

Tutorial para Introdução à Programação em JavaScript

Professor Claudio Junior

O objetivo desse tutorial é proporcionar, de modo dirigido, o entendimento do uso da Linguagem JavaScript para interagir com o HTML incorporando dinamismo à página.

O exercício dirigido disponibiliza arquivos HTML e CSS, o quais tem por objetivo resolver um fictício problema de registro de notas de alunos. Trata-se de um cadastro básico, contendo quatro inputs (Matrícula, Nome, Nota e N° de Faltas) e que, após a confirmação, adicionará o registro em uma tabela montando uma lista. O atributo situação é obtido a partir da nota e do número de faltas. Caso a nota seja menor do que sete ou o número de faltas maior do que seis, o aluno será classificado como reprovado. Caso contrário, será aprovado, como pode ser observado na Figura 1.

Cadastro de Notas e Faltas dos Alunos

Matrícula:

Nome:

Nota:

Número de Faltas:

Resultado

Matrícula	Nome	Nota	Faltas	Situação
202311001	Aldebaran Estrela	8	5	AA
202311002	John Keynes	10	0	AA
292311005	Saturnino Alves	5	9	RF

Figura 1 – Programa em execução

No HTML há um botão (confirmar) que ao ser clicado irá desencadear alguma ação. Por se tratar de um objeto, este botão possui métodos que podem ser executados e propriedades que podem ser alteradas, modificadas ou manipuladas. Nós iremos atribuir este botão a uma variável. Antes, porém, precisamos saber qual o seu nome, ou melhor, como podemos identificá-lo. Existem três formas de identificar um objeto no HTML:

1. Verificando diretamente no arquivo HTML

Observem que na linha 14 (Figura 2) há uma referência a um objeto do tipo botão:

```
<button type="button" id="confirm-button">Confirmar</button>
```

Assim, obtemos a identificação do botão que iniciará o processo de inclusão dos dados.

```
1 <form id="student-form">
2   <label for="matricula">Matrícula:</label>
3   <input type="text" id="matricula" required><br>
4
5   <label for="nome">Nome:</label>
6   <input type="text" id="nome" required><br>
7
8   <label for="nota">Nota:</label>
9   <input type="number" id="nota" required><br>
10
11  <label for="faltas">Número de Faltas:</label>
12  <input type="number" id="faltas" required><br>
13
14  <button type="button" id="confirm-button">Confirmar</button>
15 </form>
```

Figura 2 – Código parcial HTML

2. A segunda forma é quando executamos o arquivo HTML e abrimos o navegador podemos, seja por meio das Ferramentas do Desenvolvedor ou pressionando teclas de atalho **Ctrl + Shift + C** ou **Ctrl + Shift + I**, como observado na Figura 3:
 1. Será possível observar no painel o menu com as opções **Elements**, Console, Sources e **Network**;
 2. Em Elements será possível navegar por todos os elementos que compõem a página HTML;
 3. Ao colocar o mouse sobre uma TAG específica, automaticamente ela é destacada na tela de execução do HTML;
 4. É possível observar o botão Confirmar em destaque tanto no form, quanto em **Elements** e a sua hierarquia: *html* → *body* → *div-form-container* → *form#student-form*;
 5. Observamos que o botão confirmar é identificado como **id="confirm-button"**

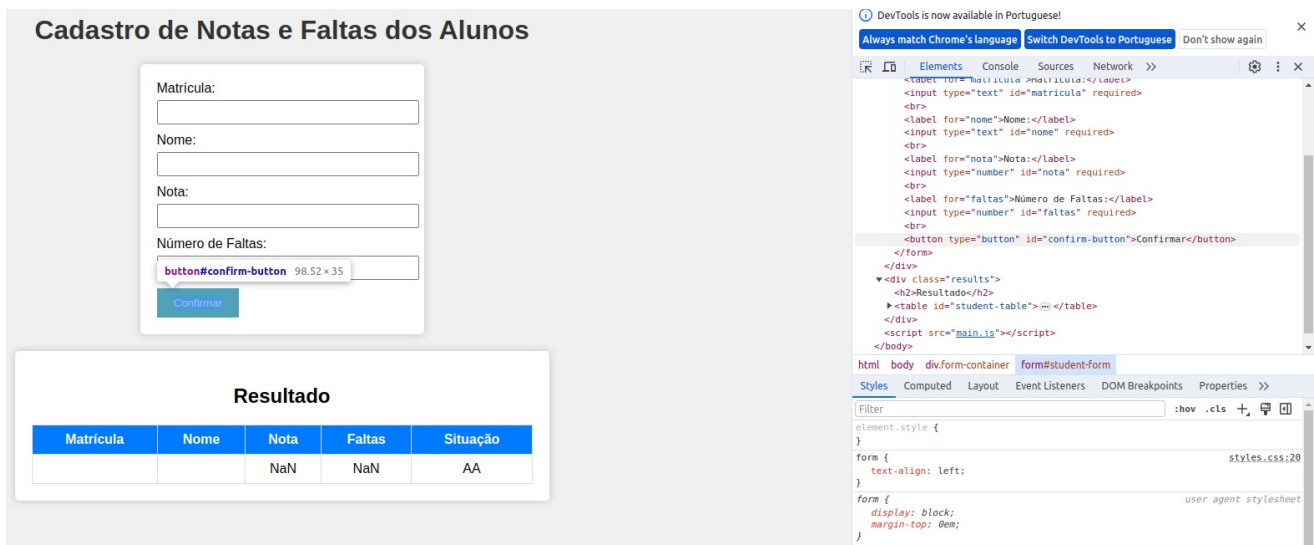


Figura 3 – Navegador e Inspeção

3. A terceira forma é também utilizando as Ferramentas do Desenvolvedor ou pressionando teclas de atalho **Ctrl + Shift + C** ou **Ctrl + Shift + I**, porém usando o **Console** (Figuras 4 e 5):
 1. Sabemos que se trata de um botão podemos, por exemplo, pesquisar todos os botões da página utilizando **document.querySelectorAll(\"button\")**;



Figura 4 – Uso do console

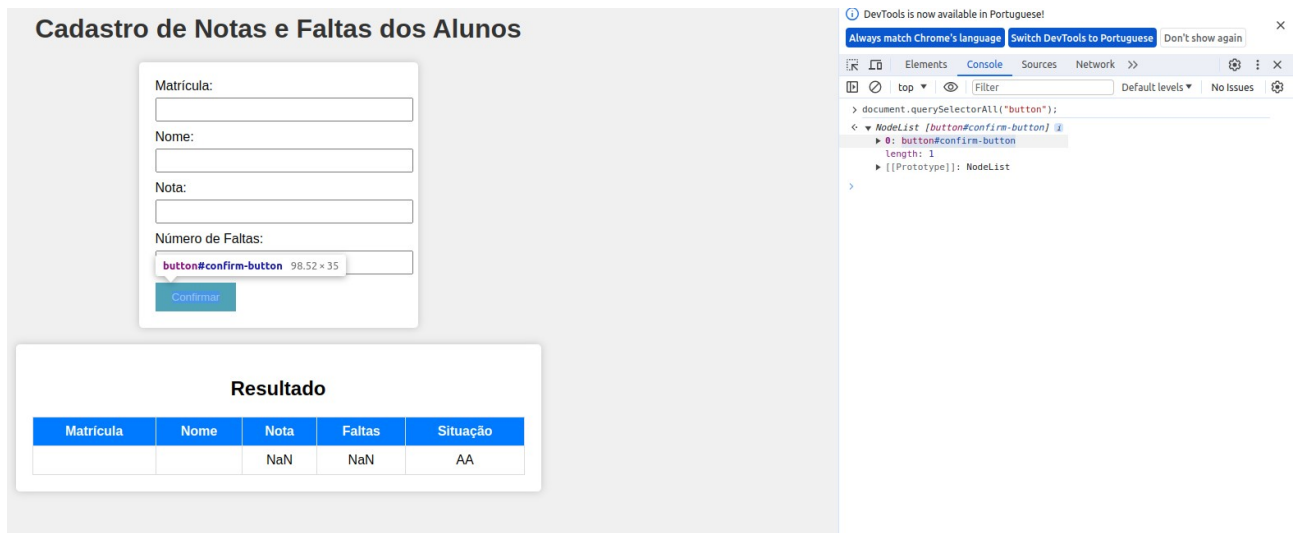


Figura 5 – Uso do console

Agora que já sabemos o nome/identificação do botão de confirmação (`id="confirm-button"`), podemos declarar uma variável e assim poderemos manipular esse elemento com mais facilidade sem precisar utilizar toda a sua declaração. A principal vantagem de usar a declaração da variável é que caso o elemento receba outra denominação, só será necessário atualizar em um único ponto.

Assim, a variável será declarada como:

```
const buttonConfirm = document.getElementById("confirm-button");
```

Neste momento, essa será a primeira linha do nosso arquivo `<main.js>`;

E agora que já identificamos o elemento (botão de confirmação) e já declaramos uma variável. O que iremos fazer? Qual o propósito desta definição?

Agora iremos utilizar o conceito de eventos, ou seja, dado determinado evento com esse elemento (botão de confirmação), iremos “armazenar os dados em um determinado local”;

Como poderemos definir este evento?

Lembre-se que a página fica **aguardando (ou escutando)** alguma ação (*movimento do mouse, duplo click, click, enter...*) para poder executar algo. Neste caso, vamos considerar que o armazenamento, ou adição, dos dados do aluno se dará após clicarmos uma vez no botão confirmar. Utilizaremos o evento Listener para que, quando seja clicado no botão venhamos executar a adição dos dados do aluno.

Utilizaremos a variável que declaramos (`buttonConfirm`) que, por se tratar de um objeto, possui métodos e propriedades. No caso vamos usar **`addEventListener`**, o qual ficará aguardando algum que algum evento ocorra com este botão e, ao ocorrer, desencadeará a execução de algo.

Como já foi dito, JS faz uso de funções para executar sua programação. Podemos utilizar o formato de função anônima ou de nos referirmos, de fato, a uma função específica, seja própria ou de terceiros. Dessa forma, assim poderia ficar chamada do evento click ao botão:

```
buttonConfirm.addEventListener("click", function() {  
    // código da função  
});
```

ou:

```
buttonConfirm.addEventListener("click", addMatricula);
```

Nossa escolha será definir uma função (**addMatricula**) e a partir dela desencadear todo o processo de adição/armazenamento dos dados. Isso porque, caso seja necessário, poderemos executar/"chamar" essa função de outros pontos do programa caso seja necessário;

```
function addMatricula(){  
    // código da função  
};
```

O que a função addMatricula vai fazer?

A função addMatricula será responsável por receber os dados existentes no formulário HTML (Matricula, Nome, Nota e Numero de Faltas) e que caracterizam o Aluno, Figura 6, e armazená-los em uma tabela. Em seguida, deve limpar os campos que foram digitados e aguardar novos dados.

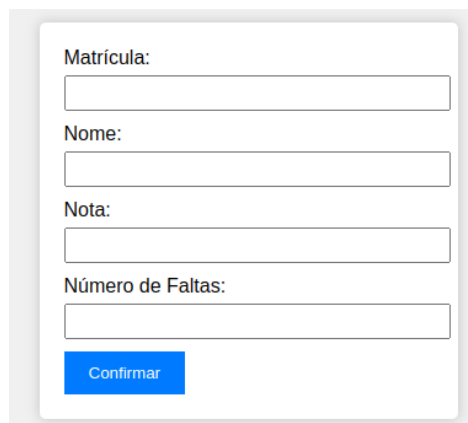
A imagem mostra um formulário web com um fundo cinza claro. O formulário é um retângulo branco com uma borda sutil. Ele contém quatro campos de entrada empilhados verticalmente. Cada campo é precedido por um rótulo: 'Matricula:', 'Nome:', 'Nota:' e 'Número de Faltas:'. Os campos de entrada são retângulos brancos com bordas cinzas. O campo de 'Nota' parece ser um campo de número. Abaixo dos campos, há um botão azul com o texto 'Confirmar' em branco.

Figura 6 – Uso do console

Como iremos obter esses dados do formulário HTML, ou seja, o que precisa ser feito para que nosso programa JS utilize os dados que foram digitados na página?

Poderemos verificar diretamente no HTML onde esses dados estão ou então, novamente o DOM nos ajudará na identificação de cada um deles. Verificando os campos dos dados diretamente no HTML, Figura 7:

```
1 <form id="student-form">  
2   <label for="matricula">Matricula:</label>  
3   <input type="text" id="matricula" required><br>  
4  
5   <label for="nome">Nome:</label>  
6   <input type="text" id="nome" required><br>  
7  
8   <label for="nota">Nota:</label>  
9   <input type="number" id="nota" required><br>  
10  
11  <label for="faltas">Número de Faltas:</label>  
12  <input type="number" id="faltas" required><br>  
13  
14  <button type="button" id="confirm-button">Confirmar</button>  
15 </form>
```

Figura 7 – Código parcial do HTML

Ou então Inspeccionando a página (Figura 8):

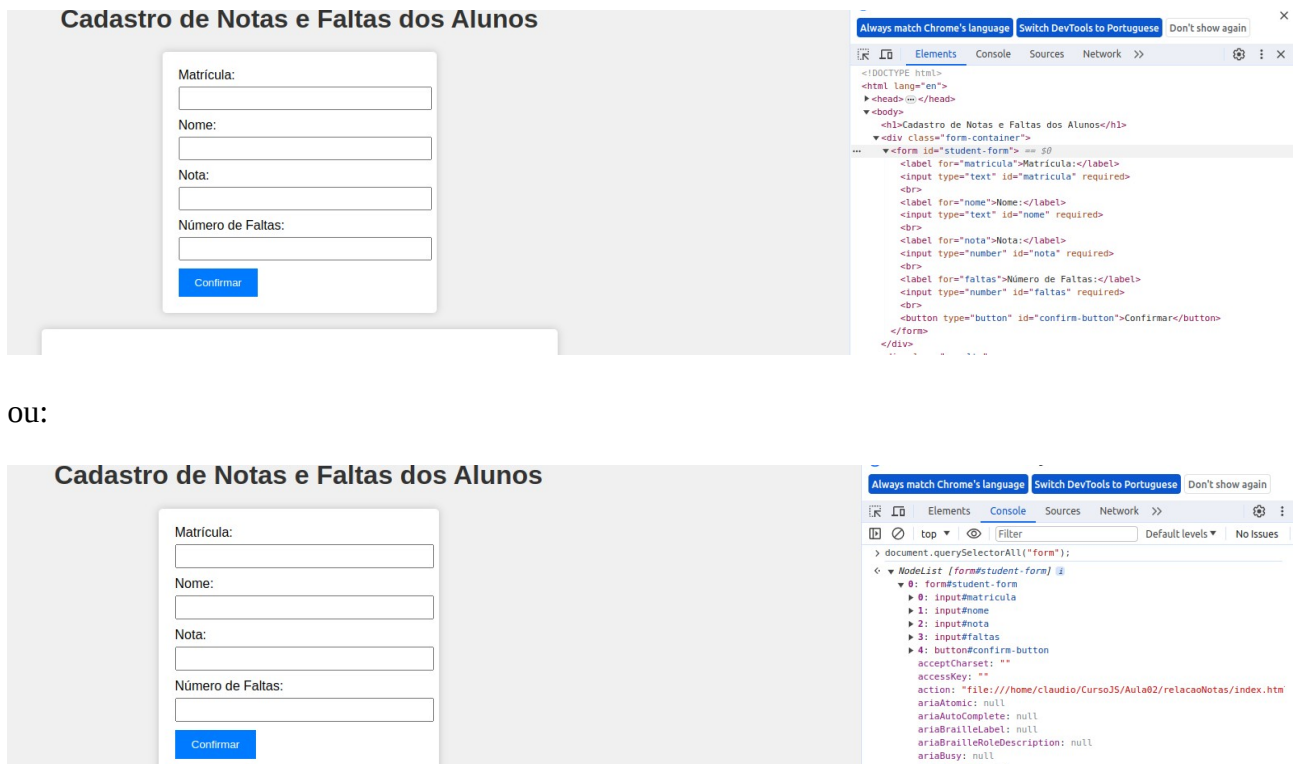


Figura 8 – Análise do navegador e da Inspeção (Elements e Console)

Percebemos que os “campos” que iremos trabalhar são elementos e estão identificados com seu id:

- #matricula;
- #nome;
- #nota;
- #falta.

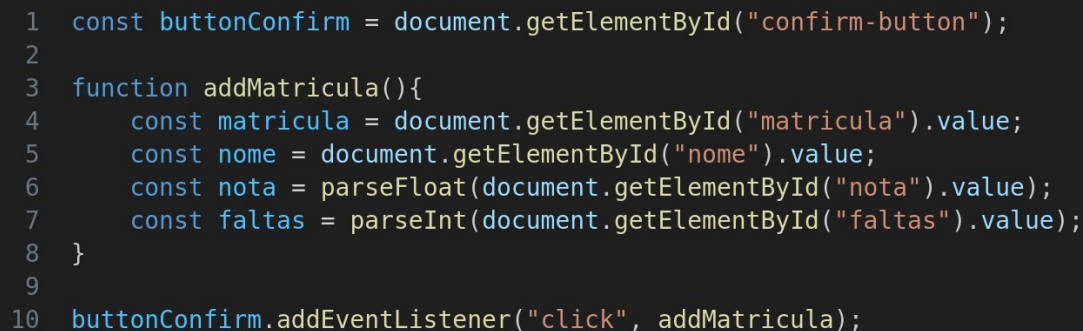
Podemos considerar que a matrícula pode ser um código alfanumérico ou apenas numérico. Vamos considerar que seja alfanumérico, ou seja, aceitando letras e números. Nome também será do tipo alfanumérico. Para estes dois casos, consideramos o tipo Texto como está no formulário HTML.

No caso da nota, ela é do tipo Float, ou seja, aceita pontos decimais. E no caso de faltas, é do tipo Inteiro. O tipo de cada um é identificado no HTML.

Dito isso, vamos fazer com que nosso programa armazene esses dados em variáveis para que possamos utilizá-las ao longo do processamento.

Se sabemos que cada um dos dados é um elemento do HTML, iremos obter esses dados utilizando o `document.getElementById("id elemento")`

Assim, começaremos a escrever a nossa função **addMatricula**, Figura 9, declarando primeiro as variáveis que iremos utilizar:



```
1  const buttonConfirm = document.getElementById("confirm-button");
2
3  function addMatricula(){
4      const matricula = document.getElementById("matricula").value;
5      const nome = document.getElementById("nome").value;
6      const nota = parseFloat(document.getElementById("nota").value);
7      const faltas = parseInt(document.getElementById("faltas").value);
8  }
9
10 buttonConfirm.addEventListener("click", addMatricula);
```

Figura 9 – Função addMatricula

Neste momento temos a declaração da variável do objeto botão confirmar, o evento click e a nossa função addMatricula com as variáveis declaradas e que serão utilizadas ao longo do processamento do programa.

As variáveis matricula e nome são do tipo texto. O DOM faz uma cópia do HTML para poder manipulá-lo. Ao utilizarmos `document.getElementById("matricula").value` e `document.getElementById("nome").value`, estamos copiando, para memória, os valores que estão digitados no formulário HTML. E assim poderemos manipulá-los, não o HTML em si, mas a cópia que o DOM fez ao carregar a página.

No caso de nota, usamos o `parseFloat` que converte o texto digitado em um número do tipo Float. Já em faltas, utilizamos o `parseInt` que converte o texto digitado no formulário em uma variável do tipo Inteiro.

Agora que já temos as variáveis, precisamos verificar se o aluno foi aprovado, reprovado por faltas ou por nota. Ou seja, a situação do aluno. Neste caso vamos declarar uma variável para armazenar a situação do aluno, a qual será atualizada dependendo da nota e do número de faltas.

O aluno será reprovado caso sua nota seja menor do que 7 ou se o número de faltas for maior do que 6. Caso contrário ele será considerado aprovado. Percebam que aí teremos uma estrutura de condição.

A função **addMatricula**, Figura 10, agora será composta por:

```

1  const buttonConfirm = document.getElementById("confirm-button");
2
3  function addMatricula(){
4      const matricula = document.getElementById("matricula").value;
5      const nome = document.getElementById("nome").value;
6      const nota = parseFloat(document.getElementById("nota").value);
7      const faltas = parseInt(document.getElementById("faltas").value);
8      let situacao = "AA";
9
10     if (nota < 7) {
11         situacao = "RA";
12     }
13
14     if (faltas > 6) {
15         situacao = "RF";
16     }
17 }
18 buttonConfirm.addEventListener("click", addMatricula);

```

Figura 10 – Função addMatricula

Declaramos a variável **situacao** como **let**, pois ela poderá ter seu valor alterado, e a inicializamos com “AA”, ou seja, aprovado. Em seguida são feitos testes/condições para alterar o seu valor. Caso a nota seja menor do que 7, será reprovado “RA” ou caso o número de faltas seja maior do 6, será reprovado por faltas “RF”.

Neste momento, apesar de já termos criado a função **addMatricula**, nós só inicializamos as variáveis e identificamos a situação do aluno. O próximo passo é fazer com que os dados confirmados sejam apresentados na tela em forma de tabela, onde identificamos como Resultado (Figura 11):

Cadastro de Notas e Faltas dos Alunos

Matricula:

Nome:

Nota:

Número de Faltas:

Confirmar

Resultado

Matricula	Nome	Nota	Faltas	Situação
202311001	Aldebaran Estrela	8	5	AA
202311002	John Keynes	10	0	AA
292311005	Saturnino Alves	5	9	RF

Figura 11 – Dados preenchidos

Se atentarmos para o nosso HTML, Figura 12, poderemos observar que ele possui um trecho de código que inicia com uma div class chamada results e que possui a montagem de uma tabela, cujo cabeçalho conforme Figura acima. As TAGs <table></table> criam a tabela. As TAGs <tr></tr> são responsáveis por criar as linhas da tabela. E as TAGs <th></th> identificam o cabeçalho da tabela.

```
1 <div class="results">
2   <h2>Resultado</h2>
3   <table id="student-table">
4     <tr>
5       <th id="matricula-header">Matrícula </th>
6       <th id="nome-header">Nome </th>
7       <th>Nota</th>
8       <th>Faltas</th>
9       <th>Situação</th>
10    </tr>
11  </table>
12 </div>
```

Figura 12 – Trecho do HTML

O mesmo pode ser observado ao abrirmos a opção de Inspeção do navegador (Figura 13).

The screenshot shows a web application titled "Cadastro de Notas e Faltas dos Alunos". It features a form with input fields for "Matrícula:", "Nome:", "Nota:", and "Número de Faltas:", along with a "Confirmar" button. Below the form is a section titled "Resultado" containing a table with the following structure:

Matrícula	Nome	Nota	Faltas	Situação
-----------	------	------	--------	----------

On the right side, the Chrome DevTools "Elements" panel is open, showing the DOM tree. It highlights the table element with the id "student-table", displaying its HTML structure, which matches the code shown in Figure 12.

Figura 13 – Navegador e Inspeção

A ideia é que a função **addMatricula** tenha um código que faça a inclusão dos dados nesta Tabela e assim os dados serão apresentados. Para tanto, e por motivos triviais, vamos criar uma nova função chamada **addRow**, passando como parâmetros os dados (matricula, nome, nota, faltas e situação) para criar uma nova linha na Tabela. Essa função será executada em sequência e após a atualização do conteúdo da variável *situacao* (Figura 14).

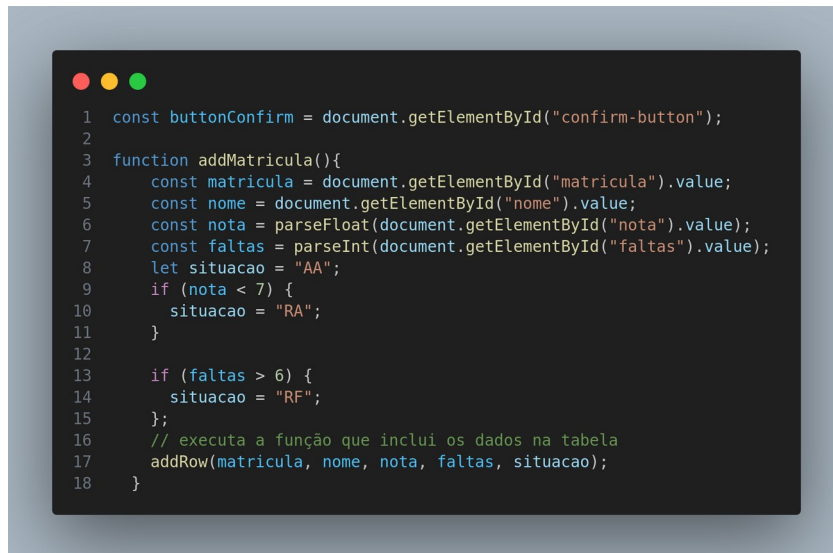


Figura 14 – Função addMatricula

Nosso objetivo é incluir uma linha na Tabela e preencher as colunas com os dados passados como parâmetros, ou seja, matrícula, nome, nota, nº de faltas e situação.

Como podemos identificar a tabela em nosso arquivo HTML?

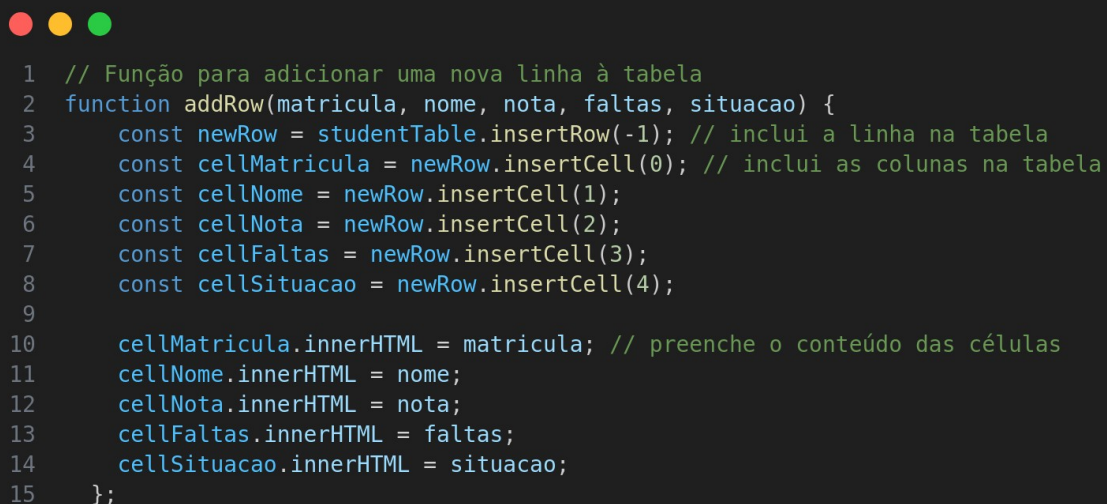
Verificamos, tanto no HTML quando na Inspeção dos Elementos, que a Tabela que conterá os dados foi criada (TAGs `<table></table>`) com a identificação `<table id="student-table">`. Vamos criar nossa **função addRow** e então, obter/carregar, esse elemento para uma variável e assim poder manipulá-lo (*lembre-se que o DOM faz uma cópia do HTML*):

```
const studentTable = document.getElementById("student-table");
```

Como toda tabela é composta de linhas e colunas, precisamos inserir uma linha e N colunas. O índice de cada coluna começa sempre com 0 e a última coluna possui o índice N-1.

Para inserir a linha, utilizaremos `const newRow = studentTable.insertRow(-1);` e para inserir as colunas utilizaremos a sintaxe `const cellnomeVar = newRow.insertCell(0);`. Em seguida, alteraremos o HTML em tempo de execução também, para que os conteúdos das células sejam alterados.

A função **addRow** (Figura 15) estará assim constituída:



```

1 // Função para adicionar uma nova linha à tabela
2 function addRow(matricula, nome, nota, faltas, situacao) {
3     const newRow = studentTable.insertRow(-1); // inclui a linha na tabela
4     const cellMatricula = newRow.insertCell(0); // inclui as colunas na tabela
5     const cellNome = newRow.insertCell(1);
6     const cellNota = newRow.insertCell(2);
7     const cellFaltas = newRow.insertCell(3);
8     const cellSituacao = newRow.insertCell(4);
9
10    cellMatricula.innerHTML = matricula; // preenche o conteúdo das células
11    cellNome.innerHTML = nome;
12    cellNota.innerHTML = nota;
13    cellFaltas.innerHTML = faltas;
14    cellSituacao.innerHTML = situacao;
15 };

```

Figura 15 – Função addRow

Para finalizar, nossa função **addMatricula** deverá limpar os campos do formulário para que novos dados possam ser digitados.



```

1 function addMatricula(){
2     const matricula = document.getElementById("matricula").value;
3     const nome = document.getElementById("nome").value;
4     const nota = parseFloat(document.getElementById("nota").value);
5     const faltas = parseInt(document.getElementById("faltas").value);
6     let situacao = "AA";
7     if (nota < 7) {
8         situacao = "RA";
9     }
10
11     if (faltas > 6) {
12         situacao = "RF";
13     };
14     // executa a função que inclui os dados na tabela
15     addRow(matricula, nome, nota, faltas, situacao);
16
17     // Limpar o formulário após a submissão
18     document.getElementById("matricula").value = "";
19     document.getElementById("nome").value = "";
20     document.getElementById("nota").value = "";
21     document.getElementById("faltas").value = "";
22 };

```

Figura 16 – Função addMatricula