# Correção de Quest:
# JavaScript Intermediário

**Aluno: Guilherme Tavares - Turma 08**
https://github.com/GuiTavs7/form_with_validation_by_js

## Requisitos Obrigatórios:

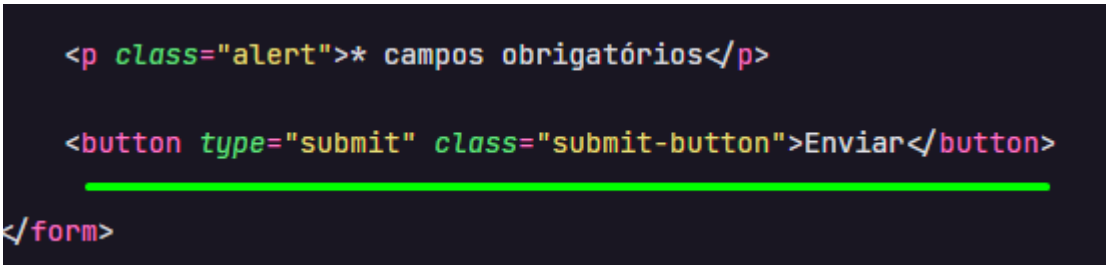
- ~~A Validação deve ser feita com JS Vanilla.~~
- ~~O fundo do formulário deve ser feito usando a imagem em anexo na aula.~~
- ~~Ao clicar para enviar o formulário, se caso algum campo não estiver preenchido, a borda do input deve ficar vermelha e uma mensagem de "campo obrigatório" deve aparecer embaixo do campo que não foi preenchido, conforme o Figma.~~

## Pontuações:

1. Em um formulário, tanto o **input** do tipo “**button**” quanto a **tag button** com o atributo **type** definido como “**button**” podem ser usados para criar botões em formulários HTML. No entanto, a **tag button** oferece mais flexibilidade em termos de personalização e permite incluir outros elementos dentro do botão. Já o **input** é mais adequado quando se precisa enviar um valor junto com o formulário.

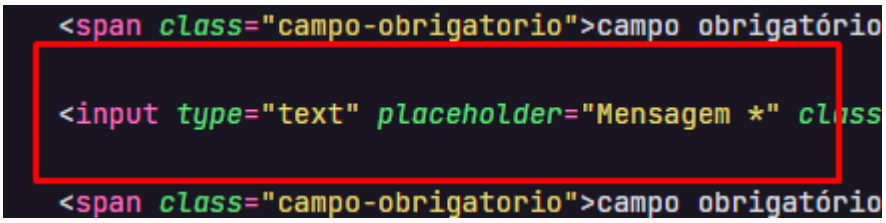
   Neste formulário, como o botão “enviar” apenas envia os dados dos outros inputs para a validação, o mais

correto seria utilizar a **tag button** com o atributo **type** definido como “**button**”, exatamente como você fez.
*Ex:*



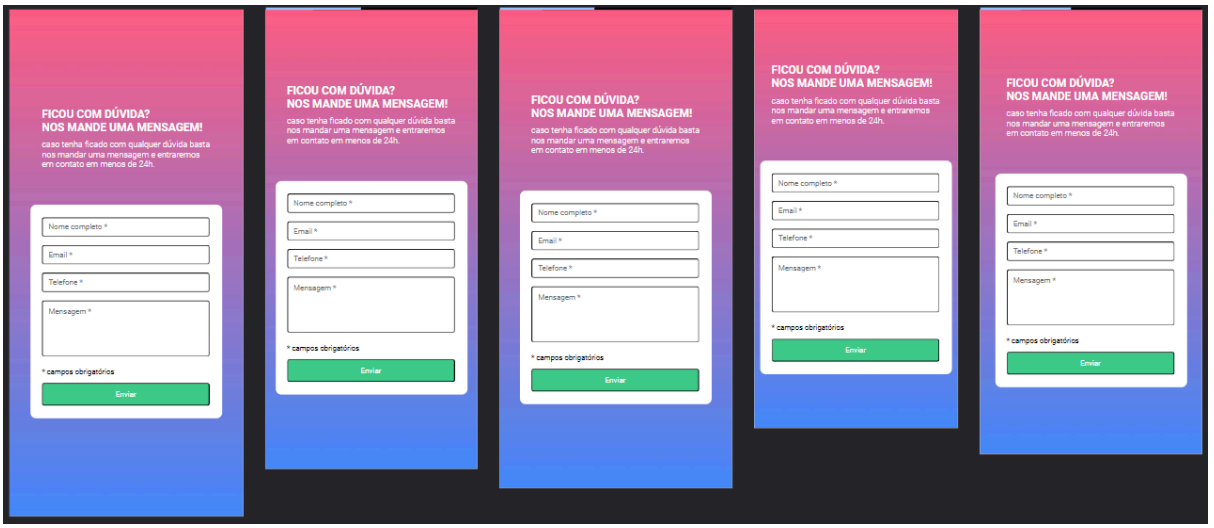

Por fim, só bastaria colocar as suas **classes** ou **ID’s** que ficaria igual. Caso não ficasse, bastaria consertar alguns pequenos pontos com o CSS.

2. Aqui **poderia ter feito** uso da **Tag TextArea**:




Essa tag é especialmente recomendada para uma *área de texto* editável em uma página web. Ela permite que usuários possam escrever e manipular um texto mais longo, como em um **formulário de comentários** ou em uma **caixa de texto para a digitação de mensagens**.

3. Percebi que seu projeto **foi bem adaptado às diferentes telas**. Apesar de não ser obrigatório, garantir que seus projetos sejam responsivos é uma prática importante, pois muitos usuários acessam aplicativos em diferentes telas.




4. Gostei da lógica do JS, tudo funciona como deveria. No entanto, por se tratar de uma validação em **vários** campos, seria uma ótima oportunidade para treinar o uso do **ForEach** - ensinado nas aulas anteriores.

   O uso do **ForEach** iria dispensar a necessidade de criar várias validações, pois ele iria automaticamente fazer a validação **em cada item** da função.

   Vou te dar um exemplo mais visual de como poderia ter feito, depois você compara com a sua versão e faz anotações sobre o que achar importante:

```
const camposFormulario = document.querySelectorAll('.campo')
const botaoEnviar = document.querySelector('.btn-enviar')

botaoEnviar.addEventListener('click', (e) =>{
    e.preventDefault()

    camposFormulario.forEach((input)=>{
        if (input.value) {
            input.classList.add('valido')
            input.nextElementSibling.classList.remove('mostrar')
        } else {
            input.classList.remove('valido')
            input.classList.add('erro')
            input.nextElementSibling.classList.add('mostrar')
        }
    })
})
```

Vou te explicar bloco por bloco:

a. **Nessa primeira parte do código**, eu usei o método *querySelectorAll()* para selecionar todos os elementos que possuem a classe **campo** e coloquei eles na variável **camposFormulario**. Em seguida, usei o método *querySelector()* para selecionar o elemento que possui a classe **btn-enviar** o armazenei na variável chamada **botaoEnviar**.

b. **Nessa segunda parte do código**, eu adicionei um evento de clique ao botão de envio, usando o método *addEventListener()*. Quando o botão é clicado, a função que é passada como segundo parâmetro é executada. Nesse caso, a primeira linha dessa função chama o método *preventDefault()* do objeto *Event* passado como

parâmetro para evitar que a página seja recarregada quando o botão é clicado.

c. **Nessa terceira última parte do código**, eu usei o método *forEach()* para percorrer todos os elementos armazenados na variável **camposFormulario**, que são os campos do formulário que devem ser validados. **Para cada campo**, o código verifica se o valor do campo (*acessado através da propriedade value*) está preenchido ou não. **Se estiver preenchido**, a classe **valido** é adicionada ao elemento e a classe **mostrar** é removida do elemento irmão (usando o método *nextElementSibling*). Se não estiver preenchido, a classe **valido** é removida, a classe **erro** é adicionada e a classe **mostrar** é adicionada ao elemento irmão. Assim, o usuário visualiza uma mensagem de erro indicando que o campo precisa ser preenchido corretamente.

   **OBS**: É importante lembrar que dois códigos fazem a mesma coisa, que é validar campos de formulário e exibir uma mensagem de erro quando necessário. A principal diferença entre o meu código e o seu é que o meu é um pouco mais simples e direto, usando uma condicional **if/else** para adicionar ou remover classes diretamente nos elementos do formulário, enquanto a sua versão utiliza **funções e condicionais separadas** para realizar essas ações, o que pode dificultar um pouco mais a leitura do código **e gerar algumas inconsistências**.
   **Seu código de 45 linhas passou a ter 18.**

No fim, completou os desafios de JavaScript Intermediário, tá mandando bem!
Anota essas observações, se preferir, e vai treinando tudo isso. Usa essas mesmas observações nos próximos projetos que vão te ajudar bastante.
Como desafio final, tenta refatorar esse seu código usando essas dicas, com a prática você pega o jeito da coisa.

**~ Boa sorte, Guilherme!** ☕