FMX Academy

INFORMAÇÕES		
MÓDULO	AULA	INSTRUTOR
04 JavaScript Essencial e Consumo de APIs	07 - Integrando Tudo: APIs, Async e Fluxos Complexos	Pedro Henrique Amadio

CONTEÚDO

1. Na última aula...

Antes de mergulharmos nos projetos finais, vamos relembrar o que vimos na última aula (Aula 06). Ótima hora para levantar dúvidas.

- que é API, endpoint e métodos HTTP.
- JSON.parse / JSON.stringify.
- fetch + async/await + try/catch.
- APIs reais: GitHub, ViaCEP, AdviceSlip.
- Criação de elementos com dados externos.

2. Trabalhando com Múltiplas APIs ao Mesmo Tempo (Promise.all)

Até agora, sempre consumimos **uma API por vez.**
Mas no projeto final, vamos precisar de **duas APIs simultaneamente**:

- GitHub -> dados do usuário
- Clima -> temperatura usando geolocalização

Buscar os dados em sequência funcionaria, mas seria **lento.**
Para resolver isso, entra o:

Promise.all(): Executando requisições em paralelo
Permite rodar várias Promises ao mesmo tempo, economizando tempo.

Exemplo simples:

```
async function carregarDados() {
  const [perfil, clima] = await Promise.all([
    fetch("https://api.github.com/users/pedro-amadio").then(r => r.json()),

fetch("https://api.weatherapi.com/v1/current.json?key=API_KEY&q=Aracatuba")
.then(r => r.json())
  ]);

  console.log(perfil);
  console.log(clima);
}
```

Por que usar?

- Mais rápido
- Organiza melhor o código
- Muito comum em sistemas reais

O que acontece se uma API falha?
Promise.all() interrompe tudo e joga erro e isso é bom, pois podemos tratar com try/catch.
É mais seguro para não gerar inconsistências.

3. Geolocalização: obtendo latitude e longitude reais

O navegador possui algumas APIs, uma delas permite capturar a localização do usuário com:

```
navigator.geolocation.getCurrentPosition()
```

Exemplo

```
navigator.geolocation.getCurrentPosition(
  (pos) => {
    const { latitude, longitude } = pos.coords;
    console.log(latitude, longitude);
  },
  (erro) => {
    console.log("Não foi possível obter a localização.");
  }
);
```

O que precisamos saber?

chrome: chrome://flags/#unsafely-treat-insecure-origin-as-s
edge: edge://flags/#unsafely-treat-insecure-origin-as-secure

- O navegador pede permissão;
- Às vezes é negado, devemos tratar erro;
- Podemos usar as coordenadas em APIs

4. Padrões Profissionais de Código (Boas Práticas)

Antes do projeto final, é essencial reforçar:

Nomes claros

```
getWeather()   // bom
PegarNegocio() // ruim
```

DRY (Don't Repeat Yourself)
Evitar repetir código, ou seja, criar funções.

Funções pequenas
Uma função = uma responsabilidade

Tratamento de erro obrigatório
Nenhuma chamada de API pode existir **sem try/catch.**

5. Fluxo Assíncrono Completo (Padrão Profissional)

Vamos ter como base de desenvolvimento do nosso projeto final o fluxo a seguir:
1. Usuário interage (clique)
2. Validamos a entrada
3. Chamamos a API (fetch)
4. Esperamos com await
5. Tratamos erros com try/catch

6. Manipulamos o DOM
7. Exibimos a resposta

6. Projeto em Sala - “Meu Mini Perfil Integrado / Dashboard”

Vamos desenvolver um Mini Perfil integrado. Usaremos algumas APIs como:

- GitHub: Para trazer as informações do seu perfil no GitHub;
- Navigator.geolocation: Para obter a localização atual do usuário;
- Cotação (https://economia.awesomeapi.com.br/json/last/USD-BRL): Para obtermos dados de cotação do dia atual.
- Frases temáticas: Usaremos a API https://api.adviceslip.com/ para obter fraases de temas escolhidos.

Sem clique, sem formulário.
Nesta atividade vamos montar um **mini dashboard** que carrega automaticamente assim que a página é aberta.

Ao carregar a página, o JavaScript irá:

- Buscar os dados de um usuário fixo do GitHub
- Tentar obter a localização atual do usuário usando o navegador
- Buscar a cotação atual do dólar (USD → BRL)
- Buscar uma frase temática (conselho) em uma API externa
- Montar um painel com todas essas informações na tela

Organização do JavaScript (em um único arquivo)
1- Primeiramente iremos definir as variáveis/constantes;

```
// ------- Área 1: Seletores do DOM e constantes -------
const areaResultado = document.getElementById("resultado");

const GITHUB_USER = "pedro-amadio";     // Usuário fixo do GitHub
const CONSELHO_TEMA = "life";           // Tema fixo para a frase
```

2- Funções e APIs

```
// ------- Área 2: Funções de API -------

async function buscarGithub() {
  const url = `https://api.github.com/users/${GITHUB_USER}`;
  const resp = await fetch(url);

  if (!resp.ok) {
    throw new Error("Erro ao buscar usuário do GitHub.");
  }

  return resp.json();
}

async function buscarCotacaoDolar() {
  const url = "https://economia.awesomeapi.com.br/json/last/USD-BRL";
  const resp = await fetch(url);

  if (!resp.ok) {
    throw new Error("Erro ao buscar cotação do dólar.");
  }

  return resp.json();
}

async function buscarConselho(tema) {
  const url = `https://api.adviceslip.com/advice/search/${tema}`;
  const resp = await fetch(url);
  const dados = await resp.json();

  if (!dados.slips || !dados.slips.length) {
    throw new Error("Nenhum conselho encontrado para esse tema.");
  }

  // Pega o primeiro conselho
  return dados.slips[0].advice;
}

// Envolve geolocalização em uma Promise para poder usar await
function obterLocalizacao() {
  return new Promise((resolve, reject) => {
    if (!navigator.geolocation) {
      reject(new Error("Geolocalização não suportada."));
      return;
    }

    navigator.geolocation.getCurrentPosition(resolve, reject);
  });
}
```

3- Função para render

```
// ------- Área 3: Função de renderização -------

function renderizarDashboard(perfil, cotacao, conselho, latitude,
longitude) {
  const dolar = cotacao.USDBRL;

  areaResultado.innerHTML = `
    <section>
      <h3>Perfil GitHub</h3>
      <img src="${perfil.avatar_url}" width="120" alt="Avatar GitHub">
      <p><strong>Nome:</strong> ${perfil.name ?? "(sem nome)"}</p>
      <p><strong>Login:</strong> ${perfil.login}</p>
      <p><strong>Repositórios públicos:</strong> ${perfil.public_repos}</p>
    </section>

    <section>
      <h3>Localização atual</h3>
      <p>Latitude: ${latitude !== null ? latitude.toFixed(4) : "Não
disponível"}</p>
      <p>Longitude: ${longitude !== null ? longitude.toFixed(4) : "Não
disponível"}</p>
    </section>

    <section>
      <h3>Cotação do Dólar (USD → BRL)</h3>
      <p>${dolar.name}</p>
      <p>1 USD = R$ ${Number(dolar.bid).toFixed(4)}</p>
    </section>

    <section>
      <h3>Conselho do dia (${CONSELHO_TEMA})</h3>
      <p>"${conselho}"</p>
    </section>
  `;
}
```

4- Evento principal que será responsável por chamar e executar tudo

```
// ------- Área 4: Evento principal (carregar ao abrir a página) -------

window.addEventListener("DOMContentLoaded", async () => {
  areaResultado.textContent = "Carregando dashboard...";

  try {
    // Tentamos pegar a localização, mas se der erro seguimos sem ela
    let latitude = null;
    let longitude = null;

    try {
      const pos = await obterLocalizacao();
      latitude = pos.coords.latitude;
      longitude = pos.coords.longitude;
    } catch {
      // Se localização falhar, simplesmente deixamos como null
      console.warn("Não foi possível obter a localização.");
    }

    // Buscamos as outras APIs em paralelo
    const [perfil, cotacao, conselho] = await Promise.all([
      buscarGithub(),
      buscarCotacaoDolar(),
      buscarConselho(CONSELHO_TEMA)
    ]);

    renderizarDashboard(perfil, cotacao, conselho, latitude, longitude);
  } catch (erro) {
    console.error(erro);
    areaResultado.textContent = "Erro ao carregar o dashboard. Tente
novamente mais tarde.";
  }
});
```

7. Desafios Extras

Para quem terminar mais rápido ou para estudar em casa:

- Adicionar:
  - Seguidores (followers) e seguindo (following) do GitHub
  - Data/hora da cotação (create_date)
- Adicionar um “load” visual (tipo “...” ou ícone) enquanto carrega
- Estilizar as <section> como cartões usando CSS

8. Preparem-se para a nossa avaliação!

Chegamos ao fim de mais um módulo!
Agora é com vocês, coloque em prática tudo que vimos em sala, publique seus projetos no GitHub e compartilhe os esforços de vocês!