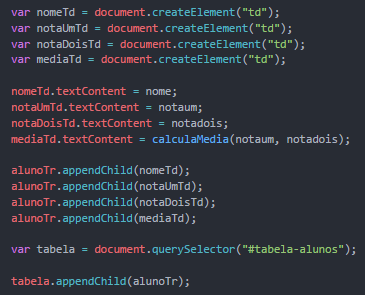
### Separando nosso código

Um dos problemas que nós encaramos agora é que ao adicionarmos um novo aluno, nós precisamos calcular sua média. Seria interessante eu repetir o código no formulário e na lista uma vez que a função que ele irá executar será a mesma?

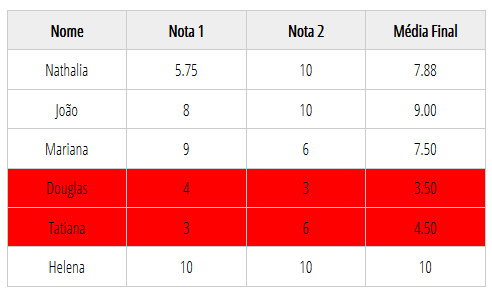
Sendo assim, vamos criar uma função que faça isso pra nós. O que nós precisamos saber sobre a média para que a função calculaMedia() seja executada?



Aqui, estamos trabalhando com a nossa lista. Uma vez que desejamos realizar o mesmo cálculo ao inserir um novo aluno, essa função também pode ser chamada.



Sendo assim, ao inserir um novo aluno, eu consigo calcular a sua devida média.



### Criando um objeto

Vamos passar os dados que recebermos do formulário para uma função e, além disso, vamos criar um objeto do tipo aluno.

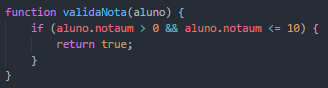




### Validando um formulário

Uma vez que conseguimos buscar os dados de nosso formulário, não seria interessante, antes de colocarmos estes dados em uma lista, validarmos os dados de entrada do usuário e inclusive mostrar uma mensagem de erro para o mesmo?

Faremos isso nos passos a seguir.



Vamos criar uma função que valida os dados de entrada do usuário.

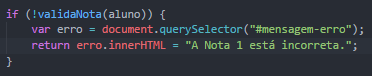


Sempre que eu precisar realizar uma chamada, eu chamo a própria função para realizar o cálculo. Sendo assim, ele só vai inserir um novo item na lista, caso os dados sejam digitados de maneira correta.

### Retornando uma mensagem de erro

Vamos adicionar uma mensagem de erro em nosso html.





<https://stackoverflow.com/questions/21311299/nodevalue-vs-innerhtml-and-textcontent-how-to-choose>