Vou lhe explicar um projeto de uma aplicação Web que usa como base dos dados um json.

Apenas entenda cada linha de código, sua lógica e regras de renderização de conteúdo. Não precisa sugerir nenhuma refatoração por enquanto.

Este **json** possui a seguine estrutura (esta é uma versão reduzida, mas que contempla todos os tipos que vamos usar):

{

  "type": "SlideGroup",

  "id": 0,

  "title": "Nosso Trânsito Multimídia",

  "productId": "000001",

  "items": [

    {

      "type": "SlideGroup",

      "id": 5,

      "title": "Legislação de Trânsito",

      "items": [

        {

          "type": "SlideGroup",

          "id": 15,

          "title": "Formação do Condutor",

          "items": [

            {

              "type": "Slide",

              "id": 61,

              "bookReference": "4",

              "title": "Pré-requisitos Para Habilitação",

              "subtitle": null,

              "audio": "",

              "elements": [

                {

                  "type": "TextElement",

                  "styleName": "title1",

                  "text": "Pré-requisitos Para"

                },

                {

                  "type": "GroupElement",

                  "elements": [

                    {

                      "type": "TextElement",

                      "styleName": "paragraph",

                      "text": "Exemplo de texto dentro de um GroupElement."

                    },

                    {

                      "type": "ImageElement",

                      "title": "Ilustração",

                      "legend": "",

                      "source": "exames para habilitação.swf",

                      "width": 0,

                      "height": 0,

                      "searchText": "avaliação psicológica\nexame de aptidão física e mental\nexame direção veicular\nexame teórico-técnico"

                    }

                  ],

                  "verticalAlign": "middle",

                  "mode": "horizontalGroup",

                  "horizontalAlign": "left",

                  "fillHeight": false

                },

                {

                  "type": "ListElement",

                  "styleName": "unorderedList",

                  "text": "Ser penalmente imputável (que possa ser processado na forma da Lei);\nSaber ler e escrever;\nPossuir documento de identidade;\nPossuir Cadastro de Pessoa Física - CPF.\n",

                  "startIndex": 1

                },

                {

                  "type": "ListElement",

                  "styleName": "numberList",

                  "text": "Primeiro passo;\nSegundo passo;\nTerceiro passo;\n",

                  "startIndex": 1

                },

                {

                  "type": "InfoBoxElement",

                  "title": "Info",

                  "elements": [

                    {

                      "type": "TextElement",

                      "styleName": "paragraph",

                      "text": "Você não poderá se habilitar ou se reabilitar se estiver respondendo por um delito de trânsito ou com sua habilitação suspensa ou cassada."

                    }

                  ],

                  "verticalAlign": "top",

                  "mode": "verticalGroup",

                  "horizontalAlign": "left",

                  "fillHeight": false,

                  "x": 932,

                  "y": 476

                },

                {

                  "type": "GridElement",

                  "alternateRowColor": true,

                  "columnNumber": 3,

                  "featureColumn": 1,

                  "columnSizes": [2.5, 2.5, 2.5],

                  "content": [

                    ["", "TEÓRICA", "PRÁTICA"],

                    ["ACC", "20 h.", "5 h."],

                    ["CNH Inicial A", "45 h.", "20 h."],

                    ["Adição Cat A", "---", "15 h."],

                    ["CNH Inicial B", "45 h.", "20 h."],

                    ["Adição Cat B", "---", "15 h."]

                  ],

                  "isFirstRowHeader": true

                },

                {

                  "type": "VideoElement",

                  "title": "Vídeo",

                  "previewImage": "direcao-defensiva-condicoes-do-condutor-via-usuario.jpg",

                  "video": "https://www.youtube.com/watch?v=VIDEO\_ID"

                },

                {

                  "type": "AppLauncherElement",

                  "path": "flecha-ao-alvo.zip"

                }

              ]

            }

          ]

        }

      ]

    }

  ]

}

**Este é nosso index.html:**

<!DOCTYPE *html*>

<html *lang*="pt-BR">

<head>

    <meta *charset*="UTF-8">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <title>Curso de Trânsito Interativo</title>

    <link *rel*="stylesheet" *href*="css/style.css">

</head>

<body>

    <!-- NOVO: Botão Hambúrguer para abrir/fechar o menu em telas pequenas -->

    <button *id*="hamburger-btn" *aria-label*="Abrir menu">

        <span></span>

        <span></span>

        <span></span>

    </button>

    <!-- NOVO: Overlay para escurecer o fundo e fechar o menu ao clicar fora -->

    <div *id*="menu-overlay" *class*="hidden"></div>

    <div *id*="app-container">

        <!-- O menu de navegação continua aqui, será controlado via CSS e JS -->

        <nav *id*="menu-navegacao">

            <h2>Índice do Curso</h2>

            <div *id*="menu-items">

                <!-- O conteúdo do menu será gerado dinamicamente pelo JS -->

            </div>

        </nav>

        <!-- A área principal onde o conteúdo do slide será exibido -->

        <main *id*="conteudo-principal">

            <header *id*="slide-header">

                <h1 *id*="slide-title"></h1>

                <h2 *id*="slide-subtitle"></h2>

            </header>

            <div *id*="slide-elements-container">

                <!-- Os elementos do slide (texto, imagens, etc.) serão renderizados aqui -->

            </div>

            <footer *id*="slide-footer">

                <div *id*="audio-player-container">

                    <audio *id*="audio-player" *controls*></audio>

                </div>

                <div *id*="controles-navegacao">

                    <button *id*="btn-anterior">Anterior</button>

                    <button *id*="btn-proximo">Próximo</button>

                </div>

            </footer>

        </main>

    </div>

    <!-- ESTRUTURA DO MODAL PARA O INFOBOX (permanece no final) -->

    <div *id*="modal-overlay" *class*="modal-hidden">

        <div *id*="modal-box">

            <header *id*="modal-header">

                <h3 *id*="modal-title"></h3>

                <button *id*="modal-close-btn" *title*="Fechar">&times;</button>

            </header>

            <div *id*="modal-content">

                <!-- O conteúdo dos elementos do InfoBox será renderizado aqui -->

            </div>

        </div>

    </div>

    <!-- Nossos scripts. O 'type="module"' permite usar import/export -->

    <script *src*="js/main.js" *type*="module"></script>

</body>

</html>

Este é nosso main.js:

/\*\*

 \* main.js

 \* Orquestrador principal da aplicação.

 \* Responsável por:

 \* - Carregar os dados do curso.

 \* - Construir a navegação interativa e responsiva.

 \* - Gerenciar o estado (slide atual).

 \* - Controlar os eventos de navegação e do modal.

 \*/

import { renderSlide, criarElemento } from './renderer.js';

// --- ELEMENTOS DA DOM ---

const menuContainer = document.getElementById('menu-items');

const btnAnterior = document.getElementById('btn-anterior');

const btnProximo = document.getElementById('btn-proximo');

const modalOverlay = document.getElementById('modal-overlay');

const modalTitle = document.getElementById('modal-title');

const modalContent = document.getElementById('modal-content');

const modalCloseBtn = document.getElementById('modal-close-btn');

const hamburgerBtn = document.getElementById('hamburger-btn');

const menuNav = document.getElementById('menu-navegacao');

const menuOverlay = document.getElementById('menu-overlay');

// --- ESTADO DA APLICAÇÃO ---

let cursoCompleto = null;

let slidesAchatados = [];

let slideAtualIndex = -1;

/\*\*

 \* Função principal que inicia a aplicação

 \*/

async function init() {

    console.log("Iniciando o curso...");

    cursoCompleto = await carregarDadosDoCurso();

    if (cursoCompleto) {

        achatarSlides(cursoCompleto.items);

        menuContainer.innerHTML = '';

        const menuPrincipal = construirNavegacao(cursoCompleto.items, 0);

        menuContainer.appendChild(menuPrincipal);

        const slideIdFromHash = parseInt(location.hash.replace('#/slide/', ''), 10);

        const startIndex = slidesAchatados.findIndex(*s* => s.id === slideIdFromHash);

        exibirSlide(startIndex !== -1 ? startIndex : 0);

    }

    configurarEventos();

}

/\*\*

 \* Carrega o data.json usando a API Fetch

 \* @returns *{Promise<Object|null>}*

 \*/

async function carregarDadosDoCurso() {

    try {

        const response = await fetch('data.json');

        if (!response.ok) throw **new** Error(`Erro HTTP! status: ${response.status}`);

        return await response.json();

    } catch (error) {

        console.error("Não foi possível carregar os dados do curso:", error);

        alert("Erro ao carregar o conteúdo do curso. Verifique o console para mais detalhes.");

        return null;

    }

}

/\*\*

 \* Cria uma lista simples (array) de todos os slides de forma recursiva.

 \* @param *{Array}* *items* - Array de itens (SlideGroups ou Slides)

 \*/

function achatarSlides(*items*) {

    for (const item of items) {

        if (item.type === 'Slide') {

            slidesAchatados.push(item);

        } else if (item.type === 'SlideGroup' && item.items) {

            achatarSlides(item.items);

        }

    }

}

/\*\*

 \* Constrói o menu de navegação de forma recursiva e retorna o elemento UL.

 \* @param *{Array}* *items* - Itens para adicionar ao menu

 \* @param *{number}* *nivel* - Nível de profundidade para indentação

 \* @returns *{HTMLUListElement}*

 \*/

function construirNavegacao(*items*, *nivel*) {

    const ul = document.createElement('ul');

    if (nivel > 0) {

        ul.style.paddingLeft = '15px';

    }

    items.forEach(*item* => {

        const li = document.createElement('li');

        if (item.type === 'Slide') {

            const a = document.createElement('a');

            a.className = 'menu-item';

            a.textContent = item.title;

            a.dataset.id = item.id;

            li.appendChild(a);

        } else if (item.type === 'SlideGroup') {

            const titleDiv = document.createElement('div');

            titleDiv.className = 'menu-group-title';

            titleDiv.textContent = item.title;

            li.appendChild(titleDiv);

            if (item.items && item.items.length > 0) {

                const subMenuUl = construirNavegacao(item.items, nivel + 1);

                subMenuUl.style.display = 'none';

                li.appendChild(subMenuUl);

            }

        }

        ul.appendChild(li);

    });

    return ul;

}

/\*\*

 \* Encontra um slide pelo seu índice na lista achatada e o exibe

 \* @param *{number}* *index* - O índice do slide no array 'slidesAchatados'

 \*/

function exibirSlide(*index*) {

    if (index < 0 || index >= slidesAchatados.length) {

        return;

    }

    slideAtualIndex = index;

    const slide = slidesAchatados[index];

    renderSlide(slide);

    btnAnterior.disabled = (slideAtualIndex === 0);

    btnProximo.disabled = (slideAtualIndex === slidesAchatados.length - 1);

    atualizarMenuActive(slide.id);

    location.hash = `#/slide/${slide.id}`;

}

/\*\*

 \* Destaca o item de menu correspondente ao slide atual

 \* @param *{number}* *slideId* - O ID do slide que está sendo exibido

 \*/

function atualizarMenuActive(*slideId*) {

    const activeItem = menuContainer.querySelector('.menu-item.active');

    if (activeItem) {

        activeItem.classList.remove('active');

    }

    const newItem = menuContainer.querySelector(`.menu-item[data-id="${slideId}"]`);

    if (newItem) {

        newItem.classList.add('active');

    }

}

/\*\*

 \* Configura os listeners de eventos para os botões, menu e modal.

 \*/

function configurarEventos() {

    // Eventos para o menu responsivo

    hamburgerBtn.addEventListener('click', () => {

        if (menuNav.classList.contains('open')) {

            fecharMenuLateral();

        } else {

            abrirMenuLateral();

        }

    });

    menuOverlay.addEventListener('click', fecharMenuLateral);

    // Listener de clique unificado para o menu

    menuContainer.addEventListener('click', (*event*) => {

        const target = event.target;

        if (target.classList.contains('menu-group-title')) {

            const subMenu = target.nextElementSibling;

            if (subMenu && subMenu.tagName === 'UL') {

                target.classList.toggle('expanded');

                subMenu.style.display = subMenu.style.display === 'none' ? 'block' : 'none';

            }

        }

        if (target.classList.contains('menu-item')) {

            const slideId = parseInt(target.dataset.id, 10);

            const index = slidesAchatados.findIndex(*s* => s.id === slideId);

            if (index !== -1) {

                exibirSlide(index);

                if (window.innerWidth <= 900) {

                    fecharMenuLateral();

                }

            }

        }

    });

    // Eventos já existentes

    btnProximo.addEventListener('click', () => exibirSlide(slideAtualIndex + 1));

    btnAnterior.addEventListener('click', () => exibirSlide(slideAtualIndex - 1));

    modalCloseBtn.addEventListener('click', fecharModal);

    modalOverlay.addEventListener('click', (*event*) => {

        if (event.target === modalOverlay) {

            fecharModal();

        }

    });

    document.addEventListener('keydown', (*event*) => {

        if (event.key === 'Escape' && !modalOverlay.classList.contains('modal-hidden')) {

            fecharModal();

        }

    });

    // NOVO: Listener para redimensionamento da janela

    window.addEventListener('resize', () => {

        // Se a janela for maior que 900px, garante que o menu mobile esteja fechado

        if (window.innerWidth > 900) {

            fecharMenuLateral();

        }

    });

}

// --- FUNÇÕES DE CONTROLE DO MENU RESPONSIVO ---

/\*\*

 \* NOVO: Abre o menu lateral em telas pequenas.

 \*/

function abrirMenuLateral() {

    menuNav.classList.add('open');

    hamburgerBtn.classList.add('active');

    menuOverlay.classList.remove('hidden');

}

/\*\*

 \* NOVO: Fecha o menu lateral em telas pequenas.

 \*/

function fecharMenuLateral() {

    menuNav.classList.remove('open');

    hamburgerBtn.classList.remove('active');

    menuOverlay.classList.add('hidden');

}

// --- FUNÇÕES DE CONTROLE DO MODAL ---

export function abrirModal(*infoBoxData*) {

    modalContent.innerHTML = '';

    modalTitle.textContent = infoBoxData.title || 'Informação';

    const groupElementData = {

        type: 'GroupElement',

        elements: infoBoxData.elements,

        mode: infoBoxData.mode || 'verticalGroup',

        verticalAlign: infoBoxData.verticalAlign,

        horizontalAlign: infoBoxData.horizontalAlign,

    };

    const content = criarElemento(groupElementData);

    if (content) {

        modalContent.appendChild(content);

    }

    modalOverlay.classList.remove('modal-hidden');

}

function fecharModal() {

    modalOverlay.classList.add('modal-hidden');

}

// Inicia a aplicação

document.addEventListener('DOMContentLoaded', init);

**Este é nosso renderer.js:**

/\*\*

 \* renderer.js

 \* Módulo responsável por renderizar o conteúdo de um slide na DOM.

 \* Ele transforma os objetos do data.json em elementos HTML.

 \*/

// Importa a função para abrir o modal, que está no main.js

import { abrirModal } from './main.js';

// Elementos da DOM onde o conteúdo será renderizado

const container = document.getElementById('slide-elements-container');

const titleEl = document.getElementById('slide-title');

const subtitleEl = document.getElementById('slide-subtitle');

const audioPlayer = document.getElementById('audio-player');

// Modo de ancoragem do InfoBox: 'auto-prev' (recomendado)

// ou 'container' (usa os valores X e Y do data.json)

const INFOBOX\_ANCHOR\_MODE = 'auto-prev';

/\*\*

 \* Atualiza a variável CSS --sec-h com a altura útil do #slide-elements-container.

 \* Isso permite ao CSS limitar a largura de players de vídeo em função da altura disponível

 \* mantendo a proporção 16:9 (ver regras com aspect-ratio no style.css).

 \*/

export function atualizarAlturaDoContainer() {

    if (!container) return;

    // clientHeight = altura interna visível (desconsidera barra de rolagem)

    const h = container.clientHeight;

    container.style.setProperty('--sec-h', h + 'px');

}

// Observa redimensionamentos de janela e do próprio container

window.addEventListener('resize', atualizarAlturaDoContainer);

if (document.readyState !== 'loading') atualizarAlturaDoContainer();

else document.addEventListener('DOMContentLoaded', atualizarAlturaDoContainer);

if (window.ResizeObserver && container) {

    const roSEC = **new** ResizeObserver(() => atualizarAlturaDoContainer());

    roSEC.observe(container);

}

/\*\*

 \* Função principal que renderiza um slide completo.

 \* @param *{Object}* *slideObject* - O objeto do slide vindo do JSON.

 \*/

export function renderSlide(*slideObject*) {

    console.log(`Renderizando slide: ID ${slideObject.id} - "${slideObject.title}"`);

    // 1. Limpa o conteúdo anterior

    container.innerHTML = '';

    // 2. Atualiza título e subtítulo

    titleEl.textContent = slideObject.title || '';

    subtitleEl.textContent = slideObject.subtitle || '';

    // 3. Atualiza o player de áudio

    if (slideObject.audio) {

        audioPlayer.src = `assets/audio/${slideObject.audio}`;

        audioPlayer.style.display = 'block';

    } else {

        audioPlayer.style.display = 'none';

        audioPlayer.src = '';

    }

    // 4. Percorre e renderiza cada elemento do slide

    if (slideObject.elements && slideObject.elements.length > 0) {

        slideObject.elements.forEach(*element* => {

            const htmlElement = criarElemento(element);

            if (htmlElement) {

                container.appendChild(htmlElement);

            }

        });

    }

    // 5. Após renderização, atualiza altura (próximo frame para layout estabilizar)

    requestAnimationFrame(() => atualizarAlturaDoContainer());

}

/\*\*

 \* Função "roteadora" que decide qual função de renderização chamar

 \* com base no tipo do elemento. Exportada para ser usada no modal.

 \* @param *{Object}* *elementObject* - O objeto do elemento.

 \* @returns *{HTMLElement|null}*

 \*/

export function criarElemento(*elementObject*) {

    // Adiciona uma verificação para garantir que o objeto existe

    if (!elementObject || !elementObject.type) {

        console.warn('Tentativa de renderizar um objeto de elemento inválido:', elementObject);

        return null;

    }

    switch (elementObject.type) {

        case 'TextElement':

            return criarTexto(elementObject);

        case 'ListElement':

            return criarLista(elementObject);

        case 'ImageElement':

            return criarImagem(elementObject);

        case 'VideoElement':

            return criarVideo(elementObject);

        case 'GridElement':

            return criarGrid(elementObject);

        case 'GroupElement':

            return criarGrupo(elementObject);

        case 'InfoBoxElement':

            return criarGatilhoInfoBox(elementObject);

        case 'SpacerElement':

            return criarEspacador(elementObject);

        case 'AppLauncherElement':

            return criarAppLauncher(elementObject);

        default:

            console.warn(`Tipo de elemento não suportado: ${elementObject.type}`);

            return null;

    }

}

// --- Funções Específicas de Renderização ---

function criarTexto(*element*) {

    const p = document.createElement('p');

    p.className = `text-element ${element.styleName || 'paragraph'}`;

    p.innerHTML = (element.text || '').replace(/\n/g, '<br>');

    return p;

}

function criarLista(*element*) {

    const isOrdered = element.styleName === 'numberList';

    const listElement = document.createElement(isOrdered ? 'ol' : 'ul');

    listElement.className = `list-element ${element.styleName}`;

    if (isOrdered && element.startIndex > 1) {

        listElement.start = element.startIndex;

    }

    const items = (element.text || '').split('\n').filter(*item* => item.trim() !== '');

    items.forEach(*itemText* => {

        const li = document.createElement('li');

        li.textContent = itemText;

        listElement.appendChild(li);

    });

    return listElement;

}

function criarImagem(*element*) {

    if (element.source && element.source.toLowerCase().endsWith('.swf')) {

        const warning = document.createElement('div');

        warning.className = 'flash-warning';

        warning.textContent = `Conteúdo interativo obsoleto (Flash: ${element.source}).`;

        return warning;

    }

    const figure = document.createElement('figure');

    figure.className = 'image-element';

    const img = document.createElement('img');

    img.src = `assets/images/${element.source}`;

    img.alt = element.searchText || element.title || 'Imagem do curso';

    img.title = element.title || '';

    if (element.width > 0) img.style.width = `${element.width}px`;

    if (element.height > 0) img.style.height = `${element.height}px`;

    // Quando a imagem terminar de carregar, atualize a altura disponível

    img.addEventListener('load', atualizarAlturaDoContainer, { once: true });

    figure.appendChild(img);

    if (element.legend) {

        const figcaption = document.createElement('figcaption');

        figcaption.textContent = element.legend;

        figure.appendChild(figcaption);

    }

    return figure;

}

// --------------------

// --- V Í D E O S  ---

// --------------------

function extrairVideoIDdoYouTube(*url*) {

    if (!url) return null;

    let videoID = '';

    try {

        const urlObj = **new** URL(url);

        if (urlObj.hostname === 'youtu.be') {

            videoID = urlObj.pathname.slice(1);

        } else if (urlObj.hostname.includes('youtube.com')) {

            videoID = urlObj.searchParams.get('v');

            // suporta /embed/VIDEO\_ID

            if (!videoID && urlObj.pathname.includes('/embed/')) {

                videoID = urlObj.pathname.split('/embed/')[1]?.split(/[/?#]/)[0] || '';

            }

        }

    } catch (error) {

        console.error("URL de vídeo inválida:", url, error);

        return null;

    }

    return videoID || null;

}

function isYouTubeUrl(*url*) {

    try {

        const u = **new** URL(url);

        return u.hostname.includes('youtube.com') || u.hostname === 'youtu.be';

    } catch {

        return false;

    }

}

function isDirectMediaUrl(*url*) {

    return /\.(mp4|webm|ogv|ogg)$/i.test(url || '');

}

// Carrega a YouTube IFrame API uma única vez e resolve quando pronta

function ensureYouTubeAPI() {

    if (window.YT && window.YT.Player) return *Promise*.resolve();

    if (window.\_ytApiReadyPromise) return window.\_ytApiReadyPromise;

    window.\_ytApiReadyPromise = **new** *Promise*((*resolve*) => {

        window.onYouTubeIframeAPIReady = () => resolve();

    });

    const tag = document.createElement('script');

    tag.src = 'https://www.youtube.com/iframe\_api';

    document.head.appendChild(tag);

    return window.\_ytApiReadyPromise;

}

// URL de embed específica para uso com API (controls=0 para UI própria)

function buildYouTubeEmbedUrlForAPI(*videoID*) {

    const params = **new** URLSearchParams({

        autoplay: '1',

        mute: '1',            // autoplay confiável

        controls: '0',        // sem UI nativa

        rel: '0',

        modestbranding: '1',

        iv\_load\_policy: '3',

        playsinline: '1',

        enablejsapi: '1',

        origin: location.origin

    });

    return `https://www.youtube.com/embed/${videoID}?${params.toString()}`;

}

// Cria o bloco de controles próprios (HTML)

function criarControlesUI() {

    const controls = document.createElement('div');

    controls.className = 'yt-controls';

    controls.innerHTML = `

      <button type="button" class="yt-btn yt-play" aria-label="Reproduzir/Pausar">▶</button>

      <button type="button" class="yt-btn yt-mute" aria-label="Ativar/Desativar som">🔇</button>

      <input class="yt-progress" type="range" min="0" max="100" value="0" step="0.1" aria-label="Progresso do vídeo" />

      <span class="yt-time" aria-live="off">00:00 / 00:00</span>

      <button type="button" class="yt-btn yt-full" aria-label="Tela cheia">⛶</button>

    `;

    return controls;

}

// Formata mm:ss

function fmtTime(*sec*) {

    sec = Math.max(0, Math.floor(sec || 0));

    const m = String(Math.floor(sec / 60)).padStart(2, '0');

    const s = String(sec % 60).padStart(2, '0');

    return `${m}:${s}`;

}

// Player HTML5 para mídia direta (fallback simples com controles nativos)

function criarVideoHTML5Direto({ *src*, *poster*, *title* }) {

  // Contêiner externo com altura natural

  const shell = document.createElement('div');

  shell.className = 'video-shell';

  // Caixa de proporção 16:9

  const inner = document.createElement('div');

  inner.className = 'video-inner';

  shell.appendChild(inner);

  const video = document.createElement('video');

  video.setAttribute('controls', 'controls');

  video.setAttribute('preload', 'metadata');

  if (poster) video.setAttribute('poster', `assets/images/${poster}`);

  if (title)  video.setAttribute('title', title);

  const source = document.createElement('source');

  source.src = src;

  if (/\.mp4$/i.test(src)) source.type = 'video/mp4';

  else if (/\.webm$/i.test(src)) source.type = 'video/webm';

  else if (/\.ogv$|\.ogg$/i.test(src)) source.type = 'video/ogg';

  video.appendChild(source);

  inner.appendChild(video);

  // Atualiza a altura do container após inserir o player

  requestAnimationFrame(() => atualizarAlturaDoContainer());

  return shell;

}

function criarVideo(*element*) {

    const url = element.video || '';

    const isYT = isYouTubeUrl(url);

    const videoID = isYT ? extrairVideoIDdoYouTube(url) : null;

    // Container/placeholder inicial com preview + botão play

    const containerVideo = document.createElement('div');

    containerVideo.className = 'video-container-lazy';

    if (element.previewImage) {

        const previewImg = document.createElement('img');

        previewImg.src = `assets/images/${element.previewImage}`;

        previewImg.alt = element.title || 'Pré-visualização do vídeo';

        // Quando a prévia carregar, atualize a altura (afeta layout)

        previewImg.addEventListener('load', atualizarAlturaDoContainer, { once: true });

        containerVideo.appendChild(previewImg);

    }

    const playButton = document.createElement('div');

    playButton.className = 'play-button-overlay';

    containerVideo.appendChild(playButton);

    // Clique: decide estratégia

    containerVideo.addEventListener('click', async () => {

        // 1) Mídia direta → <video> nativo

        if (isDirectMediaUrl(url)) {

            const player = criarVideoHTML5Direto({

                src: url,

                poster: element.previewImage || null,

                title: element.title || ''

            });

            containerVideo.replaceWith(player);

            requestAnimationFrame(() => atualizarAlturaDoContainer());

            return;

        }

        // 2) YouTube com controles próprios via IFrame API

        if (isYT && videoID) {

            await ensureYouTubeAPI();

            // Contêiner externo (altura natural)

            const shell = document.createElement('div');

            shell.className = 'video-shell';

            // Caixa 16:9 apenas para o iframe

            const inner = document.createElement('div');

            inner.className = 'video-inner';

            shell.appendChild(inner);

            // Host do iframe (a API injeta aqui)

            const playerHost = document.createElement('div');

            const hostId = `yt-player-${videoID}-${Math.random().toString(36).slice(2)}`;

            playerHost.id = hostId;

            inner.appendChild(playerHost);

            // Controles próprios (fora da caixa 16:9)

            const controls = criarControlesUI();

            shell.appendChild(controls);

            // Troca o placeholder

            containerVideo.replaceWith(shell);

            requestAnimationFrame(() => atualizarAlturaDoContainer());

            let duration = 0;

            let progressTimer = null;

            const playBtn  = controls.querySelector('.yt-play');

            const muteBtn  = controls.querySelector('.yt-mute');

            const fullBtn  = controls.querySelector('.yt-full');

            const progress = controls.querySelector('.yt-progress');

            const timeEl   = controls.querySelector('.yt-time');

            const player = **new** YT.Player(hostId, {

                videoId: videoID,

                playerVars: {

                    autoplay: 1,

                    mute: 1,

                    controls: 0,        // escondemos a UI nativa

                    rel: 0,

                    modestbranding: 1,

                    iv\_load\_policy: 3,

                    playsinline: 1,

                    origin: location.origin

                },

                events: {

                    onReady: () => {

                        duration = player.getDuration() || 0;

                        timeEl.textContent = `${fmtTime(0)} / ${fmtTime(duration)}`;

                        // Atualiza ~10x/seg

                        progressTimer = setInterval(() => {

                            const ct = player.getCurrentTime();

                            if (!isFinite(ct)) return;

                            timeEl.textContent = `${fmtTime(ct)} / ${fmtTime(duration)}`;

                            if (duration > 0) {

                                progress.value = (ct / duration) \* 100;

                            }

                        }, 100);

                        // Recalcula após o player estar pronto

                        requestAnimationFrame(() => atualizarAlturaDoContainer());

                    },

                    onStateChange: (*e*) => {

                        if (e.data === YT.PlayerState.PLAYING) {

                            playBtn.textContent = '⏸';

                        } else if (e.data === YT.PlayerState.PAUSED || e.data === YT.PlayerState.ENDED) {

                            playBtn.textContent = '▶';

                        }

                    }

                }

            });

            // Controles

            playBtn.addEventListener('click', () => {

                const state = player.getPlayerState();

                if (state === YT.PlayerState.PLAYING) player.pauseVideo();

                else player.playVideo();

            });

            muteBtn.addEventListener('click', () => {

                if (player.isMuted()) { player.unMute(); muteBtn.textContent = '🔊'; }

                else { player.mute(); muteBtn.textContent = '🔇'; }

            });

            progress.addEventListener('input', () => {

                if (duration > 0) {

                    const t = (parseFloat(progress.value) / 100) \* duration;

                    player.seekTo(t, true);

                }

            });

            // Fullscreen no shell (wrapper externo)

            fullBtn.addEventListener('click', () => {

                const el = shell;

                if (!document.fullscreenElement) el.requestFullscreen?.();

                else document.exitFullscreen?.();

            });

            // Limpeza do timer quando remover do DOM (ex.: troca de slide)

            const mo = **new** MutationObserver(() => {

                if (!document.body.contains(shell)) {

                    if (progressTimer) clearInterval(progressTimer);

                    mo.disconnect();

                }

            });

            mo.observe(document.body, { childList: true, subtree: true });

            return;

        }

        // 3) Fallback: abre o link em nova aba (p. ex. hospedador externo)

        try {

            window.open(url, '\_blank', 'noopener,noreferrer');

        } catch (e) {

            console.warn('Falha ao abrir o link de vídeo:', e);

        }

    }, { once: true });

    return containerVideo;

}

// --------------------

// ---   TABELA     ---

// --------------------

function criarGrid(*element*) {

    const table = document.createElement('table');

    table.className = 'grid-element';

    const content = element.content || [];

    const isFirstRowHeader = element.isFirstRowHeader || false;

    if (isFirstRowHeader && content.length > 0) {

        const thead = document.createElement('thead');

        const headerRow = document.createElement('tr');

        content[0].forEach((*cellText*, *colIndex*) => {

            const th = document.createElement('th');

            th.textContent = (cellText || '').trim();

            if (colIndex === element.featureColumn) {

                th.classList.add('feature-column');

            }

            headerRow.appendChild(th);

        });

        thead.appendChild(headerRow);

        table.appendChild(thead);

    }

    const tbody = document.createElement('tbody');

    const startIndex = isFirstRowHeader ? 1 : 0;

    for (let i = startIndex; i < content.length; i++) {

        const row = document.createElement('tr');

        if (element.alternateRowColor && (i - startIndex) % 2 !== 0) {

            row.classList.add('alternate-row');

        }

        content[i].forEach((*cellText*, *colIndex*) => {

            const td = document.createElement('td');

            td.textContent = (cellText || '').trim();

            if (colIndex === element.featureColumn) {

                td.classList.add('feature-column');

            }

            row.appendChild(td);

        });

        tbody.appendChild(row);

    }

    table.appendChild(tbody);

    // Atualiza após montar a tabela (caso altura mude)

    requestAnimationFrame(() => atualizarAlturaDoContainer());

    return table;

}

// --------------------

// ---   GRUPO      ---

// --------------------

function criarGrupo(*element*) {

    const div = document.createElement('div');

    div.className = 'group-container';

    div.style.display = 'flex';

    div.style.flexDirection = element.mode === 'horizontalGroup' ? 'row' : 'column';

    const hAlignMap = { left: 'flex-start', center: 'center', right: 'flex-end' };

    const vAlignMap = { top: 'flex-start', middle: 'center', bottom: 'flex-end' };

    div.style.justifyContent = hAlignMap[element.horizontalAlign] || 'flex-start';

    div.style.alignItems = vAlignMap[element.verticalAlign] || 'flex-start';

    if (element.fillHeight) {

        div.style.flexGrow = '1';

    }

    if (element.elements) {

        element.elements.forEach(*child* => {

            const childEl = criarElemento(child);

            if (childEl) div.appendChild(childEl);

        });

    }

    // Atualiza após montar o grupo (altura pode variar)

    requestAnimationFrame(() => atualizarAlturaDoContainer());

    return div;

}

// --------------------

// ---  INFOBOX     ---

// --------------------

function criarGatilhoInfoBox(*element*) {

    const button = document.createElement('button');

    button.className = 'infobox-trigger';

    button.innerHTML = 'i';

    button.title = `Info: ${element.title}`;

    // Garante que o botão posiciona relativo ao container

    button.style.position = 'absolute';

    // 1) Escolhe o elemento âncora

    let anchorEl = null;

    if (INFOBOX\_ANCHOR\_MODE === 'auto-prev') {

        // Pega o último filho "renderizado" antes do infobox

        const children = Array.from(container.children);

        for (let i = children.length - 1; i >= 0; i--) {

            const el = children[i];

            if (!el.classList.contains('infobox-trigger')) { // evita outro infobox

                anchorEl = el;

                break;

            }

        }

    }

    // Se não achou, caímos no modo container

    const useContainerOnly = INFOBOX\_ANCHOR\_MODE === 'container' || !anchorEl;

    // 2) Função para (re)posicionar

    const applyPosition = () => {

        if (!button.isConnected) return;

        const contRect = container.getBoundingClientRect();

        let left = (element.x || 0);

        let top  = (element.y || 0);

        if (!useContainerOnly && anchorEl && anchorEl.isConnected) {

            const aRect = anchorEl.getBoundingClientRect();

            left = (aRect.left - contRect.left) + (element.x || 0);

            top  = (aRect.top  - contRect.top ) + (element.y || 0);

        }

        // CLAMP: garante que o botão caiba no container

        const maxLeft = container.clientWidth  - button.offsetWidth;

        const maxTop  = container.clientHeight - button.offsetHeight;

        left = Math.max(0, Math.min(left, maxLeft));

        top  = Math.max(0, Math.min(top,  maxTop));

        button.style.left = `${left}px`;

        button.style.top  = `${top}px`;

    };

    // 3) Observadores para acompanhar mudanças de layout

    const ro = **new** ResizeObserver(() => {

        applyPosition();

        atualizarAlturaDoContainer();

    });

    ro.observe(container);

    if (anchorEl) ro.observe(anchorEl);

    // Reposiciona quando imagens/iframes do anchor terminam de carregar

    if (anchorEl) {

        anchorEl.querySelectorAll('img, video, iframe').forEach(*m* => {

            const onLoad = () => {

                applyPosition();

                atualizarAlturaDoContainer();

            };

            // imagens

            if ('complete' in m) {

                if (!m.complete) m.addEventListener('load', onLoad, { once: true });

            } else {

                // vídeos/iframes

                m.addEventListener('load', onLoad, { once: true });

            }

            // vídeos em <video> disparam 'loadedmetadata'

            m.addEventListener?.('loadedmetadata', onLoad, { once: true });

        });

    }

    // Também reposiciona no resize da janela

    const onResize = () => {

        if (!button.isConnected) {

            window.removeEventListener('resize', onResize);

            ro.disconnect();

            return;

        }

        applyPosition();

        atualizarAlturaDoContainer();

    };

    window.addEventListener('resize', onResize);

    // Cleanup quando o botão sai do DOM (troca de slide)

    const mo = **new** MutationObserver(() => {

        if (!document.body.contains(button)) {

            window.removeEventListener('resize', onResize);

            ro.disconnect();

            mo.disconnect();

        }

    });

    mo.observe(document.body, { childList: true, subtree: true });

    // 4) Click → abre modal

    button.addEventListener('click', () => abrirModal(element));

    // 5) Posiciona na próxima frame (layout estável)

    requestAnimationFrame(() => {

        applyPosition();

        atualizarAlturaDoContainer();

    });

    return button;

}

// --------------------

// ---  ESPAÇADOR   ---

// --------------------

function criarEspacador(*element*) {

    const spacer = document.createElement('span');

    spacer.className = 'spacer';

    spacer.style.width = `${*element*.width || 0}px`;

    spacer.style.height = `${*element*.height || 0}px`; // Adicionando suporte à altura

    // Atualiza após inserir espaçador (pode alterar a altura do layout)

    requestAnimationFrame(() => atualizarAlturaDoContainer());

    return spacer;

}

// --------------------

// ---  APP LAUNCH  ---

// --------------------

function criarAppLauncher(*element*) {

    const link = document.createElement('a');

    link.className = 'app-launcher-button';

    link.href = *element*.path;

    link.textContent = 'Abrir Atividade Interativa';

    link.target = '\_blank';

    link.rel = 'noopener noreferrer';

    // Mudança de layout mínima; ainda assim atualizamos por consistência

    requestAnimationFrame(() => atualizarAlturaDoContainer());

    return link;

}

**Este é nosso style.css:**

/\* --- 1. RESET BÁSICO E CONFIGURAÇÕES GLOBAIS --- \*/

\* {

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

html, body {

    height: 100%;

    overflow: hidden; /\* Impede a barra de rolagem na página inteira \*/

    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, "Helvetica Neue", Arial, sans-serif;

    background-color: #f0f2f5;

    color: #333;

    line-height: 1.6;

}

/\* --- 2. LAYOUT PRINCIPAL --- \*/

*#app-container* {

    display: grid;

    grid-template-columns: 300px 1fr;

    height: 100vh; /\* Trava a altura do container na altura da janela \*/

    min-width: 0;  /\* Evita overflow horizontal por conteúdo interno \*/

}

/\* --- 3. MENU DE NAVEGAÇÃO (COM CONTROLE DE SCROLL INTERNO) --- \*/

*#menu-navegacao* {

    background-color: #ffffff;

    border-right: 1px solid #ddd;

    display: flex;

    flex-direction: column;

    overflow: hidden;

    min-width: 0; /\* Não permita que o menu force overflow horizontal \*/

}

*#menu-navegacao* h2 {

    padding: 20px;

    margin-bottom: 0;

    font-size: 1.4em;

    color: #0056b3;

    border-bottom: 2px solid #0056b3;

    flex-shrink: 0;

}

*#menu-items* {

    flex-grow: 1;

    overflow-y: auto; /\* Agora isso vai funcionar, pois o pai tem um limite \*/

    padding: 20px;

    min-width: 0;

}

*#menu-items* ul {

    list-style-type: none;

    padding-left: 0;

}

*#menu-items* li {

    margin-bottom: 5px;

}

*.menu-item* {

    display: block;

    padding: 8px 8px 8px 20px;

    text-decoration: none;

    color: #333;

    border-radius: 4px;

    cursor: pointer;

    transition: background-color 0.2s, color 0.2s;

}

*.menu-item:hover* {

    background-color: #e9ecef;

}

*.menu-item.active* {

    background-color: #007bff;

    color: white;

    font-weight: bold;

}

*.menu-group-title* {

    font-weight: bold;

    cursor: pointer;

    padding: 8px;

    position: relative;

    border-radius: 4px;

}

*.menu-group-title:hover* {

    background-color: #f8f9fa;

}

*.menu-group-title::before* {

    content: '▶';

    font-size: 0.8em;

    margin-right: 8px;

    display: inline-block;

    transition: transform 0.2s;

}

*.menu-group-title.expanded::before* {

    transform: rotate(90deg);

}

/\* --- 4. ÁREA DE CONTEÚDO PRINCIPAL (COM CONTROLE DE SCROLL INTERNO) --- \*/

/\* A coluna principal precisa permitir que o meio possa “encolher” para rolar \*/

*#conteudo-principal* {

  display: flex;

  flex-direction: column;

  height: 100vh;              /\* força ocupar a viewport \*/

  overflow: hidden;           /\* impede overflow horizontal/vertical aqui \*/

  min-width: 0;               /\* evita overflow horizontal por filhos com min-width implícito \*/

}

*#slide-header* {

    margin-bottom: 25px;

    padding: 25px 40px 15px 40px;

    border-bottom: 1px solid #ddd;

    flex-shrink: 0;

    background-color: #fff;

}

*#slide-title* { font-size: 2.2em; color: #1a1a1a; }

*#slide-subtitle* { font-size: 1.3em; color: #666; font-style: italic; font-weight: normal; }

/\* O container central deve ser o único a rolar (apenas vertical) \*/

*#slide-elements-container* {

  flex: 1 1 auto;

  min-height: 0;              /\* ESSENCIAL para o flex item poder encolher \*/

  overflow-y: auto;           /\* rolagem vertical \*/

  overflow-x: hidden;         /\* sem rolagem horizontal \*/

  position: relative;         /\* referência para o InfoBox \*/

  padding: 25px 40px;

  min-width: 0;               /\* evita overflow horizontal por filhos \*/

  --sec-h: 60vh;              /\* fallback; JS atualiza em runtime com a altura real \*/

}

/\* Blindagem anti-overflow horizontal para textos/listas \*/

*#slide-elements-container*,

*#slide-elements-container* *.text-element*,

*#slide-elements-container* *.list-element* {

  word-wrap: break-word;

  overflow-wrap: anywhere; /\* quebra inclusive palavras longas/URLs \*/

  min-width: 0;

}

*#slide-footer* {

    padding: 20px 40px;

    border-top: 1px solid #ddd;

    display: flex;

    justify-content: space-between;

    align-items: center;

    gap: 20px;

    background-color: #ffffff;

    flex-shrink: 0;

}

/\* --- ESTILOS PARA COMPONENTES RESPONSIVOS (HAMBÚRGUER E OVERLAY) --- \*/

*#hamburger-btn* {

    display: none; /\* Escondido em telas grandes \*/

    position: fixed;

    top: 15px;

    right: 15px; /\* Mudança de left para right \*/

    width: 40px;

    height: 35px;

    background-color: #007bff;

    border: none;

    border-radius: 5px;

    z-index: 1002; /\* Acima do conteúdo, mas abaixo do menu aberto \*/

    cursor: pointer;

    flex-direction: column;

    justify-content: space-around;

    padding: 8px;

}

*#hamburger-btn* span {

    display: block;

    height: 3px;

    width: 100%;

    background: white;

    border-radius: 3px;

    transition: all 0.3s ease-in-out;

}

/\* Animação para o X de fechar \*/

*#hamburger-btn.active* span*:nth-child*(1) {

    transform: translateY(9px) rotate(45deg);

}

*#hamburger-btn.active* span*:nth-child*(2) {

    opacity: 0;

}

*#hamburger-btn.active* span*:nth-child*(3) {

    transform: translateY(-9px) rotate(-45deg);

}

*#menu-overlay* {

    position: fixed;

    top: 0;

    left: 0;

    width: 100%;

    height: 100%;

    background-color: rgba(0, 0, 0, 0.5);

    z-index: 999; /\* Fica entre o menu e o conteúdo principal \*/

    opacity: 0;

    transition: opacity 0.3s ease-in-out;

}

*#menu-overlay:not*(*.hidden*) {

    opacity: 1;

}

*.hidden* {

    display: none;

}

*#audio-player-container* { flex-grow: 1; min-width: 0; }

*#audio-player* { width: 100%; }

*#controles-navegacao* button {

    padding: 10px 20px;

    border: none;

    background-color: #007bff;

    color: white;

    border-radius: 5px;

    cursor: pointer;

    font-size: 1em;

    transition: background-color 0.2s;

}

*#controles-navegacao* button*:hover* { background-color: #0056b3; }

*#controles-navegacao* button*:disabled* { background-color: #ccc; cursor: not-allowed; }

/\* --- 5. ESTILOS PARA ELEMENTOS DO SLIDE --- \*/

/\* TextElement \*/

*.text-element* { display: block; margin-bottom: 15px; }

*.text-element.title1* { font-size: 1.8em; font-weight: bold; color: #333; }

*.text-element.title2* { font-size: 1.5em; font-weight: bold; color: #0056b3; }

*.text-element.paragraph*, *.text-element.unorderedList* { font-size: 1em; color: #444; }

/\* ListElement \*/

*.list-element* { margin-left: 25px; margin-bottom: 15px; }

*.list-element* li { margin-bottom: 8px; }

/\* ImageElement \*/

*.image-element* { margin: 15px auto; text-align: center; max-width: 100%; }

*.image-element* img {

    display: block;

    max-width: 100% !important; /\* impede overflow horizontal \*/

    height: auto !important;

    margin: 0 auto;

    border: 1px solid #eee;

}

*.image-element* figcaption { text-align: center; font-style: italic; color: #666; font-size: 0.9em; margin-top: 8px; }

*.flash-warning* { background-color: #f8d7da; border: 1px solid #f5c6cb; color: #721c24; padding: 15px; border-radius: 5px; text-align: center; }

/\* GroupElement \*/

*.group-container* {

    display: flex;

    gap: 15px;

    margin-bottom: 15px;

    width: 100%;

    flex-wrap: wrap;  /\* permite quebra de linha em grupos horizontais \*/

}

*.horizontal-group* { flex-direction: row; }

*.vertical-group* { flex-direction: column; }

/\* filhos de grupos podem encolher sem causar overflow \*/

*.group-container* > \* {

    min-width: 0;

    max-width: 100%;

}

*.horizontal-group* > \* {

    flex: 1 1 300px; /\* base flex para distribuir melhor sem estourar \*/

}

/\* SpacerElement \*/

*.spacer* {

    display: inline-block;

    flex-shrink: 0;

    max-width: 100%; /\* impede que um spacer maior crie overflow horizontal \*/

}

/\* GridElement \*/

*.grid-element* {

    width: 100%;

    border-collapse: collapse;

    margin-bottom: 15px;

    table-layout: fixed;   /\* evita “dançar” por conteúdo longo \*/

    word-break: break-word;/\* quebra conteúdo longo dentro das células \*/

}

*.grid-element* th, *.grid-element* td { border: 1px solid #ccc; padding: 10px; text-align: left; vertical-align: top; }

*.grid-element* th { background-color: #f2f2f2; font-weight: bold; }

*.grid-element* *.feature-column* { font-weight: bold; }

*.grid-element* *.alternate-row* { background-color: #f9f9f9; }

/\* ===================== \*/

/\* ====   VÍDEOS    ==== \*/

/\* ===================== \*/

/\* Capa (placeholder) E player compartilham a MESMA geometria:

   largura clampada pela altura disponível do container (var(--sec-h)) em 16:9 \*/

*.video-container-lazy*,

*.video-shell* {

  width: min(100%, calc(var(--sec-h) \* (16 / 9)));

  margin: 15px 0;

}

/\* Capa do vídeo (16:9 com imagem preenchendo a janela) \*/

*.video-container-lazy* {

  position: relative;

  aspect-ratio: 16 / 9;

  border: 1px solid #ccc;

  background: #000;

}

*.video-container-lazy* img {

  position: absolute;

  inset: 0;

  width: 100%;

  height: 100%;

  display: block;

  object-fit: cover;   /\* use 'contain' se não quiser recorte \*/

  opacity: 0.8;

}

*.video-container-lazy:hover* *.play-button-overlay* { background: rgba(0, 0, 0, 0.7); }

*.play-button-overlay* {

  position: absolute; top: 50%; left: 50%;

  transform: translate(-50%, -50%);

  width: 80px; height: 80px;

  background: rgba(0, 0, 0, 0.5);

  border-radius: 50%;

  transition: background-color 0.2s;

}

*.play-button-overlay::after* {

  content: '';

  position: absolute; top: 50%; left: 52%;

  transform: translate(-50%, -50%);

  border-style: solid; border-width: 15px 0 15px 25px;

  border-color: transparent transparent transparent white;

}

/\* Player (YouTube/HTML5) com a mesma janela 16:9 \*/

*.video-inner* {

  position: relative;     /\* referência para o filho absoluto \*/

  aspect-ratio: 16 / 9;   /\* define a altura a partir da largura \*/

  width: 100%;

  height: auto;           /\* substitui qualquer height:0 prévio \*/

  padding-bottom: 0;      /\* neutraliza técnica antiga (56.25%) \*/

  min-height: 180px;      /\* proteção extra contra altura zero \*/

  background: #000;

  border: 1px solid #ccc;

}

*.video-inner* iframe,

*.video-inner* video {

  position: absolute;

  inset: 0;               /\* top/right/bottom/left: 0 \*/

  width: 100%;

  height: 100%;

  display: block;

}

/\* Controles próprios (UI) \*/

*.yt-controls*{display:flex;align-items:center;gap:10px;padding:8px 0}

*.yt-btn*{border:none;border-radius:6px;padding:6px 10px;cursor:pointer;background:#007bff;color:#fff;font-size:14px}

*.yt-progress*{flex:1}

/\* Em telas muito largas, limite mais pelo viewport para conforto visual \*/

@media (min-width: 1280px) {

*.video-container-lazy*,

*.video-shell* {

    width: min(100%, calc(70vh \* (16 / 9)));

  }

}

/\* --- 6. RESPONSIVIDADE --- \*/

@media (max-width: 900px) {

*#app-container* {

        grid-template-columns: 1fr; /\* Coluna única \*/

    }

*#hamburger-btn* {

        display: flex; /\* Mostra o botão hambúrguer \*/

    }

*#menu-navegacao* {

        position: fixed; /\* Tira o menu do fluxo normal do grid \*/

        top: 0;

        left: 0;

        width: 300px; /\* Largura do menu quando aberto \*/

        height: 100vh;

        transform: translateX(-100%); /\* Começa fora da tela, à esquerda \*/

        transition: transform 0.3s ease-in-out;

        z-index: 1001; /\* Fica acima do overlay \*/

        box-shadow: 2px 0 15px rgba(0,0,0,0.2);

    }

*#menu-navegacao.open* {

        transform: translateX(0); /\* Move o menu para dentro da tela \*/

    }

*#conteudo-principal* {

        /\* O conteúdo principal já ocupa a tela toda, não precisa de mais mudanças \*/

    }

}

/\* ========================= \*/

/\* ===  INFOBOX & MODAL  === \*/

/\* ========================= \*/

*.infobox-trigger* {

    position: absolute;

    width: 32px; height: 32px;

    background-color: #ffc107; border: 1px solid #d39e00;

    border-radius: 50%; cursor: pointer;

    font-size: 1.2em; font-weight: bold;

    font-family: 'Courier New', Courier, monospace; color: #333;

    display: flex; justify-content: center; align-items: center;

    transition: transform 0.2s, box-shadow 0.2s;

    z-index: 10;

    max-width: 100%; overflow: hidden; /\* não cria overflow horizontal \*/

}

*.infobox-trigger:hover* { transform: scale(1.1); box-shadow: 0 0 10px rgba(0,0,0,0.2); }

*.modal-hidden* { display: none !important; }

*#modal-overlay* {

    position: fixed; top: 0; left: 0; width: 100%; height: 100%;

    background-color: rgba(0, 0, 0, 0.6);

    display: flex; justify-content: center; align-items: center;

    z-index: 1000; padding: 20px;

}

*#modal-box* {

    background-color: #fff; border-radius: 8px;

    box-shadow: 0 5px 15px rgba(0,0,0,0.3);

    width: 100%; max-width: 800px; max-height: 90vh;

    display: flex; flex-direction: column;

}

*#modal-header* { display: flex; justify-content: space-between; align-items: center; padding: 15px; border-bottom: 1px solid #eee; }

*#modal-title* { font-size: 1.2em; color: #333; }

*#modal-close-btn* { background: none; border: none; font-size: 2em; line-height: 1; cursor: pointer; color: #888; }

*#modal-content* { padding: 20px; overflow-y: auto; }

/\* AppLauncherElement \*/

*.app-launcher-button* {

    display: inline-block; padding: 12px 25px;

    background-color: #28a745; color: white; text-decoration: none;

    border-radius: 5px; font-weight: bold; text-align: center;

    margin: 10px 0; transition: background-color 0.2s;

    max-width: 100%; /\* link não estoura linha \*/

    word-break: break-word;

}

*.app-launcher-button:hover* { background-color: #218838; }