# Correção de Quest - JavaScript Avançado

**Aluno: Guilherme Tavares - Turma 08**

https://github.com/GuiTavs7/OctoFind

## Requisitos Obrigatórios:

- ~~Número de seguidores.~~
- ~~Número de seguindo.~~
- ~~10 eventos com Create e Push.~~
- ~~Endpoint recomendado.~~
- ~~Titulo do Repo + MSG do Evento.~~
- ~~Forks - Estrelas - Watchers - Linguagem Programação.~~

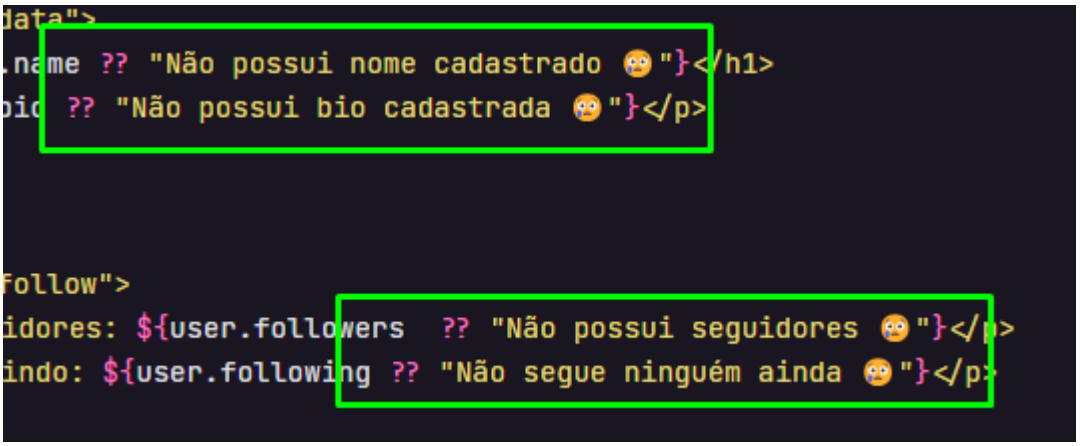
## Pontuações:

1. Gostei da criatividade com as caixas, criou seu próprio estilo e o manteve por todo o site:

2. Bom uso da validação do formulário usando a tecla ENTER:

```
document.getElementById("btn-search").addEventListener("click", () => {

    const userName = document.getElementById("input-search").value;

    if(validateEmptyInput(userName)) return;

    getUserData(userName);
})

// 2) Criando o evento de buscar os dados do usuário através do pressioname

document.getElementById("input-search").addEventListener("keyup", (e) => {
    const userName = e.target.value;
    const key = e.which || e.keyCode;
    const isEnterKeyPressed = key === 13;

    if(isEnterKeyPressed){

        if(validateEmptyInput(userName)) return;

        getUserData(userName);
    }
})
```

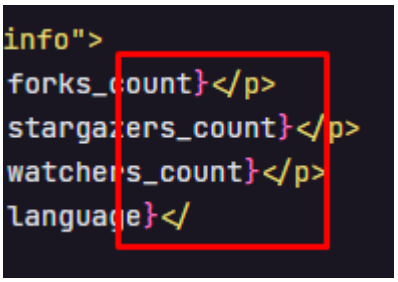
3. Tratou bem os dados que retornam Null/Undefined:




Bom uso do operador de Coalescência Nula, percebi que praticou bastante - **é uma ótima ideia e**

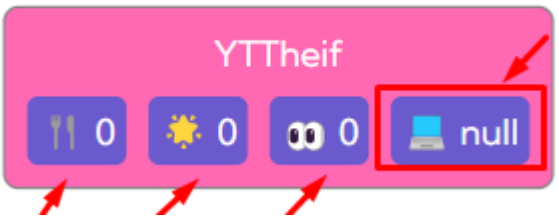
**demonstra domínio dos conceitos apresentados nas aulas**. Parabéns!

4. Aqui cabe um sisteminha de validação para tratar caso as informações do usuário venha como Null ou Undefined:



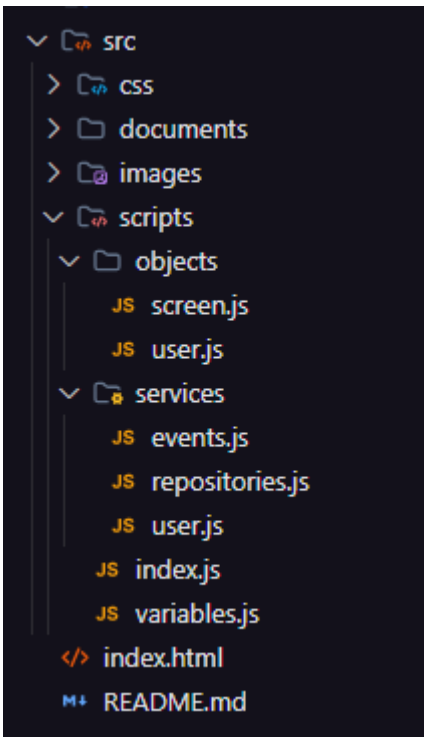

   Poderia ter seguido a mesma lógica do operador de Coalescência Nula que você já tinha feito um pouco acima.

   Evitaria isso:




   Poderia aproveitar e colocar algo como “Sem forks”, “Sem estrelas” e assim por diante em todos os outros itens, especialmente no campo das **Linguagens**.

5. Gostei da estrutura de pastas, organização excelente e com ótimos nomes:

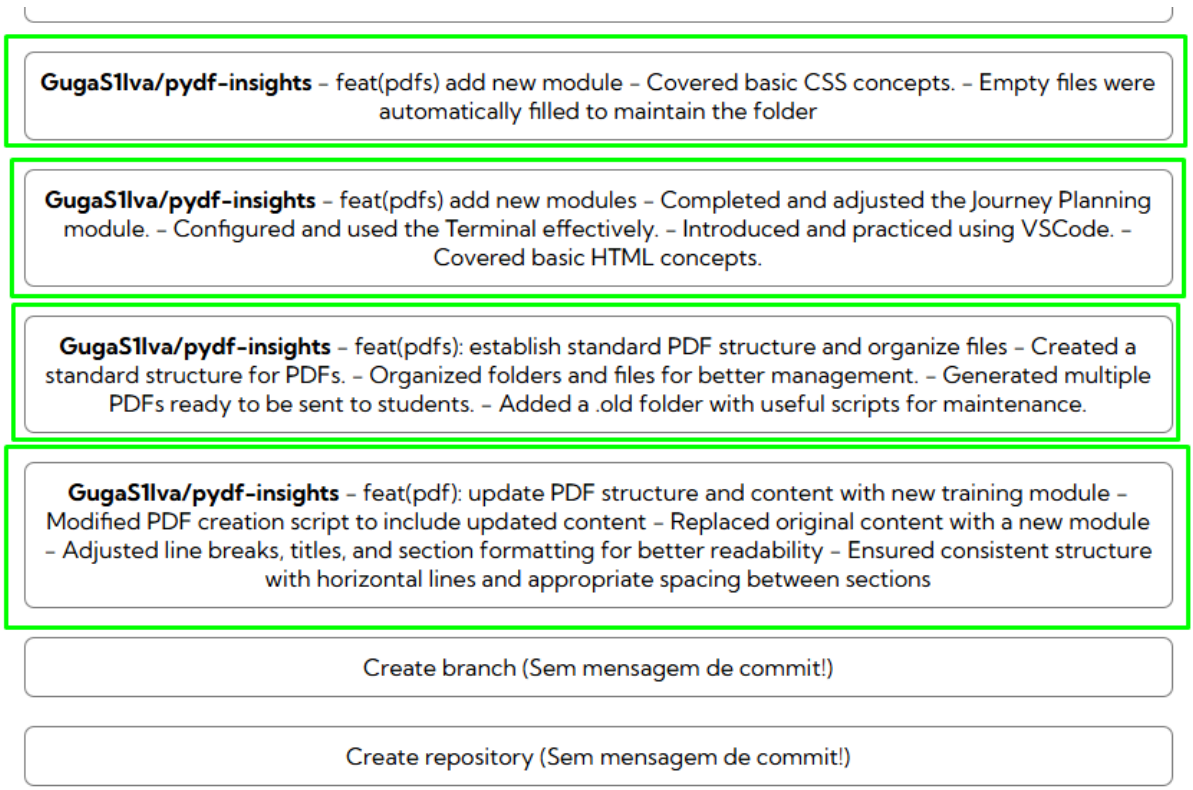



Separou bem as responsabilidades, as telas, os serviços e funcionalidades do projeto.

6. Trabalhou muito bem a **indentação** do teu código:

Eu sei que essa parte não tem **auto-complete** mas é importante tentar ao máximo deixar o seu código organizado.

7. As atividades dos usuários **estão sendo bem carregadas**:




Ficou muito bom!

8. A busca pelos diferentes tipos de eventos do usuário foi feita corretamente. No entanto, seria um ótimo caso de uso para praticar os métodos **Filter()** e **Slice()** fazendo uma busca personalizada dos eventos do usuário:

```
async function getEvents(userName) {
    const response = await fetch(`${baseUrl}${userName}/events`);
    const events = await response.json();
    return events.filter(element => element.type === 'CreateEvent'|| element.type === 'PushEvent').slice(0,
    eventsQuantity);
};
export{ getEvents };
```

```
let eventsItens = '';
user.events.forEach(element =>{
    if (element.type === 'PushEvent') {
        eventsItens += `<li>
    <h3>${element.repo.name}</h3>
    <p> -- ${element.payload.commits[0].message}</p>
    </li>`
    }else{
        eventsItens += `<li>
        <h3>${element.repo.name}</h3>
        <p> -- Criado um ${element.payload.ref_type}</p>
        </li>`
    };
});
```

O uso desses dois métodos tornaria o código mais conciso e fácil de entender, permitindo filtrar **apenas os eventos necessários** e **limitando o número de eventos retornados**.

Seria importante, nessa Quest, realizar a **validação** e o **tratamento** dos **diferentes tipos de eventos git** que o usuário pode ter feito. Utilizando o **Filter()**, o **Slice()** e o **ForEach()**, seria possível garantir um melhor desempenho do projeto e evitar possíveis erros.

No fim, completou os desafios de JavaScript Avançado, 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!** ☕