Integrating EmaModal into a WordPress Website

This guide explains how to integrate the `EmaModal` React component into a WordPress theme. Since `EmaModal` is built with React and Tailwind CSS, and WordPress is a PHP-based CMS, we need a special setup to make them work together.

We'll cover two primary methods:

1. **The Quick Method**: A simplified version using vanilla JavaScript that mimics the core functionality. This is easier to implement but misses some of the advanced features and animations.
2. **The Full React Integration**: The recommended method to use the actual `EmaModal` component with its full capabilities. This requires setting up a JavaScript build process.

Prerequisites

- Access to your WordPress theme's files (`functions.php`, `footer.php`, etc.).
- Node.js and a package manager (`npm`, `pnpm`, or `yarn`) installed on your local development machine.
- Basic knowledge of WordPress theme structure and JavaScript.

Method 1: Quick & Easy (Vanilla JavaScript)

This method recreates the essential functionality of the EmaModal without using React. It's faster to set up but less feature-rich.

Step 1: Add Tailwind CSS to your Theme

For styling, you can add the Tailwind CSS Play CDN script to your theme's header. This is great for development and simple sites. Open your theme's `header.php` file and add this line inside the `<head>` tag:

```
<script src="https://cdn.tailwindcss.com"></script>
```

> **Note:** For a production website, it is highly recommended to set up Tailwind CSS as part of a build process to purge unused styles and optimize performance.

Step 2: Add the Modal HTML to your Footer

Open your theme's `footer.php` and add the following HTML structure just before the closing `</body>` tag. Replace `YOUR_EMA_URL_HERE` with your actual EMA instance URL.

```
<!-- EmaModal Placeholder -->
<div id="ema-modal-container">
    <!-- Floating Button -->
    <div id="ema-floating-button" class="fixed bottom-5 right-5 z-50 cursor-poin
        <div class="flex items-center gap-3">
            <div class="bg-white rounded-full px-4 py-2 shadow-lg border border-
                <p class="text-gray-700 font-medium text-sm md:text-base whitesp
            </div>
            <button class="relative flex h-20 w-20 items-center justify-center r
                 <!-- Make sure the path to the logo is correct for your theme s
                 <img src="<?php echo get_template_directory_uri(); ?>/assets/im
            </button>
        </div>
    </div>

    <!-- Full-Screen Modal -->
    <div id="ema-modal-overlay" class="fixed inset-0 z-[9998] bg-black/10 backdr
    <div id="ema-modal-content" class="fixed inset-0 z-[9999] bg-white flex flex
        <!-- Close Button -->
        <button id="ema-close-button" class="absolute top-4 right-4 z-20 h-10 w-
            <svg class="h-4 w-4" xmlns="http://www.w3.org/2000/svg" viewBox="0 0
        </button>

        <!-- Loading Spinner -->
        <div id="ema-loader" class="absolute inset-0 flex flex-col items-center
             <svg class="h-12 w-12 animate-spin mb-4 text-[#0066ff]" xmlns="http
                <circle class="opacity-25" cx="12" cy="12" r="10" stroke="curren
                <path class="opacity-75" fill="currentColor" d="M4 12a8 8 0 018-
             </svg>
            <p class="text-gray-600 text-lg font-medium">EMA yükleniyor...</p>
        </div>

        <!-- Iframe -->
        <iframe
            id="ema-iframe"
            src=""
            data-src="YOUR_EMA_URL_HERE"
            title="EMA Assistant"
            class="w-full h-full border-none flex-1 opacity-0 transition-opacity
            allow="microphone; speech-recognition; camera"
```

```
        ></iframe>
    </div>
</div>
<!-- End EmaModal -->
```

Step 3: Add the JavaScript Logic

1. Create a new file in your theme's directory, for example: `/js/ema-modal.js`.
2. Add the following JavaScript code to it. This code handles opening/closing the modal and loading the iframe.

```
document.addEventListener('DOMContentLoaded', function () {
    const openButton = document.getElementById('ema-floating-button');
    const closeButton = document.getElementById('ema-close-button');
    const overlay = document.getElementById('ema-modal-overlay');
    const modalContent = document.getElementById('ema-modal-content');
    const iframe = document.getElementById('ema-iframe');
    const loader = document.getElementById('ema-loader');

    let isIframeLoaded = false;
    let isModalOpen = false;

    function openModal() {
        if (isModalOpen) return;
        isModalOpen = true;

        openButton.style.transform = 'scale(0)';
        openButton.style.opacity = '0';

        overlay.classList.remove('hidden');

        // Trigger reflow to apply CSS transitions
        void overlay.offsetWidth;
        void modalContent.offsetWidth;

        overlay.classList.add('opacity-100');
        modalContent.classList.remove('scale-0', 'opacity-0');
        modalContent.classList.add('scale-100', 'opacity-100');

        document.body.style.overflow = 'hidden';

        // Load iframe only on first open
        if (!iframe.src) {
            iframe.src = iframe.dataset.src;
            iframe.onload = () => {
```

```
                isIframeLoaded = true;
                loader.style.display = 'none';
                iframe.classList.add('opacity-100');
            };
        }
    }

    function closeModal() {
        if (!isModalOpen) return;
        isModalOpen = false;

        overlay.classList.remove('opacity-100');
        modalContent.classList.remove('scale-100', 'opacity-100');
        modalContent.classList.add('scale-0', 'opacity-0');

        setTimeout(() => {
            overlay.classList.add('hidden');
            openButton.style.transform = 'scale(1)';
            openButton.style.opacity = '1';
            document.body.style.overflow = '';
        }, 500); // Match the transition duration
    }

    openButton.addEventListener('click', openModal);
    closeButton.addEventListener('click', closeModal);
    overlay.addEventListener('click', closeModal);

    document.addEventListener('keydown', (e) => {
        if (e.key === 'Escape' && isModalOpen) {
            closeModal();
        }
    });
});
```

Step 4: Enqueue the Script in WordPress

Finally, open your theme's `functions.php` file and add the following PHP code to load your new JavaScript file correctly.

```
function theme_enqueue_ema_modal_scripts() {
    // Enqueue the script
    wp_enqueue_script(
        'ema-modal-script',
        get_template_directory_uri() . '/js/ema-modal.js',
        array(),
        '1.0.0',
```

```
        true // Load in footer
    );
}
add_action( 'wp_enqueue_scripts', 'theme_enqueue_ema_modal_scripts' );
```

Method 2: Full React Integration

This method is more complex but allows you to use the original `EmaModal.tsx` component within WordPress. This is the recommended path if you need the full functionality and plan to maintain the component.

Step 1: Set Up a React Build Environment in Your Theme

1. Navigate to your WordPress theme's directory in your terminal: `cd wp-content/themes/your-theme-name`.
2. Create a sub-directory to house your React app, e.g., `mkdir react-src && cd react-src`.
3. Initialize a new Node.js project: `pnpm init`.
4. Install React, ReactDOM, and a build tool like Vite: `bash pnpm add react react-dom pnpm add -D vite @vitejs/plugin-react`
5. Create a `vite.config.js` file in the `react-src` directory to configure the build output. We need it to output a single JS and CSS file that we can enqueue in WordPress.

```
// react-src/vite.config.js
import react from '@vitejs/plugin-react';

import { defineConfig } from 'vite';

export default defineConfig({
    plugins: [react()],
    build: {
        outDir: '../assets/dist', // Output to theme's assets/dist folder
        assetsDir: '',
        rollupOptions: {
            output: {
                entryFileNames: 'ema-modal.js',
                assetFileNames: 'ema-modal.css'
            }
        }
```

```
        }
    });
```

Step 2: Adapt the EmaModal Component

1. Copy the `EmaModal.tsx` file and its dependencies (like UI components, hooks) into your `react-src` folder. Let's assume you place `EmaModal.tsx` in `react-src/components/`.
2. **Replace Next.js specific features**:
   - Change `next/image` to a standard `<img>` tag.
   - Remove any other Next.js APIs if present.
3. Ensure all component paths are correct (e.g., `import { Button } from './ui/button'`).

Step 3: Create the React Entry Point

Create an `index.tsx` file in your `react-src` directory. This file will find a root element in the DOM and render your component into it.

```
// react-src/index.tsx
import React from 'react';

import EmaModal from './components/EmaModal';
// Adjust path as needed
import './styles.css';
import ReactDOM from 'react-dom/client';

// Your main CSS file for the React app

// Get the root element from the page
const rootElement = document.getElementById('ema-modal-root');

if (rootElement) {
    // Retrieve props passed from WordPress
    const props = (window as any).emaModalProps || {};

    const root = ReactDOM.createRoot(rootElement);
    root.render(
        <React.StrictMode>
            <EmaModal {...props} />
        </React.StrictMode>
    );
}
```

Step 4: Build and Enqueue the React App

1. **Build the App**: Run `pnpm build` from inside the `react-src` directory. Vite will generate `ema-modal.js` and `ema-modal.css` in your theme's `/assets/dist/` folder.
2. **Add Root Element to Footer**: Open `footer.php` and add `<div id="ema-modal-root"></div>` before the closing `</body>` tag.
3. **Enqueue Scripts and Styles in `functions.php`**:

```php
function theme_enqueue_react_app() {
    $asset_path = get_template_directory() . '/assets/dist/';
    $asset_uri = get_template_directory_uri() . '/assets/dist/';

    // Enqueue the CSS
    wp_enqueue_style(
        'ema-modal-styles',
        $asset_uri . 'ema-modal.css',
        array(),
        filemtime($asset_path . 'ema-modal.css') // Versioning
    );

    // Enqueue the JS
    wp_enqueue_script(
        'ema-modal-script',
        $asset_uri . 'ema-modal.js',
        array(), // Dependencies
        filemtime($asset_path . 'ema-modal.js'),
        true // Load in footer
    );

    // Pass data from PHP to JavaScript
    $modal_props = array(
        'emaUrl'      => 'YOUR_EMA_URL_HERE',
        'title'       => 'EMA - Interactive Assistant',
        'ctaLabel'    => 'Start EMA',
        'themeColor'  => '#0066ff',
        // Add other props from your WP options panel
    );
    wp_localize_script('ema-modal-script', 'emaModalProps', $modal_props);
}
add_action('wp_enqueue_scripts', 'theme_enqueue_react_app');
```

This second method provides a robust, maintainable way to run a full-fledged React application inside your WordPress theme, with data dynamically passed from the WordPress backend.