# **PLANO DE AULA**

Aula 20 | Tempo estimado: 1 hora e 30 minutos | Web Starter

Tipo da atividade: Offline

Ferramenta(s): Computador, Git, nodeJS e VS Code

Conteúdos

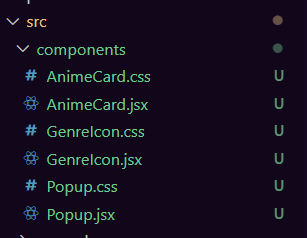
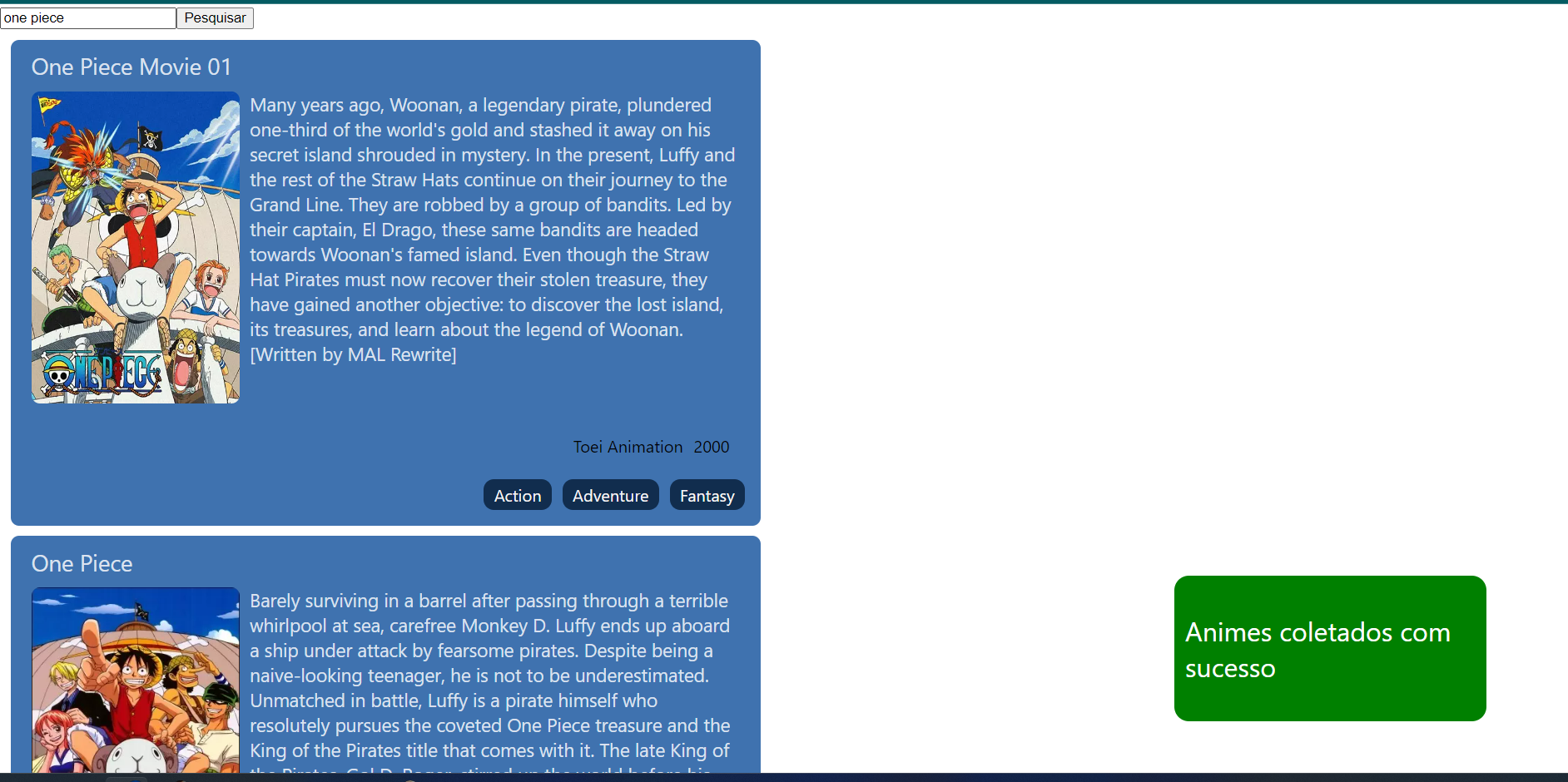
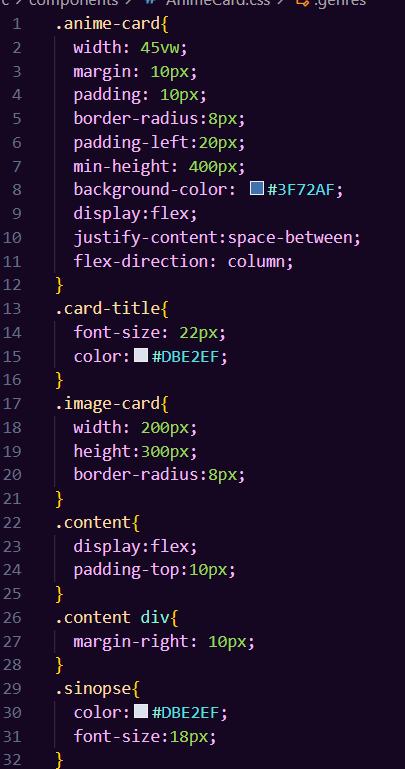
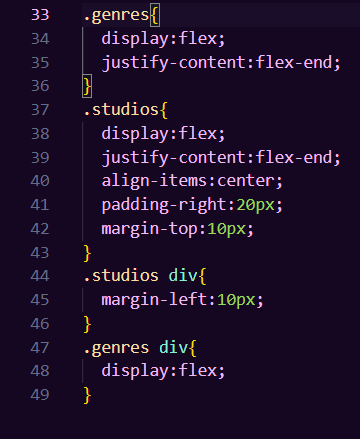
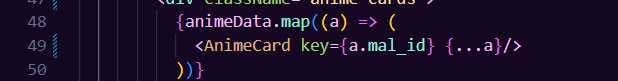
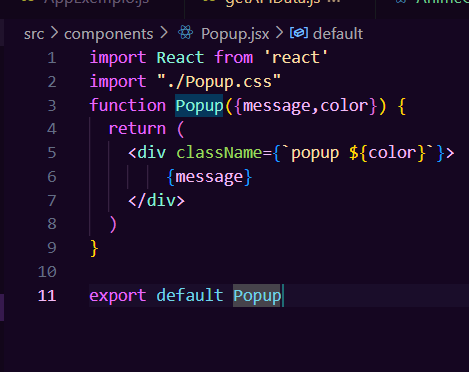
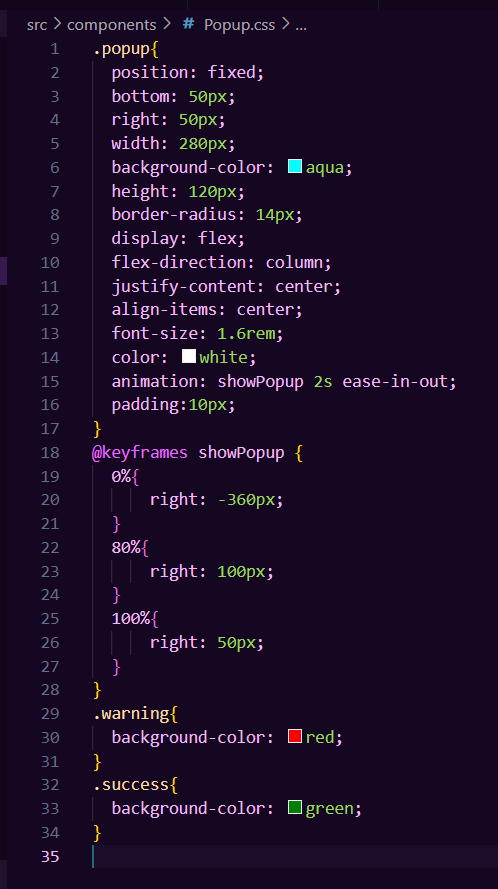
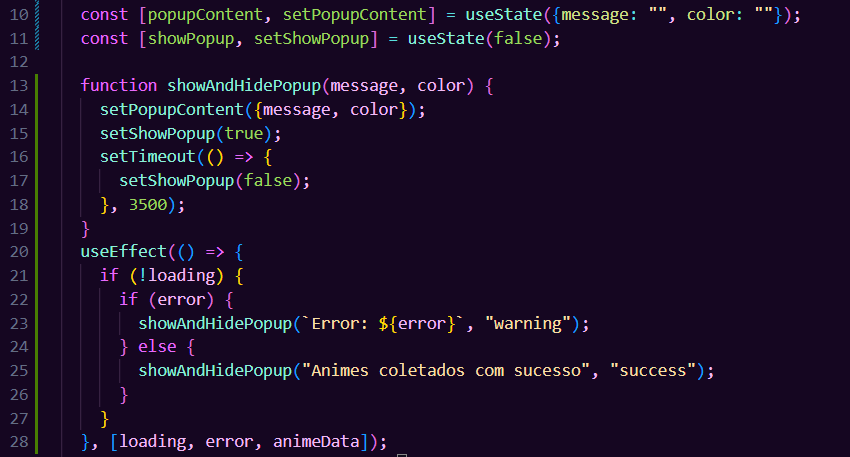
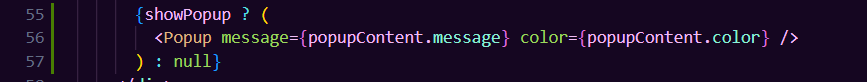
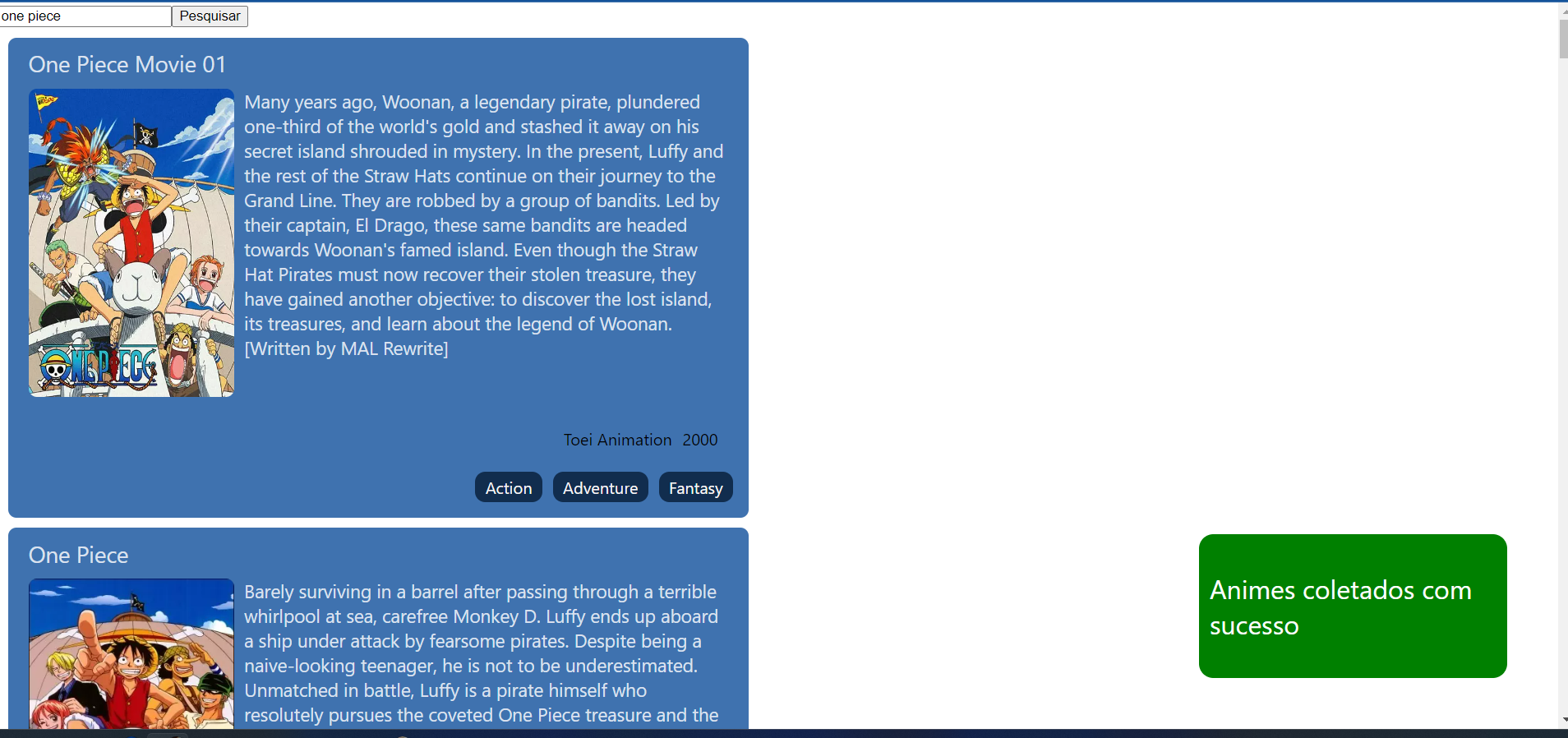
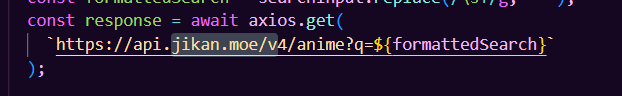
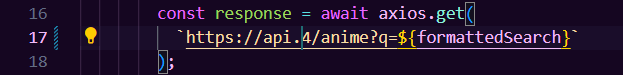
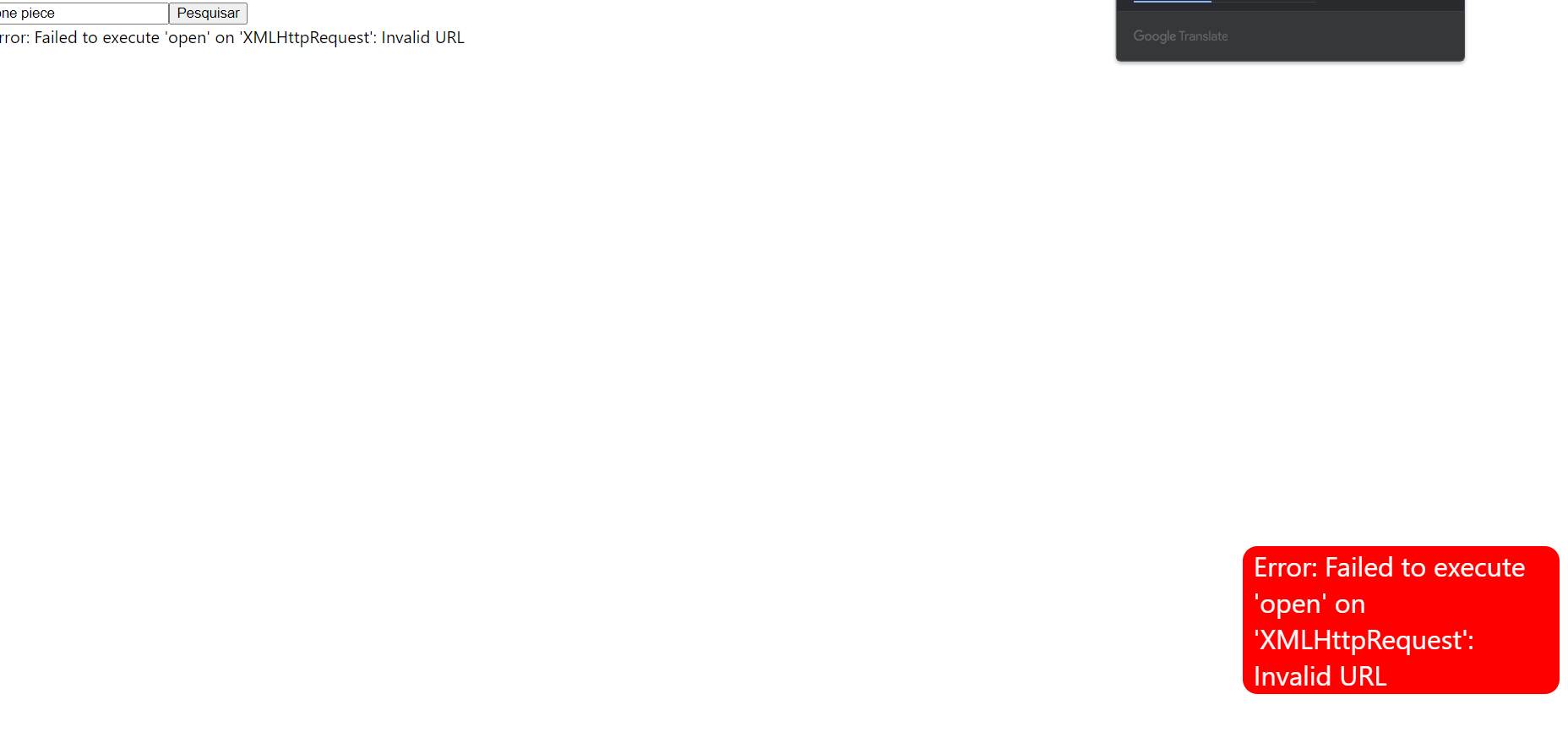
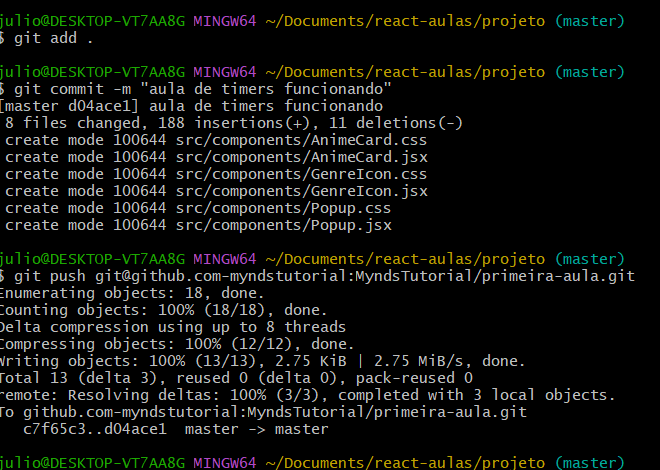
* React Timers e Animações.

Objetivos

* Ensinar sobre React;
* Aprender a instalar bibliotecas;
* Configurar um *package.json*.

Estratégias e atividades

* Criar um App que consome API.

1. Hoje vamos instalar e deixar o site bonito, isso é muito importante. Para isto, vamos criar estes arquivos:  
   
2. Vamos exibir um aviso no canto da tela que diz o que está acontecendo, mas vamos começar com como ficará o site no final:   
   
3. Começaremos pelos *animes card*. Ele será o quadrado azul, onde contém as informações, repare que ele também usa outro componente chamado *GenreIcon* que é aqueles ‘cardzinhos’ pequenos azuis:  
     
   
4. O arquivo *GenreIcon* receberá o nome que precisa mostrar:  
   
5. Agora vamos estilizar o *GenreIcon*:  
   
6. Para estilizar o card é um pouco diferente:  
     
   
7. E agora precisamos implementar o card criado no App.js (não esqueça de importar):  
   
8. Vamos criar o popup, que é aquele aviso verde na imagem de cima. O arquivo *popup.js* é bem tranquilo e ele recebe como props uma mensagem e uma cor, ficando assim:  
     
   
9. E o CSS do popup terá cores dinâmicas:  
     
   O ‘success’ será chamado quando der certo a chamada API e o ‘warning’ quando não funcionar.
10. Agora, como implementar no App.js? Primeiro vamos criar uma função para mostrar e esconder o popup usando o *setTimeout* para poder esconder depois de um tempo, sempre que a API for finalizada o código do componente vai ser renderizado e mostrado na tela:  
    
11. Porém, agora como vamos chamar o popup na tela?  
    
12. E vualá! Funcionando:  
    
13. Para simular um erro, apague um pedaço da URL do *geturl api*:  
      
      
    
14. Para finalizar, vamos subir o código no Github.

Recursos

* Git Chrome e VS Code.

Observação

* Sempre que você quiser testar algum exemplo em JavaScript, você pode digitar um código direto no navegador como por exemplo:



* Professor: por se tratar de uma API de animes, ela é como se fosse o Google. Muito cuidado com o que o aluno for pesquisar! Ele tem acesso a internet pois é uma aula de web 🙂
* Como funcionam os timers? Faça esse código com os alunos no navegador, a função *setTimeout* faz algo acontecer depois do tempo que é passado no segundo argumento:  
  

Tarefas

* Fazer um código usando o *setTimeout* para a próxima aula. Não se esqueça de salvar no seu Github.