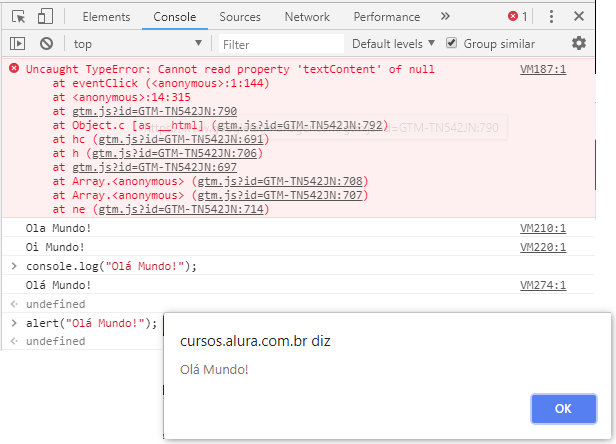
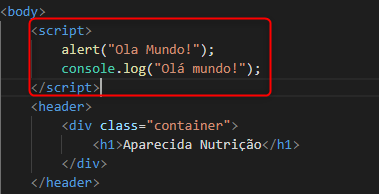
**JAVASCRIPT**

# **INTRODUÇÃO**

* Linguagem **NATIVA** da Web: Utilizado para dar dinamismo às páginas, permitindo a interação com **dados** do servidor, **validações**, **animações** e **controle** de **ações** da página.
* **Node JS**: Back-end com javascript; **Electron**: Aplicações Desktop com HTML, CSS e Javascript; **Johnny-Five:** Programação de um arduíno; **MongoDB:** Queries com sintaxe do JS.

No navegador, em **Ferramentas do Desenvolvedor >> Console**: É possível executar funções Javascript.



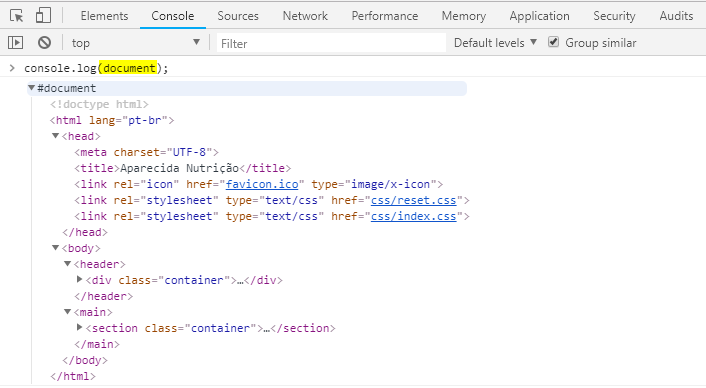


**ERRO DE SINTAXE: Uncaught SyntaxError: Unexpected identifier**

**1.2 Query Selector**

**DOM (*Document Object Model*): Representação** da página/código HTML para NAVEGADOR. Código que o navegador utiliza como representação (HTML em memória), a partir do *response* do arquivo HTML*,* para o navegador montar/**renderizar** a página **SEQUENCIALMENTE**.

**document: Representação do DOM:** Ponte entre o **HTML** e o **JS**, permitindo **manipular** o DOM.



**document.querySelector(“ELEMENTO”):** Busca de elementos dentro do DOM.

**var** textoDoElemento = elemento.**textContent**: Conteúdo dentro da Tag.

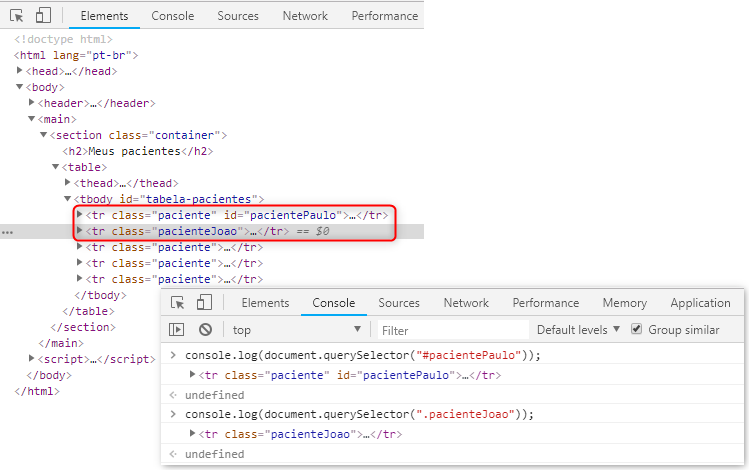


O **QuerySelector** pode ser utilizado para buscar elementos filhos.

* Seguindo o exemplo acima, poderia ser feito: **titulo.querySelector(“buscaElementosFilhos”)**;

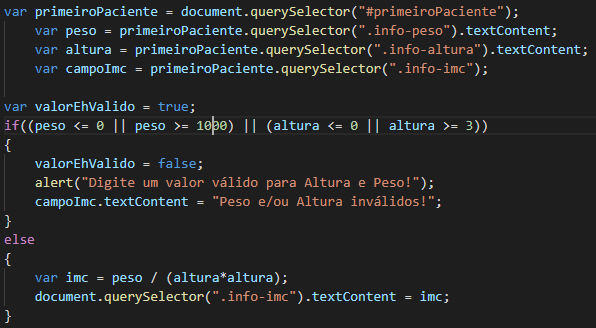
**1.3 Boas Práticas**

* **DESACOPLAMENTO COM O HTML**: Não buscar pelo **nome da TAG,** e sim pelo **ID(#titulo) ou** pelo nome da **CLASSE(.titulo).**



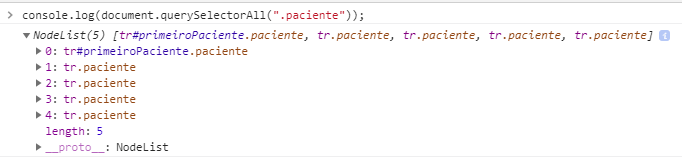
* **SEPARAR O CÓDIGO JS DO HTML**: Referenciar o arquivo JS no HTML.
* **REFERENCIAR O ARQUIVO JS NO FIM DO BODY**: Não é boa prática, é **obrigação**. O navegador renderiza o DOM de acordo com a ordem do arquivo HTML, portanto os elementos só serão encontrados após a página HTML estar formada.

# **VARIÁVEIS E OPERADORES**

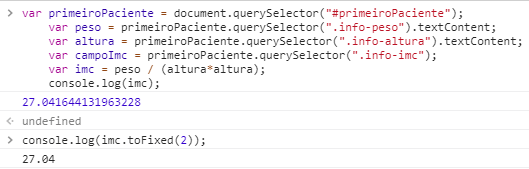
* 

# **ARRAYS, LOOPS E ESTILOS**

* **QuerySelectorAll(“x”): retorna um array de elementos, de acordo com o filtro.**



* **toFixed(int): limita o número de casas decimais a serem exibidas.**

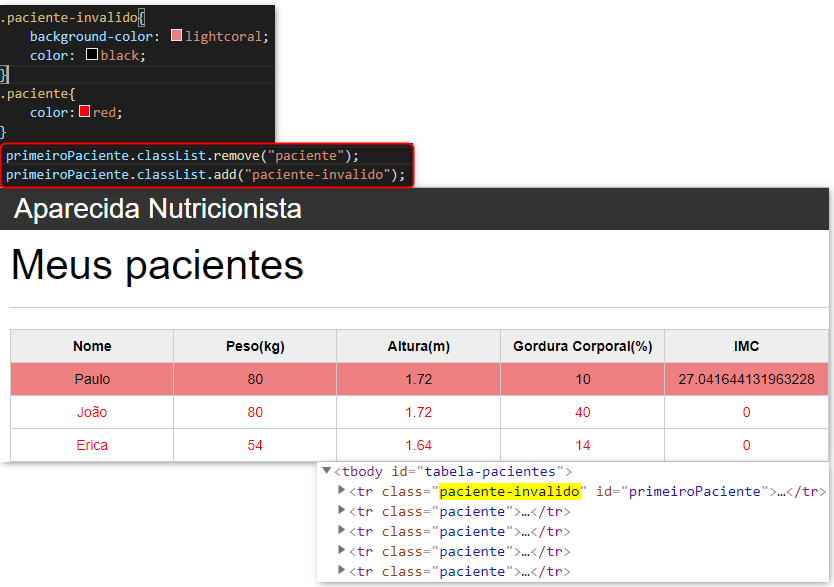


* **elemento.style: altera o desgin dos elementos. (CamelCase)**

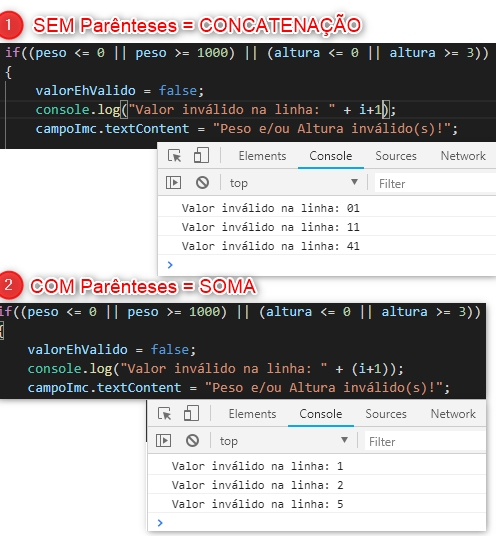
**NÃO É BOA PRÁTICA utilizar a propriedade STYLE, é importante MANTER AS RESPONSABILIDADES SEPARADAS.**

**IDEAL: criar uma classe no CSS para adicionar esta classe no elemento do JS.**

* **elemento.classList: referencia uma classe do CSS.**

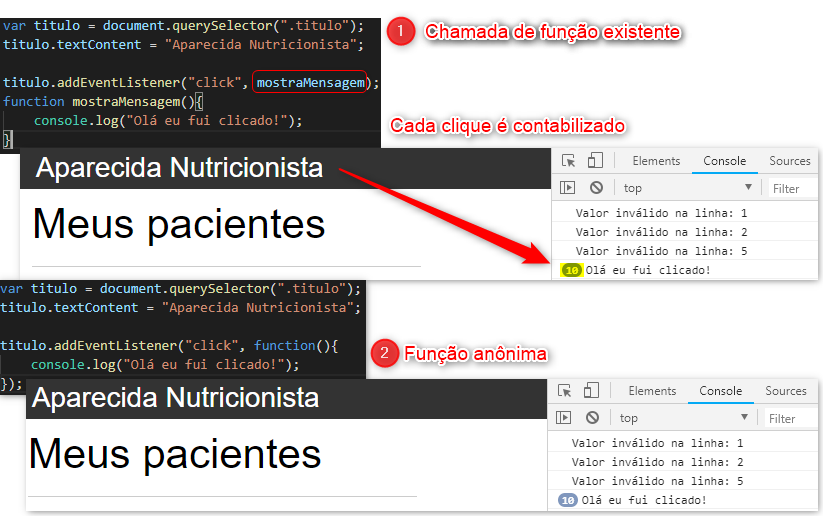


* **COLOCAR PARÊNTESES AO REALIZAR OPERAÇÕES MATEMÁTICAS A SER EXIBIDA:**

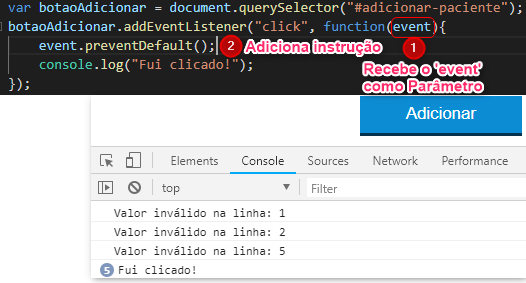


# **EVENTOS, FORMULÁRIOS E CRIANDO ELEMENTOS**

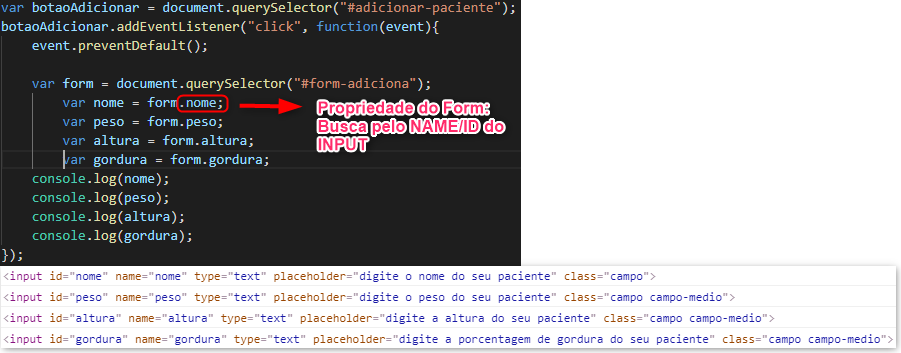
* elemento.**addEventListener**(“evento”,nome\_da\_função);
  + Método que **escuta os eventos** de um determinado elemento da página e **executa uma função** (EXISTENTE OU ANÔNIMA).



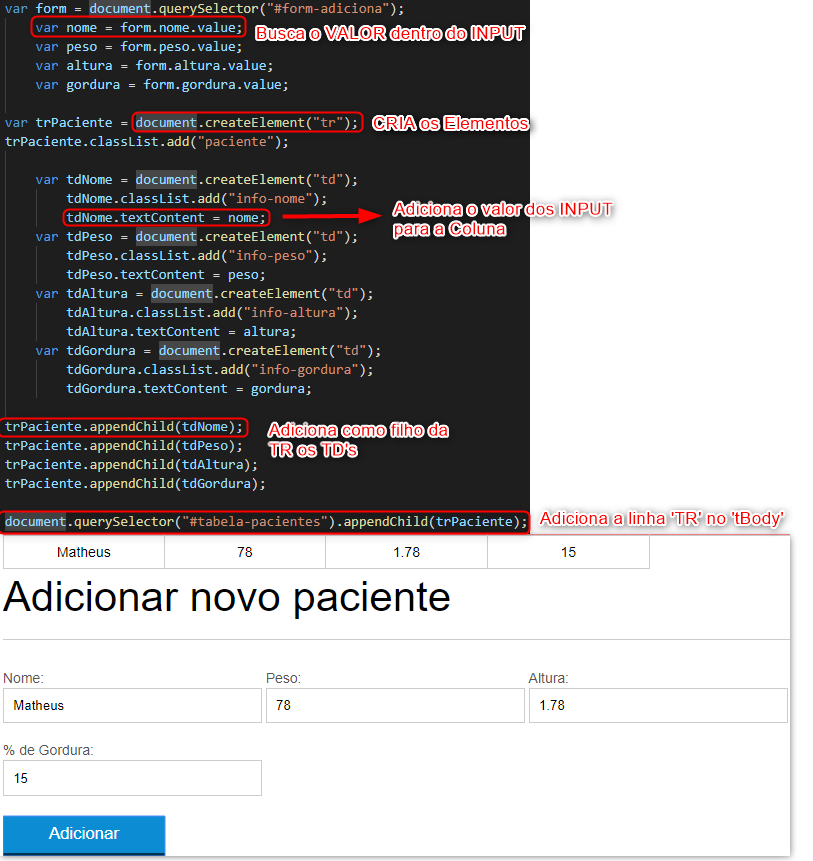
* **event.preventDefault**();
  + Previne o comportamento PADRÃO do evento feito em um elemento.
  + **Ex**: **Botão dentro de formulário** OU **Input type=’SUBMIT’** => Ao ser clicado, seu comportamento padrão dentro do formulário é limpar os dados dos campos, submeter o formulário e recarregar a página.



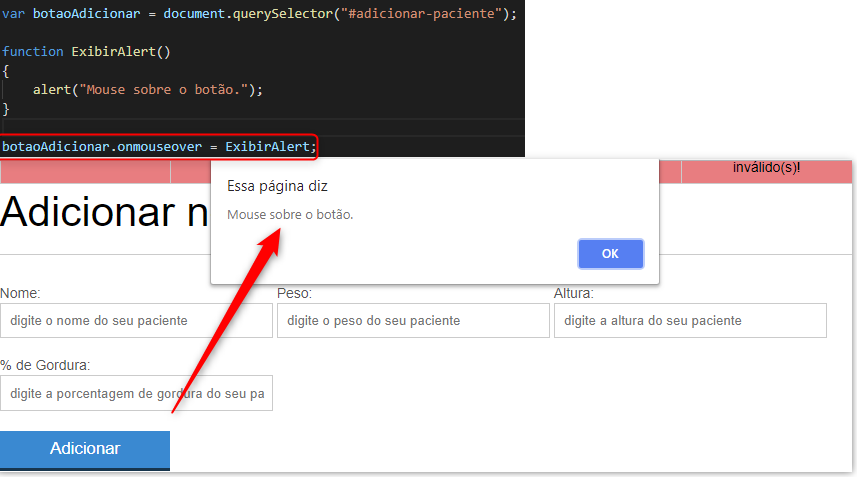
* **Buscando INPUTS do FORM:** após buscar pelo FORM, não é preciso utilizar o **querySelector** para buscar os INPUTS. O FORM recebe propriedades de acordo com o **NAME** ou **ID** dos **INPUTS**.



* **CRIANDO ELEMENTOS E ADICIONANDO COMO FILHO**
  + document**.createElement(“**nome\_do\_elemento**”);**
  + elemento.**appendChild(**elemento**);**

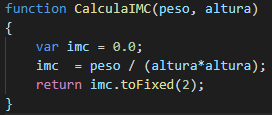


* **Event Shortcut: on + nome\_do\_evento**(MINÚSCULO)
  + Deve ser utilizado quando existir SEMPRE um **único método** para o evento x elemento.

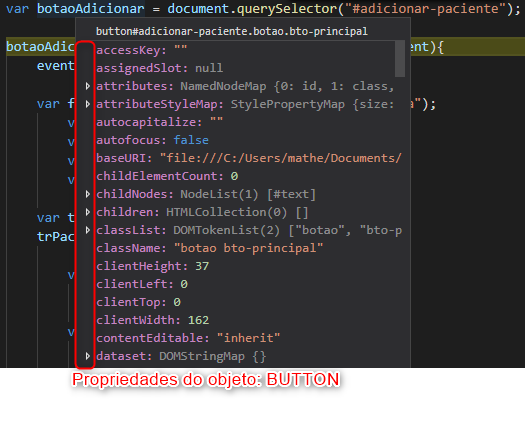


# **BOAS PRÁTICAS COM JAVASCRIPT**

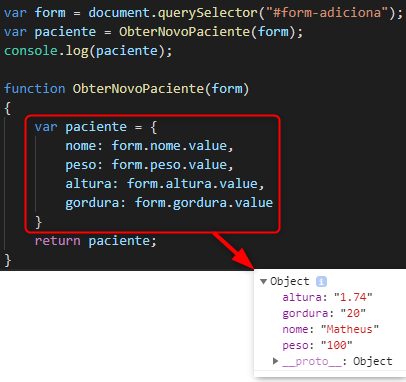
* Dividir o código em **arquivos JS** de acordo com a funcionalidade. **Ex**:
  + **Arquivo 1**: Código da página principal = responsável pelo Cálculo de IMC.
  + **Arquivo 2** - Código que manipula os dados/ações **do formulário**.
* Centralizar instruções repetidas em ***function***’s. Em funções utilizadas em mais de um arquivo, é importante criar um arquivo somente de **funções externas**.
  + Ex: **CalcularIMC**(peso,altura): função que será utilizada mais de uma vez e em mais de um arquivo.



* Criando e Manipulando **OBJETOS**
  + Ao utilizar o **querySelector**, recebemos um **OBJETO**;



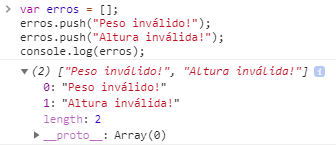
* + Criando e retornando um **objeto**:



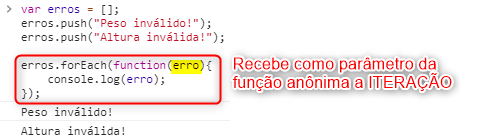
* + form.**reset**();
    - Limpa os dados dos INPUTS.

# **VALIDAÇÃO DE FORMULÁRIOS**

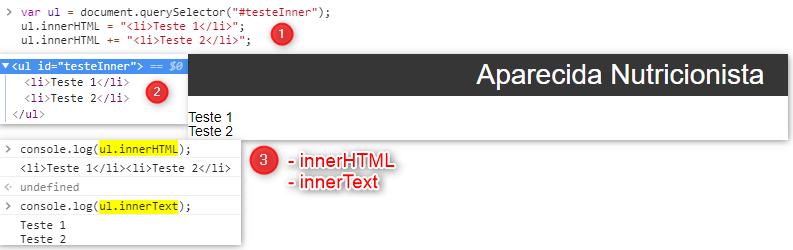
* Declarando **Array** de Erros e **Adicionando** itens



* Percorrendo o Array com o **forEach**



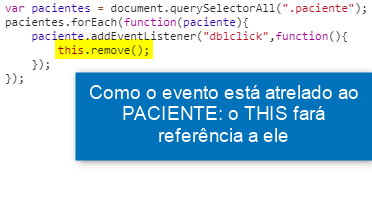
* elemento.**innerHTML**: conteúdo dentro de um componente do HTML.
  + Permite manipular diretamente o HTML por meio de uma **STRING**.



* **Eventos**: <https://developer.mozilla.org/en-US/docs/Web/Events>

# **REMOÇÃO, DELEGAÇÃO E ANIMAÇÃO**

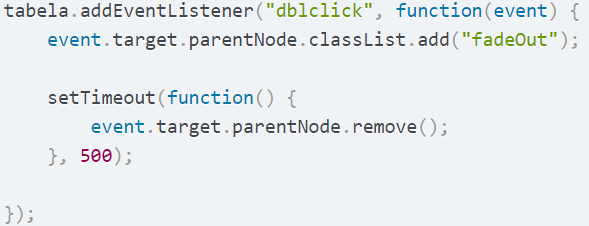
* Referenciando o acionador do evento com o **this**
* elemento**.remove():** remove o componente



* **Event Bubbling** *(“Borbulhamento” de Eventos):*Ao clicar no elemento mais profundo na árvore do DOM, o pai/avô/(...) escutam este evento, afinal os filhos estão ‘dentro’ do pai.
  + **event**.**target:** identifica o ALVO do evento. Logo, é possível utilizar o pai para ESCUTAR os eventos e a partir do evento dentro do pai, conseguimos **identificar o filho alvo da ação**.
  + **elemento**.**parentNode:** retorna o elemento PAI.

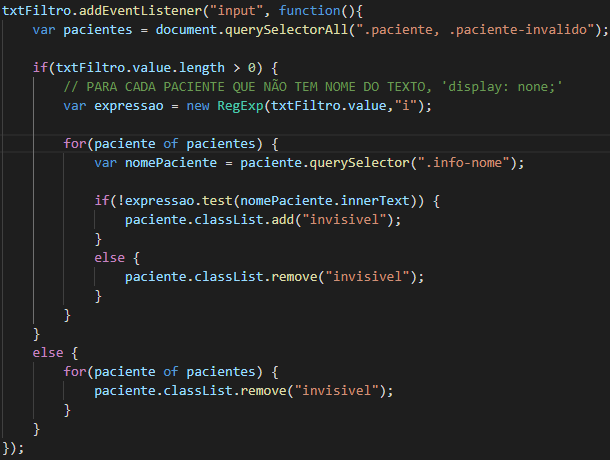


* **setTimeout()**: executa as instruções dentro do bloco após um tempo determinado.



# **FILTRANDO UMA TABELA**

* Através de evento **“input”,** é possível escutar cada dígito dentro de um campo.
* **Expressões Regulares**: Utiliza busca do ‘CTRL + F’



# **AJAX – Requisições Assíncronas**

Requisições que podem ser feitas sem prejudicar as outras funcionalidades - **Assíncronas**.

**1 –** Instanciar o objeto que fará a requisição: **XMLHttpRequest()**

**2 –** Definir qual a **URL** e qual o **Método** da requisição: **.open(“method”,”URL”)**

**.send()**: Faz a requisição.

**3 -** Adicionar **escutador** do evento **“load”** para a referência ao **REQUEST**,pois assim que carregar existirá uma **resposta**.

**4 -** Utilizar a propriedade **responseText**, para obter a **string** da resposta

**5 – JSON.parse(‘stringJSON’):** Converter para o formato **JSON.**

**6 -** Utilizar a propriedade **status**: Para tratar as requisições.