INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 14 - CONSUMO DE API

O QUE IREMOS APRENDER

01 API

02 FETCH

03 CONSUMINDO UMA API

04 SOLICITAÇÕES HTTP

05 MÉTODOS HTTP

06 CHAVES DE API

07 MÃOS AO CÓDIGO

API

API é a sigla para "Interface de Programação de Aplicativo". Ela é como um conjunto de regras e protocolos que permite que diferentes programas de computador se comuniquem e interajam entre si. Imagine uma analogia com um restaurante:

1. Você é o cliente.
2. O garçom é a API.
3. A cozinha é o sistema ou serviço que você deseja usar.

API

Ainda não entendeu? Vamos mostrar um exemplo prático.

Suponha que você esteja construindo um aplicativo de previsão do tempo e deseja obter dados de previsão do tempo de um serviço online. Em vez de ter que entender todos os detalhes de como o serviço funciona internamente, você pode usar a API do serviço de previsão do tempo. A API fornece um conjunto de regras (como URLs específicas e parâmetros de solicitação) que permitem que seu aplicativo solicite dados de previsão do tempo

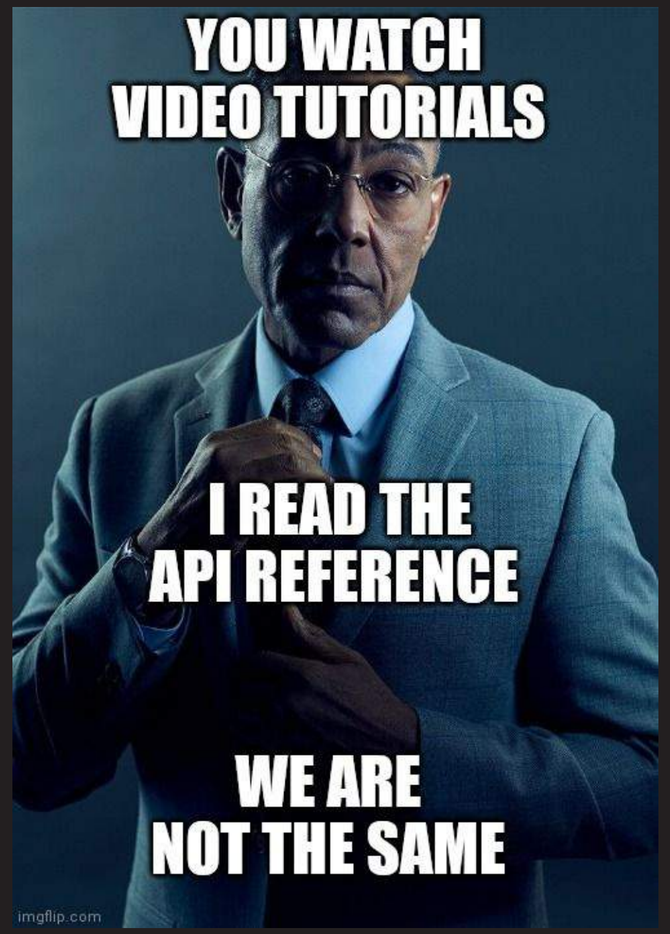
API

e receba as informações necessárias em um formato que seu aplicativo possa entender.

Em resumo, uma API é como um intermediário que permite que diferentes programas se comuniquem e compartilhem informações de maneira padronizada, simplificando a interação entre eles. Ela é essencial para o desenvolvimento de software moderno e para que diferentes sistemas possam trabalhar juntos de forma eficiente.

API

Agora ficou mais fácil, né?
Para consumirmos uma API em JavaScript, podemos usar o fetch ou o axios. Neste módulo, iremos aprender a consumir apenas via fetch, mas você pode estar pesquisando como consumir via axios.



FETCH

O fetch é uma função em JavaScript que é usada para fazer requisições de recursos (como dados de APIs ou arquivos) em uma rede. Ele é uma parte fundamental da API Fetch, introduzida no JavaScript para modernizar e simplificar a forma como as requisições de rede são feitas.

FETCH

Assíncrono: O fetch é uma operação assíncrona, o que significa que ele não bloqueia a execução do restante do código. Isso permite que outras operações continuem enquanto a requisição de rede está em andamento.

Promessas: O fetch retorna uma promessa (Promise) que é resolvida quando a requisição é concluída. Isso permite que você lide com os resultados da requisição de forma assíncrona usando .then() e .catch() para tratar sucesso e erros, respectivamente.

JSON: Normalmente, o fetch é usado para buscar dados em formato JSON de APIs, mas ele também pode ser usado para buscar outros tipos de dados, como texto ou arquivos binários.

CONSUMINDO UMA API

1.Criar uma solicitação com fetch

Você começa criando uma solicitação HTTP usando a função fetch(). Fornecendo a URL da API que deseja acessar como argumento para fetch()

```
const apiUrl = 'https://api.exemplo.com/dados'; // Substitua pela URL da API que deseja usar
    fetch(apiUrl)
        .then(response => {
            // Manipule a resposta da API aqui
        })
        .catch(error => {
            // Faça o tratamento dos erros aqui, caso encontre algum problema na solicitação
        });
```

CONSUMINDO UMA API

2. Tratar a resposta

O fetch() retorna uma promessa (Promise), o que significa que você pode usar .then() para tratar a resposta quando ela estiver disponível.

```
const apiUrl = 'https://api.exemplo.com/dados'; // Substitua pela URL da API que deseja usar
    fetch(apiUrl)
        .then(response => {
            if (!response.ok){
                throw new Error('A solicitação não foi bem-sucedida.');
            }
            return response.json();
        })
        .then(data => {
            // Faça algo com os dados da API (agora em formato JSON)
            console.log(data)
        })
        .catch(error => {
            // Faça o tratamento dos erros aqui, caso encontre algum problema na solicitação
            alert('Erro: ' + error.message)
        });
```

CONSUMINDO UMA API

3. Lidar com os erros

Assim como vimos na aula passada, o método catch é usado para lidar com erros que ocorrem durante a requisição ou qualquer parte do encadeamento de Promessas. É recomendado usar o catch após o then para capturar erros que podem ocorrer durante o processamento da resposta ou qualquer operação assíncrona subsequente.

CONSUMINDO UMA API

3. Lidar com os erros

```
const apiUrl = 'https://api.exemplo.com/dados'; // Substitua pela URL da API que deseja usar
    fetch(apiUrl)
        .then(response => {
            if (!response.ok){
                throw new Error('A solicitação não foi bem-sucedida.');
            }
            return response.json();
        })
        .then(data => {
            // Faça algo com os dados da API (agora em formato JSON)
            console.log(data)
        })
        .catch(error => {
            // Faça o tratamento dos erros aqui, caso encontre algum problema na solicitação
            alert('Erro: ' + error.message)
        });
```

CONSUMINDO UMA API

```
<script>
    const botao = document.getElementById('botao');
    const catImages = document.getElementById('catImages');
    botao.addEventListener('click', fetchCatImages);

    function fetchCatImages() {
        fetch('https://api.thecatapi.com/v1/imagens/search?limit=1') // Busca uma imagem de gatos
            .then(response => {
                if (!reponse.ok) {
                    throw new Error('A solicitação não foi bem-sucedida.');
                }
                return response.json();
            })
            .then(data => {
                catImages.innerHTML = ''; // Limpa as imagens anteriores
                data.forEach(cat => {
                    const catImage = document.createElement('img');
                    catImage.src = cat.url;
                    catImage.alt = 'iamgem de gato';
                    catImages.appendChild(catImage);
                });
            })
            .catch(error => {
                console.error('Erro: ',error);
            });
    }
</script>
```

CONSUMINDO UMA API

No código anterior :

Usamos a URL https://api.thecatapi.com/v1/images/search?limit=1 para buscar uma imagem aleatória de gatos da API "The Cat API" (você pode ajustar o número conforme desejado).

Quando o botão "Clique aqui para ver gatos" é clicado, a função fetchCatImages é chamada e busca as imagens de gatos da API.

CONSUMINDO UMA API

No código anterior :

As imagens são exibidas na página e substituem quaisquer imagens anteriores sempre que o botão é clicado.

Este código permite buscar e exibir várias imagens de gatos usando um loop forEach.

Recomendamos que você teste o código.

SOLICITAÇÕES HTTP

2. Métodos HTTP:

GET: Usado para solicitar recursos de um servidor (leitura).
POST: Usado para enviar dados para um servidor (escrita).
PUT: Usado para atualizar recursos existentes no servidor (atualização).
DELETE: Usado para remover recursos em um servidor (exclusão).

SOLICITAÇÕES HTTP

O HTTP é o protocolo fundamental da World Wide Web (WWW) usado para a comunicação entre clientes (como navegadores da web) e servidores web.

1.Funcionamento:
Os clientes enviam solicitações HTTP aos servidores para acessar recursos (páginas da web, imagens, arquivos, etc.).

Os servidores respondem com respostas HTTP que contêm os dados ou informações solicitadas.

MÉTODOS HTTP

- GET: Recuperando Dados

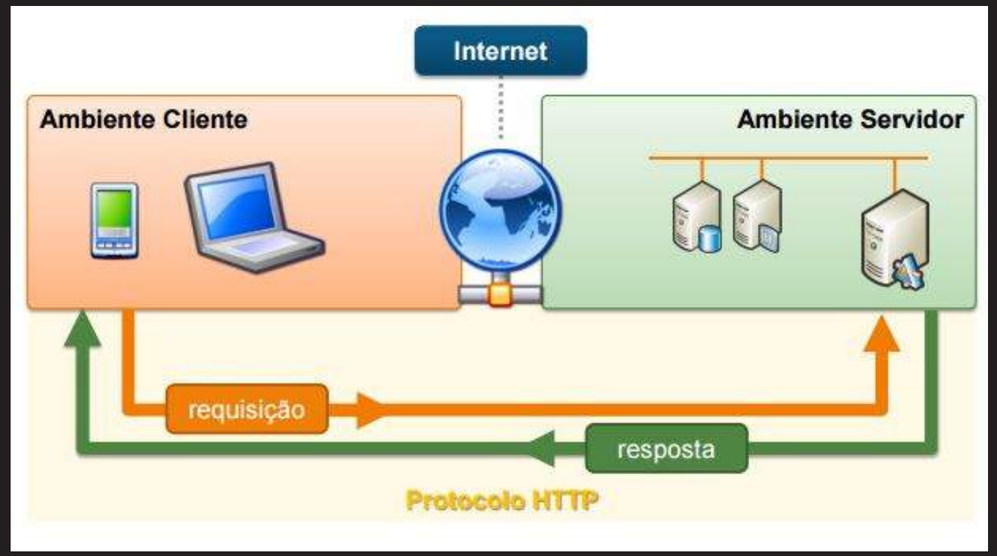
Usado para buscar informações.

Exemplo de uso: Obter informações de um livro por seu ISBN.

- POST: Enviando Dados

Usado para criar novos recursos.

Exemplo de uso: Enviar dados de um novo usuário para registro.



MÉTODOS HTTP

- PUT: Atualizando Dados

Usado para atualizar recursos existentes.

Exemplo de uso: Atualizar os detalhes de um perfil de usuário.

- DELETE: Excluindo Dados

Usado para excluir recursos.

Exemplo de uso: Excluir um item de uma lista de tarefas.

CHAVES DE API

Uma chave de API é um código que permite que você acesse e utilize serviços ou recursos oferecidos por um sistema ou aplicativo.

No entanto, eu não posso fornecer chaves de API específicas, pois elas são normalmente geradas e fornecidas pelos provedores de serviços. Nem todas as APIs possuem chaves, somente as APIs que são restritas. As chaves são gerados após o login do usuário e incluídos nas solicitações para autorização.

CHAVES DE API

Esse código ao lado, é um, exemplo de uma API que precisa de uma chave para a requisição ser bem sucedida. É sempre importante ler a documentação da API pois sempre informa se a API precisa de chave ou não.

```
const apikey = 'SUA_CHAVE_DE_API'; // Substitua 'SUA CHAVE_DE_API pela sua chave de API rea
// URL da API do OpenWeatherMap para obter informações meteorológicas
const apiUrl = `https://api.openweathermap.org/data/2.5/weather?q=London&appid=${apikey}`;
// Fazendo uma requisição GET para a API
fetch(apiUrl)
    .then(response => {
        if (Iresponse.ok) {
        throw new Error('Não foi possível obter os dados');
       }
        return response.json();
    })
    .then(data => {
        // Processando os dados retornados pela API
        console.log('Condição climática em Londres:');
        console.log(`Tempo: ${data.weather[0].description}`);
        console.log(`Temperatura: ${data.main.temp}°C`);
        console.log(`Umidade: ${data.main.humidity}%`);
    })
    .catch(error => {
        console.error('Erro ao obter dados meteorológicos:', error);
    });
```

ATIVIDADE PRÁTICA

Atividade 01

Crie uma aplicação web usando JavaScript que utilize a API do GitHub para recuperar informações de um usuário específico e exiba o nome do usuário, a imagem do proprietário e a biografia do GitHub. A aplicação deve permitir ao usuário inserir o nome de usuário do GitHub, clicar em um botão para buscar os repositórios e, em seguida, exibir os dados recuperados em uma interface de usuário amigável.

Link de onde encontrar API:https://api.github.com/

Link da API: https://api.github.com/users/{seu_usuario}

ATIVIDADE PRÁTICA

Atividade 02

Você está desenvolvendo um aplicativo de entrega de produtos online e precisa integrar uma API de CEP (Código de Endereçamento Postal) para verificar o local dos clientes. Para obter informações precisas de localização, a fim de otimizar o processo de entrega em um aplicativo de comércio eletrônico é necessário utilizar uma API de cep. Link de onde encontrar API: https://viacep.com.br/

Link da API:https://viacep.com.br/ws/{seu_cep}/json/

ATIVIDADE PRÁTICA

Atividade 03

Você está desenvolvendo um aplicativo que utiliza a PokéAPI para exibir informações sobre Pokémon. Os usuários podem pesquisar Pokémon por nome ou número da Pokédex.

Link de onde encontrar API: https://pokeapi.co/docs/v2

Link da API:https://pokeapi.co/api/v2/pokemon/{nome_pokemon}

DESAFIO PRÁTICO

Você foi contratado para desenvolver um catálogo de filmes online. A aplicação deve permitir aos usuários buscar informações sobre filmes, como título, ano de lançamento, gênero, e avaliação. Para isso, você utilizará a API pública do The Movie Database (TMDb).

Requisitos:

- A aplicação deve possuir uma página inicial com uma lista de filmes populares.

DESAFIO PRÁTICO

- Os usuários devem poder buscar filmes inserindo um termo na barra de pesquisa.
- Ao clicar em um filme, a aplicação deve exibir detalhes, como sinopse, elenco, e outras informações relevantes.
- A interface deve ser atraente e responsiva.

URL do link da API: https://www.themoviedb.org/

SE LIGA NO CONTEÚDO DA PRÓXIMA AULA!

AULA 15 DE JAVASCRIPT.
PROJETO

IN

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

INFINITY SCHOOL

VISUAL ART CREATIVE CENTER

AULA 14 - CONSUMO DE API