**Correção de Quest:**

**JavaScript Intermediário**

**Aluno: Paulo Víctor de Oliveira Horn - Turma 16**

[**https://github.com/PauloHorn/DevQuest-Turma-16/tree/main/modulo-javascript-intermediario/21-exercicios/desafio-js**](https://github.com/PauloHorn/DevQuest-Turma-16/tree/main/modulo-javascript-intermediario/21-exercicios/desafio-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. Logo de cara é possível observar um erro muito perigoso:

- Seu site não é renderizado na tela e nem no GitHub Pages. Isso ocorre porque seu arquivo html principal (**index.html**) não está na raíz do repositório:  


Isso é péssimo, tanto para recrutadores quanto para “usuários comuns”. Recomendo que dê uma assistida nas primeiras aulas - onde os gêmeos explicam o conceito do **index.html** e porque ele deve estar sempre na raíz do seu repositório.

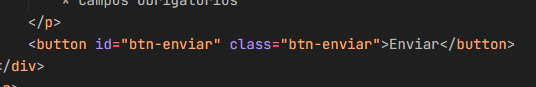
Não é errado criar esse repo pra ir praticando durante as aulas e ir jogando conteúdo aí dentro - pelo contrário, é uma ótima forma de estudar e praticar junto dos gêmeos. No entanto, cada **Projeto** deve ter seu próprio repositório então nas próximas Quests, desafios técnicos ou projetos únicos, tenta criar um repo pra cada um, garanto que vai te ajudar bastante.

**Cada Projeto** deve ter seu próprio repositório então nas próximas Quests, desafios técnicos ou projetos únicos, tenta criar um repo pra cada um, garanto que vai te ajudar bastante.

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 “**submit**”, 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.

1. 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**.

1. Na **tag TextArea**, existe uma propriedade CSS chamada **Resize**, ela é responsável por esse pedacinho aqui da tag:



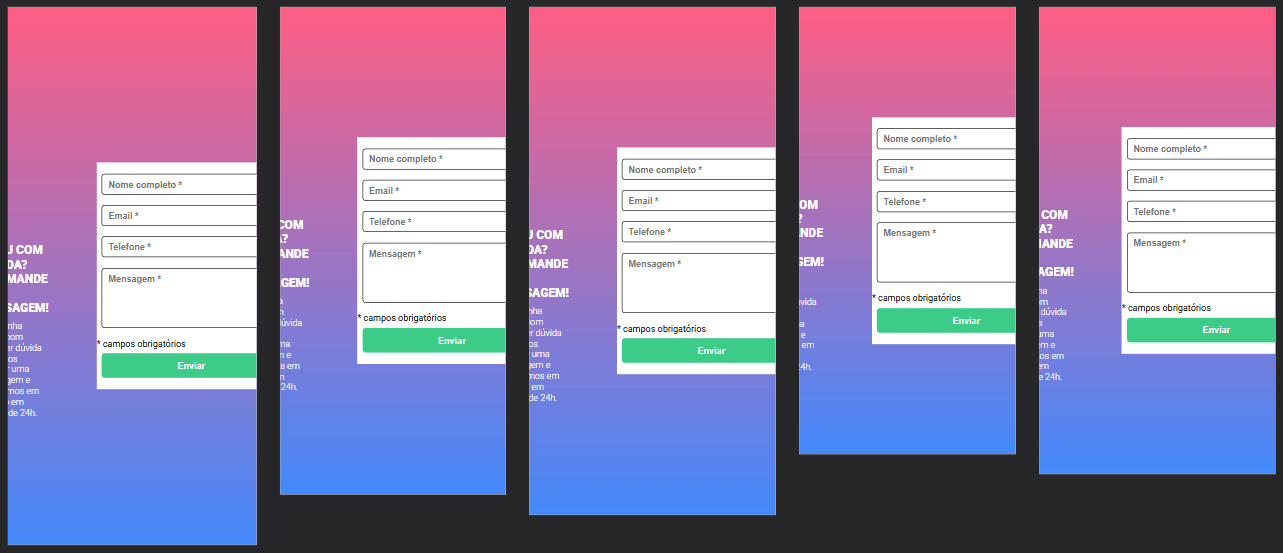
Avalia bem o contexto, se não for útil é bom que seja removida pois o usuário pode acabar manipulando de forma errada.

*EX:*

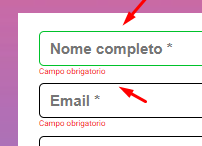


Para remover, basta declarar a propriedade **Resize** como **None**.

1. Percebi que seu projeto **não 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.



1. Seu projeto apresenta alguns problemas:

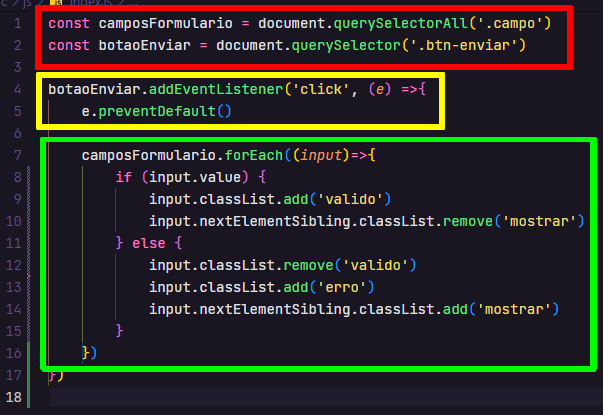


Ao submeter o formulário com o campo preenchido e logo em seguida esvaziá-lo, **o campo permanece com a borda verde + mensagem de erro.** No lugar disso, deveria o campo deveria ficar vermelho novamente junto da mensagem “**Campo Obrigatório\***”. Além disso, a borda do campo nunca preenchido também deveria ficar vermelha.

1. 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:



Vou te explicar bloco por bloco:

* 1. **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**.
  2. **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.
  3. **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.

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, Paulo!** ☕