WordPress

About **~40–45% of all websites** on the internet run on **WordPress** (recent surveys typically place it around **43%**). □□
If you mean only sites using a known CMS, WordPress accounts for about **~60%+** of that segment.

[Some Midjourney Book Covers I created: Link]

- WordPress: A Beginner-Friendly Guide to the World’s Most Popular Website Builder □□
- SpinupWP
  - Goal: fast and stable WordPress on SpinupWP (Bricks + many plugins + large uploads) □
  - XML-RPC in WordPress: what it is □□
- Detailed summary of the video: WordPress 7 and the truth about its AI features □□
- FlyWP
  - NGINX oder OpenLiteSpeed?
- oEmbed
  - oEmbed steuern (in WP)
  - oEmbed Provider sehen (in WP)
  - oEmbed Verwaltungs-Plugin (für WP) [Opus 4.6]
  - oEmbed Plugin | eigene Provider definieren
  - Skill (für KIs): oEmbed Provider Configuration Generator

WordPress: A Beginner-Friendly Guide to the World’s Most Popular Website Builder □□

WordPress is a **content management system (CMS)**—software that lets you **create, edit, and publish a website** without needing to build everything from scratch in code. It started in **2003** as a blogging tool, but it has evolved into a flexible platform used for many kinds of sites: personal blogs, business websites, portfolios, news sites, online communities, and even full e-commerce stores.

A helpful way to think about it is this: **WordPress is like the control panel and engine for a website**. You log in to an admin area, write or upload content, choose a design, and add features—then WordPress handles how the site is stored and displayed to visitors.

---

1) What a “CMS” actually means (in plain language)

A **CMS** is a system that helps you manage website content—similar to how a document editor helps you write and format a report, but for a website.

With WordPress you can typically:

- Create pages (like **Home**, **About**, **Contact**)
- Write posts (blog-style content, often time-based)
- Upload images and media
- Organize content with categories/tags
- Control menus and site navigation
- Manage multiple users (authors, editors, admins)

Instead of manually editing lots of HTML files, you work inside WordPress's dashboard, and it takes care of publishing everything in the right place.

---

2) WordPress.org vs WordPress.com (a common confusion)

These two are related, but not the same:

1. **WordPress.org ("self-hosted WordPress")**
   - Free, open-source WordPress software you install on a web host.
   - You have **maximum control** over design, plugins, and customization.
   - You're responsible for things like hosting, updates, and backups (or you pay a host to help).
2. **WordPress.com (hosted service)**
   - A commercial service that runs WordPress for you.
   - Easier to start, but customization may be more limited depending on your plan.

If someone says "I built my site on WordPress," they could mean either one. When people talk about WordPress as a CMS platform, they often mean **WordPress.org**.

---

3) How WordPress works behind the scenes (without getting too technical)

Even if you never touch code, it helps to know the basic structure:

- **WordPress runs on a web server** (a computer connected to the internet that serves websites).
- It's written primarily in **PHP**.

- It stores content (your posts, pages, settings) in a **database**—commonly **MySQL** or **MariaDB**.

When someone visits your site:

1. Their browser requests a page.
2. WordPress finds the right content in the database.
3. It combines that content with your chosen design.
4. It sends the finished page to the visitor's browser.

So, you can change text, images, or layout in the dashboard, and the site updates without you manually rebuilding pages.

---

4) Themes: controlling how your site looks □□

A **theme** controls the appearance of your WordPress site—layout, typography, colors, and often some built-in design features.

Key ideas:

- You can **switch themes** without rewriting all your content.
- Many themes offer customization options (logos, colors, templates).
- Developers often use a **child theme** to customize safely, so updates don't overwrite changes.

Think of your content as the "words and pictures," and the theme as the "magazine layout" that presents them.

---

5) Plugins: adding features like building blocks □□

A major reason WordPress is so widely used is its **plugin system**.

A **plugin** is an add-on that can extend what your website can do, for example:

- Search engine optimization (SEO)
- Contact forms
- Online stores (e-commerce)
- Security tools
- Performance and caching
- Membership systems
- Photo galleries
- Multilingual support

In other words: **themes are mostly about design; plugins are mostly about functionality** (though there can be overlap).

---

6) The editor: creating pages and posts with “blocks”

Since 2018, WordPress has included a block-based editor called **Gutenberg**.

Instead of one big text box, you build pages using **blocks**, such as:

- Paragraphs and headings
- Images and galleries
- Buttons
- Columns and layouts
- Embeds (YouTube, social posts, etc.)

This makes it easier for beginners to build structured pages without special tools.

---

7) Who uses WordPress—and why it’s so popular

WordPress is one of the most widely used website platforms in the world. People choose it because it’s:

- **Flexible**: works for simple blogs through complex business sites
- **Extensible**: huge ecosystem of themes and plugins

- **Accessible to non-developers**: many tasks are point-and-click
- **Backed by a large community**: tutorials, forums, developers, agencies
- **Open-source**: the core software is free and maintained by community contributors (with support from organizations like the WordPress Foundation)

---

8) Hosting and setup: what you need to run WordPress

To run self-hosted WordPress (WordPress.org), you typically need:

- A **domain name** (e.g., `yourname.com`)
- **Web hosting** (a service that runs your site online)
- A WordPress installation (often offered as “one-click install”)

There are two common hosting styles:

1. **Shared hosting**
   - Cheaper; your site shares server resources with other sites.
2. **Managed WordPress hosting**
   - More expensive; often includes performance tuning, security hardening, backups, and WordPress-specific support.

---

9) Security and maintenance (the parts beginners shouldn’t ignore) □□

WordPress itself is actively maintained, but like any popular software, it can be a target—especially through vulnerable plugins or outdated sites.

Good basic practices include:

- Keep **WordPress core**, **themes**, and **plugins updated**
- Use reputable themes/plugins and remove ones you don’t use

- Use strong passwords and consider **two-factor authentication**
- Have regular **backups**
- Consider a security plugin or managed hosting security features

Most serious WordPress security problems for beginners come from **not updating** or using **poor-quality plugins**.

---

10) When WordPress is a great fit (and when it might not be)

WordPress is a great fit if you want:

- A website you can update yourself (without a developer for every change)
- Lots of design choices and add-on features
- Blogging or content publishing capabilities
- A platform that can grow over time

It may not be ideal if:

- You need a very custom web application with unusual requirements
- You want a “set it and forget it” site with zero maintenance (though managed hosting helps a lot)
- You require extremely strict, locked-down environments where third-party plugins aren’t allowed

---

Quick recap □

- **WordPress is a CMS** for building and managing websites.
- You typically use **themes** for design and **plugins** for features.
- It runs on web hosting, using **PHP + a database**.
- It’s popular because it’s flexible, approachable, and has a huge ecosystem—but it still needs basic maintenance.

If you tell me what kind of site you want to build (blog, business site, portfolio, store, etc.), I can recommend a simple “starter path” (themes/plugins/hosting approach) that fits your goals. □□

SpinupWP

SpinupWP is a cloud-based, managed control panel designed to simplify setting up and managing WordPress sites on your own servers (like DigitalOcean, Vultr, or Linode). It bridges the gap between DIY and managed hosting, handling server optimization, security (SSL, firewalls), and backups while offering full root access.

SpinupWP

Goal: fast and stable WordPress on SpinupWP (Bricks + many plugins + large uploads)

> I use SpinupWP, and it allows you to configure the following settings. Please explain these settings to me and recommend what I should set: I'll be using BricksBuilder and a whole range of other plugins. I also want to upload large files and edit large pages (a lot of content on a single page). My goal is to achieve excellent performance and stability.
>
> Nginx
>
> - Client Max Body Size
> - Client Body Timeout
> - FastCGI Read Timeout
>
> PHP
>
> - Upload Max File Size
> - Post Max Size
> - Memory Limit
> - Max Execution Time
> - Max File Uploads
> - Max Input Vars
> - Max Input Time
> - Worker Management
>   - Static
>   - Max Workers
>   - Max Requests
>   - Dynamic

    - Max Workers
    - Max Requests
    - Start Workers
    - Mind Idle Workers
    - Max Idle Workers
  - On-Demand
    - Max Workers
    - Max Requests
    - Idel Worker Timeout

Goal: fast *and* stable WordPress on SpinupWP (Bricks + many plugins + large uploads)

SpinupWP is essentially letting you tune two layers:

1. **Nginx** (the web server): controls request size limits and how long it will wait for data.
2. **PHP-FPM** (PHP workers): controls how much PHP can consume per request and how many requests can run concurrently.

Below is what each setting does and what I'd set as a strong default for a "performance + stability" Bricks/WordPress site that also needs **large uploads** and **large admin edits**. (You can tighten later once you know your real needs.)

Nginx settings

1) **Client Max Body Size**

- **What it is:** Maximum allowed size of an HTTP request body (uploads, big POST requests, some editor saves).
- **If too low:** Uploads fail with `413 Request Entity Too Large`.
- **Trade-off:** Higher values allow bigger uploads, but also increase abuse potential (someone can try to send huge bodies).

Recommendation

- Set to the **largest file you realistically upload**, with buffer.
  - Typical “safe but flexible”: **256M**
  - If you *truly* upload big media/backups via WP: **512M** (or **1G** if you must)
- If you’re using direct-to-object-storage uploads (S3, etc.), you can keep it lower.

---

2) **Client Body Timeout**

- **What it is:** How long Nginx will wait to receive the request body from the client (e.g., slow upload connection).
- **If too low:** Large uploads over slow connections fail mid-way (timeout).
- **Trade-off:** Too high can keep connections open longer (slightly more exposure to slowloris-style behavior, though Nginx is good at handling this).

Recommendation

- **120s–300s**
  - Start at **300s** if you regularly upload large files from typical home/office connections.

---

3) **FastCGI Read Timeout**

- **What it is:** How long Nginx will wait for a response from PHP (via PHP-FPM).
- **If too low:** Long operations (imports, image processing, complex saves) fail with `504 Gateway Timeout`.
- **Trade-off:** Too high can allow truly “stuck” PHP requests to tie up capacity longer.

Recommendation

- **120s–300s**
  - Start at **300s** if you do imports, big edits, or heavy plugins.
  - For normal sites with no long operations, you’d keep it closer to 60–120s.

PHP (runtime) settings

4) **Upload Max File Size (`upload_max_filesize`)**

- **What it is:** Max size for a *single uploaded file* handled by PHP.
- **If too low:** WP media uploads fail.

Recommendation

- Set based on your real uploads:
  - Common: **256M**
  - Heavy media workflows: **512M**
- Must be **≤ Post Max Size** (below).

---

5) **Post Max Size (`post_max_size`)**

- **What it is:** Max size of the entire POST request (can include multiple files + form fields + editor payload).
- **If too low:** Uploads or large editor saves can fail even if `upload_max_filesize` is high enough.

Recommendation

- Set to **a bit higher** than `upload_max_filesize`:
  - If upload is 256M → post max **300M–320M**
  - If upload is 512M → post max **600M–640M**

---

6) **Memory Limit (`memory_limit`)**

- **What it is:** Max memory PHP can use per request (front-end requests, WP admin, WP-Cron, AJAX, builder operations).
- **If too low:** White screens, fatal errors, “Allowed memory size exhausted”, builder instability.
- **Trade-off:** Higher per-request memory means fewer concurrent requests fit in RAM (important when combined with PHP worker count).

Recommendation (Bricks + plugins + big pages)

- Start at **512M**
- If you see memory-related fatals or heavy Woo/ACF/SEO stacks: **768M** (or **1024M** only if you truly need it and have RAM headroom)

> Stability tip: it's usually better to run *fewer PHP workers with enough memory* than many workers that cause swapping/oom.

---

7) **Max Execution Time (`max_execution_time`)**

- **What it is:** Max CPU time a PHP script is allowed to run.
- **If too low:** Imports, backups, resizing, and some builder operations can fail.
- **Trade-off:** Too high lets inefficient tasks run longer and hold workers.

Recommendation

- **120s–300s**
  - Start at **300s** if you do imports/large operations.
  - If you don't, **120s** is often plenty.

---

8) **Max File Uploads (`max_file_uploads`)**

- **What it is:** Max number of files that can be uploaded in one request.
- **If too low:** Multi-file upload UIs may fail or partially upload.

Recommendation

- **50** (safe default)
- If you never bulk upload huge batches: **20** is fine.

---

9) **Max Input Vars (`max_input_vars`)**

- **What it is:** Limits how many input fields PHP will accept (forms, page builder payloads, mega menus, complex admin screens).
- **If too low:** Settings don't save completely; missing fields; builder content or meta boxes not persisting.
- **Trade-off:** Higher values slightly increase parsing overhead and can be abused (but typically fine behind WP auth).

Recommendation (large builder pages)

- Start at **10,000**
- If you have extremely large pages / many dynamic fields: **20,000**
- Avoid absurdly high numbers unless you need them.

---

10) **Max Input Time (`max_input_time`)**

- **What it is:** Time limit for PHP to parse input data (POST/GET/file uploads).
- **If too low:** Large POST payloads (big editor saves) can fail.

Recommendation

- **120s–300s**
  - Start at **300s** for your use case.

PHP-FPM Worker Management (this is the big performance/stability lever)

PHP-FPM runs a pool of PHP “workers.” Each concurrent uncached request generally needs a worker. More workers = more concurrency, but also more RAM usage. Too many workers = memory pressure → swapping → slow site → crashes.

Choosing the mode: **Static vs Dynamic vs On-Demand**

1. **Static**
   - Always keeps exactly `Max Workers` running.
   - Best for predictable, steady traffic and lowest latency.
   - Can waste RAM on quiet sites.
2. **Dynamic** *(usually best general-purpose choice)* □□
   - Keeps a baseline and scales up/down with demand.
   - Great balance of responsiveness and RAM efficiency.
3. **On-Demand**
   - Starts workers only when needed; kills them after idle timeout.

- Efficient for very low traffic.
- Slight latency spikes as workers spawn; can feel worse in WP admin if constantly cold-starting.

Recommendation

- Use **Dynamic** for “excellent performance + stability” on a typical production WP server.

Worker parameters (what they mean + what to set)

A) **Max Workers (`pm.max_children`)**

- **What it is:** Maximum number of PHP requests that can run at the same time.
- **If too low:** Requests queue → slow TTFB under load, admin feels laggy when multiple users.
- **If too high:** RAM exhaustion → swapping/oom → site instability.

How to size it (rule of thumb)

- Estimate average memory per PHP worker under your workload:
  - many WP sites: **80–200MB per worker**
  - heavy builders/admin ops can spike higher
- Then ensure: `max_workers × (avg worker RAM)` fits comfortably in available RAM **after** OS + Nginx + DB + caches.

Recommendation (practical starting points)

- If your server is **2GB RAM**: **4–6 workers**
- **4GB RAM**: **6–12 workers**
- **8GB RAM**: **12–24 workers**
- If you set `memory_limit=512M`, do **not** assume each worker uses 512M—but plan conservatively.

B) **Max Requests (`pm.max_requests`)**

- **What it is:** How many requests a worker handles before it’s recycled.
- **Why it matters:** Mitigates memory leaks/fragmentation from plugins/themes over time.

- **Trade-off:** Too low = more respawns (minor overhead).

Recommendation

- **500–1000**
  - Start at **500** if you run lots of plugins and want stability.
  - Use **1000** if everything is stable and you want slightly fewer respawns.

Dynamic mode-specific

C) **Start Workers (`pm.start_servers`)**

- **What it is:** Workers spawned on PHP-FPM start/reload.
- **Recommendation**
- Set to about **25–50% of Max Workers**
  - Example: max 12 → start **4–6**

D) **Min Idle Workers (`pm.min_spare_servers`)**

- **What it is:** Minimum number of idle workers to keep ready.
- **Recommendation**
- About **15–30% of Max Workers**
  - Example: max 12 → min idle **2–4**

E) **Max Idle Workers (`pm.max_spare_servers`)**

- **What it is:** Maximum idle workers to keep around.
- **Recommendation**
- About **50–70% of Max Workers**
  - Example: max 12 → max idle **6–8**

On-Demand mode-specific

F) **Idle Worker Timeout (`pm.process_idle_timeout`)**

- **What it is:** How long an idle worker lives before it’s killed.
- **Recommendation**
- **10s–30s**
  - For WP admin comfort, lean **30s**
  - For ultra-low-traffic efficiency, **10s–15s**

Suggested “strong baseline” config (most Bricks sites)

Use this as a starting point, then adjust based on real usage. □□

Nginx

1. **Client Max Body Size: 512M**
2. **Client Body Timeout: 300s**
3. **FastCGI Read Timeout: 300s**

PHP

1. **Upload Max File Size: 512M**
2. **Post Max Size: 640M**
3. **Memory Limit: 512M** *(go 768M if you hit memory fatals)*
4. **Max Execution Time: 300**
5. **Max File Uploads: 50**
6. **Max Input Vars: 10,000** *(20,000 if large builder pages still don’t save fully)*
7. **Max Input Time: 300**

PHP-FPM (Dynamic)

- **Max Workers:** depends on RAM:
  - **4GB RAM → 10**
  - **8GB RAM → 18**

- **Max Requests: 500**
- **Start Workers:** ~40% of max
- **Min Idle Workers:** ~20% of max
- **Max Idle Workers:** ~60% of max

Concrete examples:

1. **If 4GB RAM**
    1. Max Workers: **10**
    2. Max Requests: **500**
    3. Start Workers: **4**
    4. Min Idle Workers: **2**
    5. Max Idle Workers: **6**
2. **If 8GB RAM**
    1. Max Workers: **18**
    2. Max Requests: **500**
    3. Start Workers: **6**
    4. Min Idle Workers: **3**
    5. Max Idle Workers: **10**

Two quick guardrails (important for “stability”)

1. **Don’t oversubscribe workers**
    - If you see swapping (disk activity high, load average high, site suddenly slow), reduce **Max Workers** first.
2. **Make sure limits agree**
    - `client_max_body_size` (Nginx) must be ≥ `post_max_size` (PHP) must be ≥ `upload_max_filesize` (PHP), otherwise you get confusing failures.

If you tell me these 4 details, I can give a precise worker count ⚙

1. Server RAM (and whether DB is on the same server)
2. CPU cores
3. Traffic level (avg + peak concurrent visitors)

4. Whether you run WooCommerce / membership / heavy search/filter plugins

I'll then recommend an exact **Dynamic** worker configuration sized to your hardware and workload.

SpinupWP

XML-RPC in WordPress: what it is □□

XML-RPC is a WordPress feature (available at `https://your-site.com/xmlrpc.php`) that lets **external apps/services** communicate with your site using a remote procedure call protocol (XML over HTTP).

Common legitimate uses

- **Jetpack** (some features rely on XML-RPC depending on configuration)
- **Mobile/desktop apps** that publish posts remotely
- Some **automation/integration** tools
- Historically: **pingbacks/trackbacks** (these are largely outdated and often abused)

Why people disable it □□

XML-RPC is not "insecure by default," but it's a **frequent target** because:

1. **Brute-force login attempts**
   - Attackers can try many passwords efficiently via `system.multicall`.
2. **DDoS amplification / abuse**
   - Especially via **pingback** mechanisms on sites that allow it.
3. **It's often unnecessary**
   - Many modern setups use the **REST API** instead.

Should you disable it in SpinupWP? □

Recommended default

If you **don’t explicitly need XML-RPC**, then **yes—disable it**. This is a common hardening step and usually reduces attack noise meaningfully.

Keep it enabled if you rely on it

Don’t disable (or disable *partially*) if you use:

- **Jetpack features that require XML-RPC**
- A **mobile publishing app** or legacy integration that depends on `xmlrpc.php`
- Certain third-party services that still use XML-RPC to post/update content

If you’re unsure, check your logs for requests to `/xmlrpc.php` and confirm whether any are coming from known services you use.

What “disabling” should mean (best practice)

Ideally, you want one of these outcomes:

1. **Block XML-RPC entirely**
   Best if you never use it.
2. **Disable only pingbacks** (but keep other methods)
   Useful if you need XML-RPC for publishing/Jetpack but want to stop common abuse.

SpinupWP typically offers server-level controls (e.g., Nginx rules) or guidance to block `xmlrpc.php`. If you tell me how you’re planning to disable it (SpinupWP toggle, custom Nginx snippet, plugin), I can advise the safest approach.

Quick decision checklist □□

- You **use Jetpack** and depend on its remote features → **Don’t disable blindly** (consider disabling pingbacks only).
- You **never publish remotely**, don’t use Jetpack, and don’t need legacy integrations → **Disable it**.

- You’re getting lots of suspicious hits to `/xmlrpc.php` → **Disable or restrict it** (and consider adding rate-limiting/WAF).

If you share whether you use **Jetpack**, any **mobile publishing**, or specific integrations, I’ll recommend the most appropriate SpinupWP configuration.

Detailed summary of the video: WordPress 7 and the truth about its AI features

The video's core message is quite clear:

> **WordPress 7 is *not* a major end-user AI release in the way many people seem to expect.**
> It does **not** ship with a built-in AI website generator or deeply integrated AI content system in core.
> Instead, it mainly lays **foundational groundwork** for future AI integration.

In other words, the presenter is pushing back against the hype. WordPress 7 does include something AI-related in core, but it is **very limited** and mostly infrastructure-oriented rather than feature-rich.

https://youtu.be/MRuKphdka7E

The main takeaway

According to the video, the idea that WordPress 7 is the big "AI-centric" release is **overstated**.

What WordPress 7 actually gives you in core is essentially:

1. A new **Connectors** area in Settings
2. A way to connect supported AI providers via API keys
3. A framework that other plugins can use

What it does **not** give you by default is:

1. AI content generation throughout the editor
2. AI site building in core

3. A “describe your website and let WordPress build it” experience
4. A polished consumer-facing AI workflow built directly into WordPress itself

So the presenter’s verdict is:

- **The hype is ahead of the actual release**
- **The underlying work is real and meaningful**
- **The useful AI features currently depend on an additional plugin, not WordPress core itself**

What the presenter installs and tests

The presenter is running:

1. **WordPress 7 beta 6**
2. The **WordPress Beta Tester** plugin
3. The **2025 theme**

He emphasizes that this is **not** a general review of WordPress 7. The video is narrowly focused on the AI side.

What is actually new in WordPress 7 core

The new Connectors page

In WordPress 7, under **Settings**, there is a new page called **Connectors**.

The presenter shows this page and makes a big point of saying:

> **“This page is basically the main AI-related feature in core.**

Its purpose is to act as a **central location for API/service connections**, especially AI providers.

He explains that the current built-in connectors include providers such as:

1. **Anthropic**
2. **Google**
3. **OpenAI**
    - Although the wording in the transcript around the built-ins is slightly messy, the practical point is that supported AI services can be connected from here.

What Connectors actually do

The Connectors interface does not itself provide AI features. Instead, it:

1. Lets you install the supporting connector/plugin for a provider
2. Prompts you for the provider's API key
3. Stores that connection so other plugins can use it

This is important: the presenter repeatedly says that after connecting a provider, **nothing visibly happens on its own**.

So the Connectors page is best understood as:

- a **shared infrastructure layer**
- a **centralized credential/configuration location**
- a **developer-friendly standard entry point**

rather than a feature users will directly feel as "AI in WordPress."

Why this matters

This centralization solves a practical UX problem.

Before this kind of approach, lots of AI-enabled plugins might each ask the user to:

1. Create an API key
2. Go into that plugin's own settings page
3. Paste the same or similar credentials again

With Connectors, the idea is that plugins can rely on a shared system rather than duplicating setup flows.

That may not sound exciting, but it is actually significant from an architecture perspective. It suggests WordPress is trying to create a **common AI integration standard** instead of letting the

ecosystem remain fragmented.

The real AI features come from the AI plugin, not core

After setting up Connectors, the presenter notices a message indicating that the **AI plugin** can use those connectors for tasks like:

1. Generating featured images
2. Generating alt text
3. Generating titles
4. Generating excerpts
5. And more

That leads to the crucial clarification:

WordPress 7 core is not where the AI functionality lives

To access those features, you must install a separate plugin:

- formerly called **AI Experiments**
- now renamed simply **AI plugin**

The presenter seems pleased that “Experiments” was removed from the name, because that label can sound unstable or risky.

Still, the plugin is presented as a preview of the future, not yet as a fully mature, deeply embedded core feature set.

What the AI plugin adds

Once installed, the AI plugin provides a settings page with various AI-enabled editor features. The presenter refers to these as “editor experiments,” even though the plugin name has changed.

The features shown include:

1. **Excerpt generation**
2. **Alt text generation for images**
3. **Image generation**
4. **Review notes**
5. **Content summarization**
6. **Title generation**
7. **Abilities Explorer**

These features rely on the connector/API setup already configured.

Specific AI features demonstrated in the editor

The presenter opens a post and demonstrates what changes once the AI plugin is active.

1. Generate featured image

A new option appears to create an **AI-generated featured image**.

The presenter is personally skeptical of this feature. He says he generally dislikes AI image generation unless it is done with:

1. A lot of careful prompting
2. Strong creative direction
3. Significant iteration
4. Example-based guidance

So while the feature exists, he is not enthusiastic about its practical value in its current form.

2. Title generation or regeneration

The plugin can analyze the post content and suggest a new title.

The flow appears to be:

1. Click into the title area
2. Use **Regenerate**

3. The plugin sends prompts to the connected AI service
4. It returns a revised title suggestion

The presenter sees this as useful, but somewhat limited. He notes that it would be better if it offered:

1. Multiple alternatives
2. More control
3. More flexibility in how titles are generated

He also points out that this is only for the **post title**, not for headings within the content like H2s or H3s.

3. Excerpt generation

The plugin can generate or regenerate a post excerpt.

The presenter demonstrates this and finds it directionally promising, though he notes the result is:

- somewhat long
- lacking user control over the length

So the idea is good, but the implementation still feels early.

4. Review notes

This is one of the more interesting features in the video.

The AI can go through blocks/sections of the content and generate **review notes**—essentially feedback or suggestions relating to:

1. Grammar
2. Structure
3. Content quality
4. Possibly clarity or readability

The presenter ties this into broader collaboration/note-related functionality coming in WordPress 7.

However, he notes a limitation:

- the tool provides feedback,
- but does not yet offer a smooth “apply this suggestion directly” workflow.

So it is more of an AI-assisted editorial review layer than an AI rewriting assistant.

He finds it genuinely cool, though he questions how essential it is if the content was already heavily created or refined in an external AI tool.

5. Summary generation

The AI can generate a summary block, typically placed near the top of the post.

The presenter sees the usefulness, but immediately notices that the summary can feel too close to the wording of the original content. So again, the feature is promising, but not especially refined yet.

The broader point: this is a preview of where WordPress is going

A recurring theme throughout the video is:

> **These tools matter less as finished features and more as signs of future direction.**

The presenter views the AI plugin as a **demonstration layer** or **prototype path** for what might later become:

1. Core functionality
2. Official opt-in features
3. Standardized AI integrations across the WordPress ecosystem

This is why he keeps saying to “hold tight.” His position is not that WordPress AI efforts are fake or unimportant. Quite the opposite: he believes a lot of real work and investment is going into them. His argument is simply that **WordPress 7 itself is not the final expression of that work.**

The roadmap and upcoming versions

The presenter also references a project board or roadmap for the AI plugin.

He appears to be looking at:

- the current plugin version, which he identifies as **version 6**
- future development toward **0.7**
- longer-term plans toward **1.0**
- additional future-release ideas beyond that

He describes the board as containing stages such as:

1. Backlog
2. In discussion
3. To do
4. In progress
5. Needs review
6. Done

His point here is that there is **substantial ongoing development**, and the currently visible features are only a slice of what is being worked on.

So while the plugin's current capabilities may feel limited, they are part of an active build-out rather than a dead-end experiment.

The "Abilities Explorer" and what it implies

Toward the end, the presenter highlights one of the most strategically interesting aspects: **AI abilities**.

What are abilities?

In the presenter's explanation, abilities are pieces of functionality exposed by core or plugins that can be invoked or connected to by AI systems, agents, or external tooling.

He loosely frames this as something that could tie into:

1. AI agents
2. MCP-style integrations
3. More advanced programmatic AI interactions
4. A richer capability layer than a normal API alone

He even describes it as being somewhat like:

> **the API on steroids-ish**

That is an informal description, but the implication is important: WordPress is not only exploring AI as a content helper, but also as a **capability platform** that external systems could potentially act on.

What the presenter sees

After activating the AI plugin, under **Tools** there is an **Abilities Explorer**.

It shows a total number of available abilities, with some provided by:

1. **Core**
2. **The AI plugin**

The presenter mentions seeing **13 total abilities** in his environment.

He demonstrates invoking one associated with core, which returns local environment information.

The specific mechanics are not deeply explained in the video, but the significance is this:

- WordPress is beginning to define machine-usable capabilities
- that could matter a lot for future automation and agent-based workflows

For developers and technical product people, this may actually be the most consequential part of the entire story.

The presenter’s overall conclusion

His final position can be summarized like this:

What WordPress 7 is *not*

It is **not**:

1. A fully AI-powered WordPress
2. A mature AI site-building platform in core
3. The release where WordPress “goes full AI”
4. An out-of-the-box solution for AI-driven website creation

What WordPress 7 *is*

It **is**:

1. A release that introduces **AI infrastructure**
2. A release that starts standardizing **AI service connections**
3. A release that supports optional AI functionality through the **AI plugin**
4. A signal that WordPress is actively preparing for much deeper AI capabilities later

His tone is basically:

- **temper expectations now**
- but **be optimistic about the direction** □□

Practical interpretation: what this means for different kinds of users

If you are a regular WordPress site owner

You should not expect WordPress 7 alone to suddenly transform your workflow with AI.

Without extra plugins, the AI experience in core sounds minimal. You will mostly just see the new **Connectors** framework.

If you are a content creator or editor

The AI plugin may be worth experimenting with for:

1. Title suggestions
2. Excerpt generation
3. Summaries
4. Review notes
5. Alt text generation

But based on the presenter’s demo, these feel more like **early assistive tools** than polished production features.

If you are a plugin developer

This release may be more meaningful than it looks.

The Connectors system suggests WordPress is creating shared conventions for:

1. Authentication with AI providers
2. Reusable AI integrations
3. Ecosystem-level interoperability

That could reduce duplicated work across plugins and improve UX consistency.

If you are interested in AI agents and automation

The **Abilities Explorer** and the concept of exposed abilities may be the most exciting part of all this. It points toward a future where WordPress can be interacted with more intelligently by external systems, not just by human users clicking around the admin UI.

Additional context that helps make sense of this

A bit of broader perspective may help here.

Many platforms right now are under pressure to appear “AI-first,” so there is often a gap between:

1. **marketing perception**
2. **actual product reality**

This video is basically trying to close that gap for WordPress 7.

What WordPress appears to be doing is fairly sensible:

1. Build the underlying infrastructure first
2. Standardize provider connections
3. Experiment in plugin space
4. Learn what works
5. Later decide what belongs in core

That is actually a very WordPress-style approach. Historically, WordPress often matures ideas in the plugin ecosystem or in adjacent initiatives before locking them into core.

From a product strategy standpoint, that is probably wiser than rushing half-baked AI features into the main platform.

A concise bottom line

If I had to summarize the entire video in one paragraph, it would be this:

> **WordPress 7 is not the big flashy AI release many people have been expecting. Its only real AI-related addition in core is a new Connectors system for managing AI service integrations. The actual AI features—such as title generation, excerpt generation, summaries, review notes, alt text, and image generation—come from a separate AI plugin, not from WordPress core itself. Still, the video argues that this should not be seen as a disappointment so much as the laying of groundwork for a much more capable AI future in WordPress.**

My added assessment

Based on the video and on how WordPress usually evolves, the most important thing here is not the current end-user feature list. It is the fact that WordPress seems to be establishing:

1. **shared AI connection architecture**
2. **plugin-accessible AI infrastructure**
3. **capability exposure through abilities**
4. **an experimentation path outside core**

That is the kind of groundwork that can later enable much more impressive user-facing features.

So the presenter's "bare-bones truth" seems fair:

- **No, WordPress 7 is not the AI revolution release**
- **Yes, meaningful AI groundwork is absolutely happening**
- **And yes, the future direction is genuinely interesting** □□

If you want, I can also turn this into one of the following:

1. a **short executive summary**
2. a **bullet-point key takeaways list**
3. a **clean blog-style recap**
4. a **YouTube chapter breakdown of the video**

FlyWP

:>

FlyWP

NGINX oder OpenLiteSpeed?

Klar — für **WordPress auf FlyWP** ist die Frage **Nginx oder OpenLiteSpeed (OLS)** ziemlich wichtig, weil sie beeinflusst:

- **wie Caching funktioniert**
- **wie einfach das Setup ist**
- **wie kompatibel Plugins und Regeln sind**
- **wie viel Feintuning du später brauchst**
- **wie „standard“ oder „spezialisiert“ dein Stack ist**

Ich erkläre es dir praxisnah für **WordPress**, nicht nur theoretisch.

Kurzfassung vorab

Wenn du einfach eine solide, professionelle Standard-Lösung willst:

Nginx nehmen.

Wenn du besonders stark auf eingebautes Server-Caching mit LiteSpeed Cache setzen willst:

OpenLiteSpeed nehmen.

Meine grobe Empfehlung:

- **Nginx** = besser, wenn du **Stabilität, Verbreitung, Standardisierung, Kompatibilität und saubere Server-Setups** willst
- **OpenLiteSpeed** = besser, wenn du **maximale WordPress-Performance mit möglichst wenig manuellem Cache-Tuning** willst, vor allem mit dem **LiteSpeed Cache Plugin**

Was sind die beiden überhaupt?

Nginx

Nginx ist ein sehr weit verbreiteter Webserver und Reverse Proxy. Er ist im Hosting- und DevOps-Bereich extrem etabliert.

Typische Einsatzbereiche:

- WordPress-Hosting
- Reverse Proxy vor Apps
- Load Balancing
- statische Dateien sehr effizient ausliefern
- SSL-Termination
- High-Traffic-Websites

Nginx ist im professionellen Hosting-Umfeld fast schon ein Standard.

OpenLiteSpeed

OpenLiteSpeed ist die Open-Source-Version von LiteSpeed Web Server. Er ist speziell im WordPress-Bereich bekannt, weil er sehr gut mit dem **LiteSpeed Cache Plugin** zusammenspielt.

Typische Einsatzbereiche:

- WordPress-Hosting
- Performance-optimierte WP-Installationen
- einfaches serverseitiges Full-Page-Caching
- Nutzer, die LiteSpeed-Ökosystem nutzen wollen

OpenLiteSpeed ist beliebt, weil viele sagen:
„Mit LiteSpeed + LSCache bekommt man sehr schnell sehr gute WordPress-Performance.“

Der wichtigste Unterschied für WordPress

Der wichtigste Unterschied ist **nicht nur der Webserver selbst**, sondern **das Caching-Modell**.

Nginx bei WordPress

Mit Nginx läuft WordPress sehr gut — oft exzellent.
Aber: Das Caching ist meistens **nicht automatisch so eng in WordPress integriert** wie bei LiteSpeed/OpenLiteSpeed.

Du nutzt dann typischerweise:

- FastCGI Cache auf Serverebene
- Redis Object Cache
- Cloudflare Cache
- oder WordPress-Plugins wie:
  - WP Rocket
  - FlyingPress
  - W3 Total Cache
  - ggf. kein Full-Page-Cache-Plugin, sondern FlyWP/FastCGI-Setup

Das kann sehr stark sein, aber manchmal etwas technischer.

Vorteil: flexibel, standardisiert
Nachteil: oft etwas mehr Tuning nötig

OpenLiteSpeed bei WordPress

OpenLiteSpeed ist besonders stark, wenn du das **LiteSpeed Cache Plugin** verwendest.

Dann hast du:

- serverseitiges Full-Page-Caching
- Cache-Purging direkt aus WordPress
- ESI-Unterstützung
- Browser Cache
- Bildoptimierung (über Plugin/Ökosystem)
- CSS/JS-Optimierung
- QUIC.cloud-Anbindung optional
- gute WooCommerce-/Membership-/dynamische Seiten-Features

Das ist der Hauptgrund, warum viele OLS/LiteSpeed für WordPress mögen:
Das Zusammenspiel zwischen Webserver und Plugin ist sehr eng.

Vorteil: sehr starke integrierte Performance-Lösung
Nachteil: stärker an LiteSpeed-Ökosystem gebunden

Vergleich im Detail

---

1. Performance

Nginx

Nginx ist sehr schnell und ressourcenschonend.
Für statische Dateien und als Reverse Proxy ist Nginx exzellent.

Mit richtig konfiguriertem:

- PHP-FPM
- FastCGI Cache

- Redis
- OPcache
- Brotli/Gzip
- HTTP/2 oder HTTP/3
- gutem CDN

… ist Nginx für WordPress **extrem schnell**.

Realistisch betrachtet:

Nginx ist **nicht langsam** gegenüber OLS.
In guten Setups ist der Unterschied oft kleiner, als Marketing vermuten lässt.

OpenLiteSpeed

OpenLiteSpeed ist ebenfalls sehr performant und gerade bei WordPress oft beeindruckend schnell, weil:

- das serverseitige Caching sehr direkt funktioniert
- LiteSpeed Cache stark optimiert ist
- viele Caching-/Optimierungsfunktionen direkt zusammenspielen

In der Praxis:

Auf einer typischen WordPress-Seite kann OLS mit LSCache oft **schneller „out of the box“** wirken als ein eher schlichtes Nginx-Setup.

Aber wichtig:

Wenn Nginx gut konfiguriert ist, kann Nginx genauso hervorragend performen.
OLS gewinnt oft eher bei **Komfort + Integration**, nicht automatisch in jedem Fall bei der Rohleistung.

---

2. Caching

Das ist der Kernpunkt.

Nginx-Caching

Bei Nginx hast du verschiedene Wege:

a) FastCGI Cache

- sehr schnell
- serverseitig
- ideal für anonyme Besucher
- aber Konfiguration/Purge/Regeln können komplexer sein

b) Redis Object Cache

- beschleunigt DB-lastige Anfragen
- ersetzt kein Full-Page-Caching
- sehr nützlich für WordPress

c) Plugin-basiertes Caching

z. B. WP Rocket / FlyingPress

- bequem
- aber oft nicht ganz so tief auf Serverebene integriert wie LSCache

d) CDN/Cloudflare

- kann viel Last abfangen
- ergänzt den Servercache

Fazit bei Nginx: sehr stark, aber eher modular aufgebaut.

OpenLiteSpeed-Caching

Mit OpenLiteSpeed ist das **LiteSpeed Cache Plugin** das große Argument.

Das bringt:

- Full-Page-Cache auf Serverebene
- automatisches Purging
- Cache-Regeln pro Seitentyp
- WooCommerce-Ausnahmen
- eingeloggte Nutzer differenzierter behandelbar
- ESI für dynamische Blöcke
- vieles bequem aus dem WP-Backend steuerbar

Fazit bei OLS: das Gesamtpaket ist oft „runder“, wenn du WordPress zentriert denkst.

---

3. Kompatibilität mit WordPress-Plugins

Nginx

Nginx ist extrem kompatibel mit WordPress allgemein.
Fast jedes Hosting-Setup, Managed-Hosting, DevOps-Setup oder moderne Serverarchitektur funktioniert bestens mit Nginx.

Aber:

- manche `.htaccess`-Regeln aus Apache-Welt funktionieren nicht direkt
- man muss Dinge in Nginx-Syntax umsetzen
- Plugin-Anleitungen erwähnen oft Apache-Regeln zuerst

Bei FlyWP ist das aber meist kein großes Problem, weil vieles vorstrukturiert ist.

OpenLiteSpeed

OLS ist für WordPress ebenfalls sehr kompatibel.
Wenn du das LiteSpeed-Ökosystem nutzt, ist es oft sogar besonders angenehm.

Vorteil:

- viele WordPress-Nutzer verwenden LSCache
- Plugin ist sehr mächtig

Möglicher Nachteil:

- manche Spezialsetups, Dokumentationen oder Custom-Rewrite-Anleitungen sind häufiger für Apache oder Nginx dokumentiert
- du bist ein bisschen mehr in einer „LiteSpeed-Welt“

---

4. Bedienung und Administration

Nginx

Nginx wird typischerweise über Konfigurationsdateien verwaltet.
Das ist:

- sehr sauber
- sehr mächtig
- gut automatisierbar
- im DevOps-Bereich beliebt

Aber:

- nicht immer anfängerfreundlich
- Rewrite-Regeln und Spezialfälle brauchen Erfahrung

Wenn FlyWP dir das meiste abnimmt, ist Nginx oft sehr angenehm, weil du die Stärke von Nginx bekommst, ohne alles manuell bauen zu müssen.

OpenLiteSpeed

OLS hat ein eigenes WebAdmin-Panel und ein eigenes Verwaltungsmodell.

Das kann angenehm sein, wenn du:

- lieber GUIs magst
- bestimmte Dinge komfortabel einstellen willst

Aber:

- wer aus Linux-/DevOps-/Nginx-Welt kommt, empfindet es teils als ungewohnter
- Dokumentation und Best Practices sind insgesamt weniger universell als bei Nginx

---

5. Verbreitung und „Industrie-Standard“

Nginx

Nginx ist deutlich verbreiteter im professionellen Hosting-, Agentur-, SaaS- und Infrastruktur-Umfeld.

Das bedeutet:

- mehr Tutorials
- mehr Community-Wissen
- mehr Standard-Deployments
- mehr Erfahrungswerte
- leichter, Admins/Entwickler zu finden, die damit arbeiten können

Wenn du später wächst oder jemand anders den Server übernehmen soll, ist Nginx oft der „sicherere Standard“.

OpenLiteSpeed

OLS ist im Vergleich spezieller.
Es ist nicht exotisch, aber deutlich weniger universell als Nginx.

Das heißt nicht, dass es schlecht ist — nur:

- weniger allgemeine Standardisierung
- mehr Spezialisierung auf LiteSpeed-Use-Cases
- Know-how ist etwas seltener

---

6. .htaccess / Rewrite-Regeln

Das ist ein praktischer Punkt.

Nginx

Nginx nutzt **keine** `.htaccess`.
Alles läuft über die Hauptkonfiguration.

Vorteile:

- schneller
- sauberer
- zentral verwaltet

Nachteil:

- wenn ein Plugin Apache-Regeln erwartet, musst du sie für Nginx umdenken

---

OpenLiteSpeed

LiteSpeed-/OpenLiteSpeed-Umgebungen sind oft Apache-kompatibler bzw. näher an diesem Modell als Nginx.
Das kann bei manchen Anwendungen und Weiterleitungen angenehmer sein.

Für typische WordPress-Seiten ist das aber meist kein riesiger Entscheidungsfaktor, solange FlyWP das Setup gut managed.

---

7. WooCommerce, Membership, dynamische Seiten

Hier wird Caching kritisch.

Nginx

WooCommerce auf Nginx funktioniert gut, aber:

- Warenkorb
- Checkout
- My Account
- eingeloggte Nutzer
- personalisierte Inhalte

… müssen sauber aus dem Cache ausgeschlossen werden.

Das ist machbar und Standard, aber man muss wissen, was man tut oder ein gutes Managed-Setup haben.

OpenLiteSpeed

Hier punktet OLS oft durch die enge Verzahnung mit LSCache:

- Cache-Ausnahmen
- Purging
- dynamische Teile
- eingeloggte Zustände

… sind oft angenehmer zu handhaben.

Deshalb ist OLS/LiteSpeed im WooCommerce-Bereich für manche Nutzer sehr attraktiv.

---

8. Ressourcenverbrauch

Beide sind effizient.

Nginx

- sehr ressourcenschonend

- bewährt bei vielen gleichzeitigen Verbindungen
- hervorragend als Reverse Proxy

OpenLiteSpeed

- ebenfalls performant
- ebenfalls auf Effizienz ausgelegt
- WordPress-spezifisch oft sehr stark

Im Normalfall ist das **nicht der Hauptentscheidungsfaktor**.
Beide sind gut genug, solange der Server nicht winzig dimensioniert oder grob falsch konfiguriert ist.

---

9. Flexibilität außerhalb von WordPress

Nginx

Wenn du neben WordPress später noch willst:

- Node.js-Apps
- Python-Apps
- Reverse Proxies
- APIs
- komplexe Multi-App-Setups
- Container-/Proxy-Strukturen

… dann ist Nginx oft die flexiblere und universellere Wahl.

OpenLiteSpeed

OLS ist nicht auf WordPress beschränkt, aber seine Stärke wird besonders bei WordPress sichtbar.
Für allgemeine Infra-/Proxy-/DevOps-Szenarien ist Nginx meistens der naheliegendere Standard.

10. Lock-in / Ökosystem-Frage

Nginx

Mit Nginx bist du eher in einem offenen, sehr breit genutzten Standard-Ökosystem.
Du kannst Plugins, Redis, CDN, verschiedene Cache-Strategien frei kombinieren.

OpenLiteSpeed

Mit OLS profitierst du stark, wenn du das LiteSpeed-Ökosystem annimmst:

- LiteSpeed Cache
- QUIC.cloud
- serverseitige LiteSpeed-spezifische Features

Das ist nicht zwingend schlecht — aber eben ein stärkeres Ökosystem-Modell.

Ganz konkret für FlyWP

FlyWP fragt dich das nicht ohne Grund, denn beide Wege führen zu unterschiedlichen Hosting-Stilen.

Wenn du bei FlyWP Nginx wählst

Dann bekommst du typischerweise:

- klassischen, professionellen Standard-Stack
- sehr gute Performance
- hohe Kompatibilität
- gutes langfristiges Fundament
- evtl. mehr Fokus auf Redis, FastCGI, Cloudflare, PHP-Tuning

Das ist oft die beste Wahl, wenn du:

- mehrere Projekte hosten willst
- einen sauberen „Standard-Stack“ bevorzugst
- später flexibel bleiben willst
- nicht speziell auf LiteSpeed Cache setzen willst

Wenn du bei FlyWP OpenLiteSpeed wählst

Dann bekommst du typischerweise:

- ein stärker WordPress-zentriertes Performance-Setup
- sehr gute Zusammenarbeit mit LiteSpeed Cache
- oft schnelle Ergebnisse mit weniger manuellem Cache-Gefrickel
- gute Option für Performance-orientierte WP-Seiten

Das ist oft die beste Wahl, wenn du:

- gezielt WordPress hostest
- LiteSpeed Cache nutzen willst
- möglichst viel aus einem integrierten Caching-System holen willst
- keine Scheu vor einem etwas spezielleren Stack hast

Typische Szenarien

Szenario 1: Normale WordPress-Seite / Business-Website / Blog

Empfehlung: Nginx

- reicht völlig aus
- standardisiert
- robust

- mit Redis + gutem Cache sehr schnell

Szenario 2: Performance-Fokus mit wenig manuellem Tuning

Empfehlung: OpenLiteSpeed

- vor allem wenn du LSCache aktiv nutzen willst
- oft sehr gute Ergebnisse direkt im Betrieb

Szenario 3: WooCommerce-Shop

Beides geht.
Leichte Tendenz:

- **OpenLiteSpeed** wenn du LiteSpeed Cache voll nutzen willst
- **Nginx** wenn du lieber auf den etablierten Standard-Stack setzt und weißt, wie du Caching sauber konfigurierst, oder FlyWP das gut abstrahiert

---

Szenario 4: Du willst später vielleicht mehr als WordPress machen

Empfehlung: Nginx

Szenario 5: Du willst möglichst „industry standard“

Empfehlung: Nginx

Wo OpenLiteSpeed oft glänzt

OpenLiteSpeed wird oft dann als „besser“ empfunden, wenn Leute sagen:

- „Ich habe WordPress installiert“
- „LiteSpeed Cache aktiviert“
- „ohne viel Aufwand ist die Seite schnell“

Das ist ein echter Pluspunkt.
Nicht nur Marketing — das Zusammenspiel ist wirklich gut.

Wo Nginx oft gewinnt

Nginx gewinnt oft bei:

- Standardisierung
- Dokumentation
- Team-/Admin-Kompatibilität
- Infrastruktur-Flexibilität
- langfristiger Wartbarkeit
- universellem Einsatz

Es ist oft die „seriöse Default-Wahl“, wenn man keinen speziellen LiteSpeed-Vorteil ausnutzen will.

Missverständnis: „OpenLiteSpeed ist immer schneller als Nginx“

Das ist zu pauschal.

Die Wahrheit ist eher:

- **Schlecht konfiguriertes Nginx** kann langsamer wirken als OLS mit LSCache
- **Gut konfiguriertes Nginx** ist extrem schnell
- **OLS + LSCache** liefert oft schneller gute Ergebnisse mit weniger Aufwand
- Die Performance hängt stark ab von:
  - PHP-Version
  - OPcache
  - Redis/Object Cache
  - Bildoptimierung
  - Theme/Plugins
  - Datenbank
  - CDN
  - TTFB durch App-Logik
  - externen Requests
  - WooCommerce-/Logged-in-Traffic

Der Webserver allein ist selten der einzige Hebel.

Meine ehrliche Empfehlung für dich bei FlyWP

Wenn du mich fragst, **ohne weitere Spezialanforderungen**:

Nimm **Nginx**, wenn:

- du eine **solide, verbreitete, professionelle Standard-Lösung** willst
- du flexibel bleiben willst
- du evtl. später andere Dinge als WordPress machen willst
- du nicht speziell vom LiteSpeed-Ökosystem abhängig sein möchtest

Nimm **OpenLiteSpeed**, wenn:

- dein Fokus **fast nur auf WordPress** liegt
- du **LiteSpeed Cache** bewusst einsetzen willst
- du möglichst unkompliziert starke Cache-Performance möchtest
- du mit einem etwas spezielleren Stack okay bist

Meine Standard-Empfehlung in einem Satz

Für die meisten Nutzer auf FlyWP würde ich eher **Nginx** empfehlen.

Für WordPress-Performance-Enthusiasten, die gezielt **LiteSpeed Cache** nutzen wollen, eher **OpenLiteSpeed**.

Wenn du es ganz einfach haben willst

Wähle **Nginx**, falls du gerade unsicher bist.

Warum?
Weil Nginx in der Regel:

- die sicherere Standardentscheidung ist
- langfristig weniger speziell ist
- mit FlyWP sehr gut funktioniert
- für WordPress absolut stark genug ist

Entscheidungsmatrix

Nginx wählen, wenn dir wichtig ist:

- Standard-Stack
- breite Verbreitung
- Flexibilität
- DevOps-/Admin-Freundlichkeit
- saubere langfristige Wartbarkeit

OpenLiteSpeed wählen, wenn dir wichtig ist:

- LiteSpeed Cache
- WordPress-zentriertes Tuning
- schnelle Cache-Erfolge ohne viel Handarbeit
- besonders starke Server-Cache-Integration

Meine praktische Empfehlung nach Nutzertyp

Anfänger, der „einfach eine gute Lösung“ will:

Nginx

WordPress-Power-User mit Fokus auf Pagespeed:

OpenLiteSpeed

Agentur / mehrere Projekte / zukunftssicher:

Nginx

Einzelne WP-Site mit maximalem LSCache-Nutzen:

OpenLiteSpeed

oEmbed

oEmbed steuern (in WP)

In WordPress kannst du **oEmbed auf 2 Ebenen steuern**:

1. **was WordPress selbst einbetten darf**
2. **wo auf deiner Seite fremde Inhalte eingebettet werden dürfen**

1. Welche Anbieter WordPress per oEmbed akzeptiert

WordPress unterstützt standardmäßig bestimmte Provider, z. B. YouTube, Vimeo usw.

Anzeigen, welche Provider aktiv sind

Per Code kannst du die oEmbed-Providerliste beeinflussen.

Beispiel in `functions.php` oder besser in einem kleinen Plugin:

```
add_action('init', function () {
    global $wp_oembed;

    echo '<pre>';
    print_r($wp_oembed->providers);
    echo '</pre>';
});
```

Das ist nur zum Testen sinnvoll.

---

2. Bestimmte oEmbed-Provider entfernen

Wenn du z. B. verhindern willst, dass Inhalte von Twitter/X oder Vimeo automatisch eingebettet werden:

```
add_action('init', function () {
    wp_oembed_remove_provider('#https?://(www\.)?twitter\.com/.+#i');
    wp_oembed_remove_provider('#https?://(www\.)?x\.com/.+#i');
    wp_oembed_remove_provider('#https?://(www\.)?vimeo\.com/.+#i');
});
```

Je nach WP-Version können die Regex-Muster leicht abweichen.

---

3. Eigene Provider hinzufügen

Wenn du willst, dass WordPress auch URLs eines eigenen Dienstes per oEmbed verarbeitet:

```
add_action('init', function () {
    wp_oembed_add_provider(
        '#https?://example\.com/watch/.+#i',
        'https://example.com/oembed',
        true
    );
});
```

- erstes Argument = URL-Muster
- zweites = oEmbed-Endpunkt
- drittes = Regex ja/nein

---

4. Auto-Embed komplett abschalten

Wenn du nicht willst, dass WordPress nackte URLs automatisch in Embeds umwandelt:

```
remove_filter('the_content', array($GLOBALS['wp_embed'], 'autoembed'), 8);
```

Damit werden einfache URLs im Content nicht mehr automatisch eingebettet.

---

5. REST/oEmbed-Endpunkte deaktivieren

Wenn du verhindern willst, dass deine WordPress-Seite selbst als oEmbed-Provider arbeitet oder oEmbed-Endpunkte bereitstellt:

```
remove_action('rest_api_init', 'wp_oembed_register_route');
add_filter('embed_oembed_discover', '__return_false');
remove_filter('oembed_dataparse', 'wp_filter_oembed_result', 10);
remove_action('wp_head', 'wp_oembed_add_discovery_links');
remove_action('wp_head', 'wp_oembed_add_host_js');
```

Das ist eher sinnvoll, wenn du oEmbed aus Sicherheits-, Datenschutz- oder Performance-Gründen einschränken willst.

---

6. Nur bestimmte Embeds im Editor erlauben

Wenn du gezielt steuern willst, **welche URLs Autoren einfügen dürfen**, kannst du beim Speichern oder Rendern prüfen, ob eine URL auf einer Allowlist steht.

Einfaches Beispiel:

```
add_filter('pre_oembed_result', function ($result, $url, $args) {
    $allowed = [
        'youtube.com',
        'youtu.be',
        'vimeo.com',
```

```
    ];

    $host = wp_parse_url($url, PHP_URL_HOST);
    if (!$host) {
        return false;
    }

    foreach ($allowed as $domain) {
        if (str_contains($host, $domain)) {
            return $result;
        }
    }

    return '<p>Einbettung dieses Anbieters ist nicht erlaubt.</p>';
}, 10, 3);
```

Sauberer wäre eine exakte Host-Prüfung statt `str_contains()`.

---

7. Datenschutz: Embeds nur nach Klick laden

Wenn es dir auch um DSGVO geht, solltest du Embeds nicht sofort laden, sondern erst nach Zustimmung oder Klick. Das betrifft vor allem:

- YouTube
- Vimeo
- Spotify
- X/Twitter
- Instagram

Dafür gibt es:

- Consent-Plugins
- lokale Platzhalterlösungen
- 2-Klick-Lösungen

Praxis: Was du wahrscheinlich meinst

Wenn du fragst *„wie kann ich steuern, was in WordPress per oEmbed eingebunden wird?“*, meinst du meistens eines von diesen drei Zielen:

A) Nur bestimmte Anbieter erlauben

→ Provider entfernen oder Allowlist bauen

B) Automatische Einbettung ganz verhindern

→ `autoembed` entfernen

C) Datenschutz / Performance verbessern

→ oEmbed-Endpunkte und Auto-Discovery deaktivieren, ggf. mit Klicklösung

---

Empfohlene saubere Lösung

Wenn du nur bestimmte Embeds erlauben willst, würde ich so vorgehen:

- Autoembed aktiv lassen
- unerwünschte Provider entfernen
- optional nur Allowlist-Domains erlauben
- für externe Medien Consent/Klicklösung nutzen

oEmbed Provider sehen (in WP)

Ja — am einfachsten direkt per PHP.

1) Alle aktiven oEmbed-Provider ausgeben

Temporär in `functions.php` oder besser in ein kleines Plugin:

```
add_action('admin_init', function () {
    global $wp_oembed;

    echo '<pre>';
    print_r($wp_oembed->providers);
    echo '</pre>';
    exit;
});
```

Dann einmal `/wp-admin/` aufrufen.

Wichtig:

- Das ist nur zum Testen.
- Danach wieder entfernen, sonst wird das Admin-Backend blockiert.

In der Ausgabe siehst du die registrierten URL-Muster und die zugehörigen oEmbed-Endpunkte.

---

2) Prüfen, ob YouTube aktiv ist

WordPress speichert die Provider als Muster. Du kannst gezielt danach suchen:

```
add_action('admin_init', function () {
    global $wp_oembed;

    $found = false;

    foreach ($wp_oembed->providers as $pattern => $data) {
        if (stripos($pattern, 'youtube') !== false || stripos($pattern, 'youtu.be') !== false)
{
            echo '<p><strong>YouTube-Provider aktiv:</strong></p>';
            echo '<pre>';
            print_r([$pattern => $data]);
            echo '</pre>';
            $found = true;
        }
    }

    if (!$found) {
        echo '<p><strong>Kein YouTube-Provider gefunden.</strong></p>';
    }

    exit;
});
```

---

3) Sauberer Check als true/false

Wenn du nur wissen willst, **ob YouTube aktiv ist**, ohne alles auszugeben:

```
function my_is_youtube_oembed_active() {
    global $wp_oembed;

    foreach ($wp_oembed->providers as $pattern => $data) {
        if (stripos($pattern, 'youtube.com') !== false || stripos($pattern, 'youtu.be') !==
false) {
            return true;
        }
    }
```

```
    return false;
}

add_action('admin_notices', function () {
    if (my_is_youtube_oembed_active()) {
        echo '<div class="notice notice-success"><p>YouTube oEmbed ist aktiv.</p></div>';
    } else {
        echo '<div class="notice notice-warning"><p>YouTube oEmbed ist nicht
aktiv.</p></div>';
    }
});
```

---

4) Noch direkter: Test mit echter URL

Praktisch ist auch zu prüfen, ob WordPress eine echte YouTube-URL als Embed erkennt:

```
add_action('admin_init', function () {
    $url = 'https://www.youtube.com/watch?v=dQw4w9WgXcQ';
    $embed = wp_oembed_get($url);

    if ($embed) {
        echo '<p><strong>YouTube funktioniert als oEmbed.</strong></p>';
        echo $embed;
    } else {
        echo '<p><strong>YouTube funktioniert NICHT als oEmbed.</strong></p>';
    }

    exit;
});
```

Das ist oft aussagekräftiger als nur die Providerliste.

---

5) Per WP-CLI, falls du Zugriff hast

Wenn du SSH/WP-CLI hast:

```
wp eval 'global $wp_oembed; print_r($wp_oembed->providers);'
```

Oder nur YouTube filtern:

```
wp eval '
global $wp_oembed;
foreach ($wp_oembed->providers as $pattern => $data) {
    if (stripos($pattern, "youtube") !== false || stripos($pattern, "youtu.be") !== false) {
        print_r([$pattern => $data]);
    }
}
'
```

Empfehlung

Am besten ist meist:

1. **Providerliste ausgeben**
2. **mit `wp_oembed_get()` eine echte URL testen**

Denn ein Provider kann zwar registriert sein, aber der Embed kann trotzdem durch andere Filter/Plugins gestört werden.

oEmbed Verwaltungs-Plugin (für WP) [Opus 4.6]

> Nicht getestet!

Ein umfassendes WordPress-Plugin zur Verwaltung von oEmbed-Einbettungen. Hier ist der Implementierungsplan:

Implementierungsplan

1. **Haupt-Plugin-Datei** erstellen mit Plugin-Header, Aktivierung/Deaktivierung und Laden der Komponenten
2. **Admin-Einstellungsseite** mit Tabs für:
   - Provider-Verwaltung (aktivieren/deaktivieren einzelner oEmbed-Provider)
   - Globale Einstellungen (Cache-Dauer, max. Breite/Höhe, Lazy Loading)
   - Eigene Provider hinzufügen
   - Blockliste für URLs
3. **Frontend-Filter** die oEmbed-Ausgabe modifizieren (Wrapper, responsive Container, Lazy Loading)
4. **CSS & JS Assets** für Admin und Frontend
5. **Uninstall-Routine** zum sauberen Entfernen

Schritt 1: Haupt-Plugin-Datei

Diese Datei registriert das Plugin, lädt alle Komponenten und definiert Aktivierungs-/Deaktivierungs-Hooks.

```
<?php
/**
 * Plugin Name: oEmbed Manager
 * Plugin URI: https://example.com/oembed-manager
```

```
 * Description: Umfassende Verwaltung von oEmbed-Einbettungen – Provider aktivieren/deaktivieren, eigene Provider hinzufügen, Darstellung anpassen, Blockliste und mehr.
 * Version: 1.0.0
 * Author: zcrow
 * Author URI: https://example.com
 * License: GPL-2.0+
 * License URI: https://www.gnu.org/licenses/gpl-2.0.html
 * Text Domain: oembed-manager
 * Domain Path: /languages
 */

if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

define( 'JEPP_OEM_VERSION', '1.0.0' );
define( 'JEPP_OEM_PLUGIN_DIR', plugin_dir_path( __FILE__ ) );
define( 'JEPP_OEM_PLUGIN_URL', plugin_dir_url( __FILE__ ) );
define( 'JEPP_OEM_PLUGIN_BASENAME', plugin_basename( __FILE__ ) );

/**
 * Standardoptionen bei Aktivierung setzen.
 */
function jepp_oem_activate() {
    $defaults = array(
        'disabled_providers'  => array(),
        'custom_providers'    => array(),
        'blocked_urls'        => '',
        'max_width'           => 800,
        'max_height'          => 0,
        'cache_duration'      => 86400,
        'responsive'          => 1,
        'lazy_loading'        => 1,
        'wrapper_class'       => 'jepp-oembed-wrap',
        'disable_in_feed'     => 0,
        'disable_on_frontend' => 0,
        'strip_inline_styles' => 0,
    );
```

```
    if ( false === get_option( 'jepp_oem_settings' ) ) {
        add_option( 'jepp_oem_settings', $defaults );
    }
}
register_activation_hook( __FILE__, 'jepp_oem_activate' );

/**
 * Aufräumen bei Deaktivierung (Cache leeren).
 */
function jepp_oem_deactivate() {
    global $wpdb;
    $wpdb->query(
        "DELETE FROM {$wpdb->postmeta} WHERE meta_key LIKE '_oembed_%'"
    );
    $wpdb->query(
        "DELETE FROM {$wpdb->options} WHERE option_name LIKE '_transient_oembed_%' OR
option_name LIKE '_transient_timeout_oembed_%'"
    );
}
register_deactivation_hook( __FILE__, 'jepp_oem_deactivate' );

// Komponenten laden
require_once JEPP_OEM_PLUGIN_DIR . 'includes/class-jepp-oem-settings.php';
require_once JEPP_OEM_PLUGIN_DIR . 'includes/class-jepp-oem-providers.php';
require_once JEPP_OEM_PLUGIN_DIR . 'includes/class-jepp-oem-frontend.php';

/**
 * Plugin initialisieren.
 */
function jepp_oem_init() {
    load_plugin_textdomain( 'oembed-manager', false, dirname( JEPP_OEM_PLUGIN_BASENAME ) .
'/languages' );

    new Jepp_OEM_Settings();
    new Jepp_OEM_Providers();
    new Jepp_OEM_Frontend();
}
add_action( 'plugins_loaded', 'jepp_oem_init' );

/**
```

```
 * Settings-Link in der Plugin-Liste.
 */
function jepp_oem_plugin_action_links( $links ) {
    $settings_link = '<a href="' . admin_url( 'options-general.php?page=jepp-oembed-manager' )
. '">'
                     . __( 'Einstellungen', 'oembed-manager' ) . '</a>';
    array_unshift( $links, $settings_link );
    return $links;
}
add_filter( 'plugin_action_links_' . JEPP_OEM_PLUGIN_BASENAME, 'jepp_oem_plugin_action_links'
);
```

Schritt 2: Provider-Verwaltung

Diese Klasse kümmert sich um das Deaktivieren vorhandener und Registrieren eigener oEmbed-Provider.

```
<?php
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

/**
 * Verwaltet oEmbed-Provider: deaktivieren, eigene hinzufügen, URLs blockieren.
 */
class Jepp_OEM_Providers {

    /** @var array */
    private $settings;

    public function __construct() {
        $this->settings = get_option( 'jepp_oem_settings', array() );

        // Provider deaktivieren
        add_filter( 'oembed_providers', array( $this, 'filter_providers' ), 99 );

        // Eigene Provider registrieren
        add_action( 'init', array( $this, 'register_custom_providers' ), 20 );
```

```
        // URLs blockieren
        add_filter( 'pre_oembed_result', array( $this, 'block_urls' ), 10, 3 );
    }

    /**
     * Deaktivierte Provider aus der Liste entfernen.
     *
     * @param array $providers
     * @return array
     */
    public function filter_providers( $providers ) {
        $disabled = isset( $this->settings['disabled_providers'] ) ? $this-
>settings['disabled_providers'] : array();

        if ( empty( $disabled ) ) {
            return $providers;
        }

        foreach ( $providers as $pattern => $data ) {
            // $data[0] = endpoint URL, $data[1] = regex flag
            $endpoint = is_array( $data ) ? $data[0] : $data;

            foreach ( $disabled as $disabled_key ) {
                if ( stripos( $pattern, $disabled_key ) !== false || stripos( $endpoint,
$disabled_key ) !== false ) {
                    unset( $providers[ $pattern ] );
                }
            }
        }

        return $providers;
    }

    /**
     * Eigene Provider registrieren.
     */
    public function register_custom_providers() {
        $custom = isset( $this->settings['custom_providers'] ) ? $this-
>settings['custom_providers'] : array();
```

```
        if ( empty( $custom ) ) {
            return;
        }

        foreach ( $custom as $provider ) {
            if ( empty( $provider['pattern'] ) || empty( $provider['endpoint'] ) ) {
                continue;
            }
            $regex = ! empty( $provider['regex'] ) ? true : false;
            wp_oembed_add_provider( $provider['pattern'], $provider['endpoint'], $regex );
        }
    }

    /**
     * Bestimmte URLs blockieren.
     *
     * @param null|string $result
     * @param string      $url
     * @param array       $args
     * @return null|false|string
     */
    public function block_urls( $result, $url, $args ) {
        $blocked_raw = isset( $this->settings['blocked_urls'] ) ? $this-
>settings['blocked_urls'] : '';

        if ( empty( $blocked_raw ) ) {
            return $result;
        }

        $blocked_lines = array_filter( array_map( 'trim', explode( "\n", $blocked_raw ) ) );

        foreach ( $blocked_lines as $blocked ) {
            if ( empty( $blocked ) ) {
                continue;
            }
            // Unterstützung für Wildcards: * → .*
            $pattern = '#' . str_replace( '\*', '.*', preg_quote( $blocked, '#' ) ) . '#i';
            if ( preg_match( $pattern, $url ) ) {
                return false; // Blockiert
            }
```

```
    }

    return $result;
}

/**
 * Gibt alle in WordPress registrierten Standard-Provider zurück.
 *
 * @return array  Array mit Schlüssel = Pattern, Wert = Endpoint
 */
public static function get_default_providers() {
    require_once ABSPATH . WPINC . '/class-wp-oembed.php';
    $oembed    = _wp_oembed_get_object();
    $providers = $oembed->providers;

    $list = array();
    foreach ( $providers as $pattern => $data ) {
        $endpoint = is_array( $data ) ? $data[0] : $data;
        // Lesbaren Namen aus dem Pattern oder der Endpoint-URL extrahieren
        $name = self::extract_provider_name( $pattern, $endpoint );
        $list[ $pattern ] = array(
            'name'     => $name,
            'endpoint' => $endpoint,
            'pattern'  => $pattern,
        );
    }

    // Alphabetisch nach Name sortieren
    uasort( $list, function ( $a, $b ) {
        return strcasecmp( $a['name'], $b['name'] );
    });

    return $list;
}

/**
 * Versucht einen lesbaren Provider-Namen zu extrahieren.
 *
 * @param string $pattern
 * @param string $endpoint
```

```
     * @return string
     */
    private static function extract_provider_name( $pattern, $endpoint ) {
        // Aus Endpoint-URL den Hostnamen holen
        $host = wp_parse_url( $endpoint, PHP_URL_HOST );
        if ( $host ) {
            $host = preg_replace( '#^(www\.|publish\.)#i', '', $host );
            $parts = explode( '.', $host );
            if ( count( $parts ) >= 2 ) {
                return ucfirst( $parts[ count( $parts ) - 2 ] );
            }
            return ucfirst( $host );
        }

        // Fallback: Pattern bereinigen
        $clean = preg_replace( '#https?://|[^a-zA-Z0-9\.]#', '', $pattern );
        $clean = preg_replace( '#^(www\.)#', '', $clean );
        $parts = explode( '.', $clean );
        return ucfirst( $parts[0] ?: 'Unknown' );
    }
}
```

Schritt 3: Admin-Einstellungsseite

Die Settings-Klasse baut eine übersichtliche Admin-Seite mit Tabs.

```
<?php
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

/**
 * Admin-Einstellungsseite für den oEmbed Manager.
 */
class Jepp_OEM_Settings {

    /** @var string */
    private $option_name = 'jepp_oem_settings';
```

```
/** @var array */
private $settings;

public function __construct() {
    $this->settings = get_option( $this->option_name, array() );

    add_action( 'admin_menu', array( $this, 'add_menu_page' ) );
    add_action( 'admin_init', array( $this, 'register_settings' ) );
    add_action( 'admin_enqueue_scripts', array( $this, 'enqueue_assets' ) );
}

/**
 * Menüeintrag unter "Einstellungen".
 */
public function add_menu_page() {
    add_options_page(
        __( 'oEmbed Manager', 'oembed-manager' ),
        __( 'oEmbed Manager', 'oembed-manager' ),
        'manage_options',
        'jepp-oembed-manager',
        array( $this, 'render_page' )
    );
}

/**
 * Admin-Assets laden.
 */
public function enqueue_assets( $hook ) {
    if ( 'settings_page_jepp-oembed-manager' !== $hook ) {
        return;
    }
    wp_enqueue_style(
        'jepp-oem-admin',
        JEPP_OEM_PLUGIN_URL . 'assets/css/admin.css',
        array(),
        JEPP_OEM_VERSION
    );
    wp_enqueue_script(
        'jepp-oem-admin',
```

```
            JEPP_OEM_PLUGIN_URL . 'assets/js/admin.js',
            array( 'jquery' ),
            JEPP_OEM_VERSION,
            true
        );
    }

    /**
     * Settings API registrieren.
     */
    public function register_settings() {
        register_setting( 'jepp_oem_group', $this->option_name, array(
            'type'              => 'array',
            'sanitize_callback' => array( $this, 'sanitize_settings' ),
        ) );
    }

    /**
     * Eingaben bereinigen.
     */
    public function sanitize_settings( $input ) {
        $clean = array();

        // Deaktivierte Provider
        $clean['disabled_providers'] = array();
        if ( ! empty( $input['disabled_providers'] ) && is_array( $input['disabled_providers']
) ) {
            $clean['disabled_providers'] = array_map( 'sanitize_text_field',
$input['disabled_providers'] );
        }

        // Eigene Provider
        $clean['custom_providers'] = array();
        if ( ! empty( $input['custom_providers'] ) && is_array( $input['custom_providers'] ) )
{
            foreach ( $input['custom_providers'] as $cp ) {
                if ( empty( $cp['pattern'] ) || empty( $cp['endpoint'] ) ) {
                    continue;
                }
                $clean['custom_providers'][] = array(
```

```
                    'pattern'  => sanitize_text_field( $cp['pattern'] ),
                    'endpoint' => esc_url_raw( $cp['endpoint'] ),
                    'regex'    => ! empty( $cp['regex'] ) ? 1 : 0,
                );
            }
        }

        // Blockliste
        $clean['blocked_urls'] = '';
        if ( ! empty( $input['blocked_urls'] ) ) {
            $clean['blocked_urls'] = sanitize_textarea_field( $input['blocked_urls'] );
        }

        // Numerische Werte
        $clean['max_width']       = isset( $input['max_width'] ) ? absint( $input['max_width']
) : 800;
        $clean['max_height']      = isset( $input['max_height'] ) ? absint(
$input['max_height'] ) : 0;
        $clean['cache_duration']  = isset( $input['cache_duration'] ) ? absint(
$input['cache_duration'] ) : 86400;

        // Checkboxen
        $clean['responsive']          = ! empty( $input['responsive'] ) ? 1 : 0;
        $clean['lazy_loading']        = ! empty( $input['lazy_loading'] ) ? 1 : 0;
        $clean['disable_in_feed']     = ! empty( $input['disable_in_feed'] ) ? 1 : 0;
        $clean['disable_on_frontend'] = ! empty( $input['disable_on_frontend'] ) ? 1 : 0;
        $clean['strip_inline_styles'] = ! empty( $input['strip_inline_styles'] ) ? 1 : 0;

        // Wrapper-Klasse
        $clean['wrapper_class'] = ! empty( $input['wrapper_class'] )
            ? sanitize_html_class( $input['wrapper_class'] )
            : 'jepp-oembed-wrap';

        // oEmbed-Cache leeren
        $this->flush_oembed_cache();

        return $clean;
    }

    /**
```

```
     * oEmbed-Cache leeren.
     */
    private function flush_oembed_cache() {
        global $wpdb;
        $wpdb->query(
            "DELETE FROM {$wpdb->postmeta} WHERE meta_key LIKE '_oembed_%'"
        );
        $wpdb->query(
            "DELETE FROM {$wpdb->options} WHERE option_name LIKE '_transient_oembed_%' OR
option_name LIKE '_transient_timeout_oembed_%'"
        );
    }

    /**
     * Einstellungsseite rendern.
     */
    public function render_page() {
        if ( ! current_user_can( 'manage_options' ) ) {
            return;
        }

        $active_tab = isset( $_GET['tab'] ) ? sanitize_key( $_GET['tab'] ) : 'providers';
        $tabs = array(
            'providers' => __( 'Provider', 'oembed-manager' ),
            'display'   => __( 'Darstellung', 'oembed-manager' ),
            'custom'    => __( 'Eigene Provider', 'oembed-manager' ),
            'blocklist' => __( 'Blockliste', 'oembed-manager' ),
        );
        ?>
        <div class="wrap jepp-oem-wrap">
            <h1><?php esc_html_e( 'oEmbed Manager', 'oembed-manager' ); ?></h1>

            <nav class="nav-tab-wrapper jepp-oem-tabs">
                <?php foreach ( $tabs as $slug => $label ) : ?>
                    <a href="<?php echo esc_url( add_query_arg( array( 'page' => 'jepp-oembed-
manager', 'tab' => $slug ), admin_url( 'options-general.php' ) ) ); ?>"
                       class="nav-tab <?php echo $active_tab === $slug ? 'nav-tab-active' :
''; ?>">
                        <?php echo esc_html( $label ); ?>
                    </a>
```

```
                <?php endforeach; ?>
            </nav>

            <form method="post" action="options.php">
                <?php
                settings_fields( 'jepp_oem_group' );

                switch ( $active_tab ) {
                    case 'display':
                        $this->render_tab_display();
                        break;
                    case 'custom':
                        $this->render_tab_custom();
                        break;
                    case 'blocklist':
                        $this->render_tab_blocklist();
                        break;
                    default:
                        $this->render_tab_providers();
                        break;
                }

                // Hidden Fields für die anderen Tabs mitgeben, damit sie nicht überschrieben
werden
                $this->render_hidden_fields( $active_tab );

                submit_button( __( 'Einstellungen speichern', 'oembed-manager' ) );
                ?>
            </form>
        </div>
        <?php
    }

    /**
     * Tab: Provider verwalten.
     */
    private function render_tab_providers() {
        $providers = Jepp_OEM_Providers::get_default_providers();
        $disabled  = isset( $this->settings['disabled_providers'] ) ? $this-
>settings['disabled_providers'] : array();
```

```
        ?>
        <div class="jepp-oem-section">
            <h2><?php esc_html_e( 'oEmbed-Provider aktivieren / deaktivieren', 'oembed-
manager' ); ?></h2>
            <p class="description"><?php esc_html_e( 'Deaktivierte Provider werden nicht mehr
für die automatische Einbettung verwendet.', 'oembed-manager' ); ?></p>

            <div class="jepp-oem-provider-actions" style="margin: 10px 0;">
                <button type="button" class="button jepp-oem-select-all"><?php esc_html_e(
'Alle aktivieren', 'oembed-manager' ); ?></button>
                <button type="button" class="button jepp-oem-deselect-all"><?php esc_html_e(
'Alle deaktivieren', 'oembed-manager' ); ?></button>
            </div>

            <table class="widefat jepp-oem-provider-table">
                <thead>
                    <tr>
                        <th style="width:50px;"><?php esc_html_e( 'Aktiv', 'oembed-manager' );
?></th>
                        <th><?php esc_html_e( 'Provider', 'oembed-manager' ); ?></th>
                        <th><?php esc_html_e( 'URL-Pattern', 'oembed-manager' ); ?></th>
                        <th><?php esc_html_e( 'Endpoint', 'oembed-manager' ); ?></th>
                    </tr>
                </thead>
                <tbody>
                    <?php foreach ( $providers as $pattern => $info ) :
                        $key        = md5( $pattern );
                        $is_disabled = in_array( $key, $disabled, true );
                        ?>
                        <tr>
                            <td>
                                <input type="checkbox"
                                       class="jepp-oem-provider-checkbox"
                                       name="<?php echo esc_attr( $this->option_name );
?>[disabled_providers][]"
                                       value="<?php echo esc_attr( $key ); ?>"
                                    <?php checked( false, $is_disabled ); ?>
                                       data-inverted="1"
                                >
                            </td>
```

```
                        <td><strong><?php echo esc_html( $info['name'] ); ?></strong></td>
                        <td><code><?php echo esc_html( $pattern ); ?></code></td>
                        <td><code><?php echo esc_html( $info['endpoint'] ); ?></code></td>
                    </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>
    <?php
}

/**
 * Tab: Darstellungsoptionen.
 */
private function render_tab_display() {
    $s = $this->settings;
    ?>
    <div class="jepp-oem-section">
        <h2><?php esc_html_e( 'Darstellungseinstellungen', 'oembed-manager' ); ?></h2>
        <table class="form-table">
            <tr>
                <th scope="row"><?php esc_html_e( 'Maximale Breite (px)', 'oembed-manager'
); ?></th>
                <td>
                    <input type="number" name="<?php echo esc_attr( $this->option_name );
?>[max_width]"
                           value="<?php echo esc_attr( $s['max_width'] ?? 800 ); ?>"
min="0" step="1" class="small-text">
                    <p class="description"><?php esc_html_e( '0 = keine Beschränkung',
'oembed-manager' ); ?></p>
                </td>
            </tr>
            <tr>
                <th scope="row"><?php esc_html_e( 'Maximale Höhe (px)', 'oembed-manager'
); ?></th>
                <td>
                    <input type="number" name="<?php echo esc_attr( $this->option_name );
?>[max_height]"
                           value="<?php echo esc_attr( $s['max_height'] ?? 0 ); ?>"
min="0" step="1" class="small-text">
```

```
                        <p class="description"><?php esc_html_e( '0 = keine Beschränkung',
'oembed-manager' ); ?></p>
                    </td>
                </tr>
                <tr>
                    <th scope="row"><?php esc_html_e( 'Cache-Dauer (Sekunden)', 'oembed-
manager' ); ?></th>
                    <td>
                        <input type="number" name="<?php echo esc_attr( $this->option_name );
?>[cache_duration]"
                               value="<?php echo esc_attr( $s['cache_duration'] ?? 86400 );
?>" min="0" step="1" class="regular-text">
                        <p class="description"><?php esc_html_e( '86400 = 1 Tag. 0 = Cache
deaktiviert.', 'oembed-manager' ); ?></p>
                    </td>
                </tr>
                <tr>
                    <th scope="row"><?php esc_html_e( 'Wrapper CSS-Klasse', 'oembed-manager'
); ?></th>
                    <td>
                        <input type="text" name="<?php echo esc_attr( $this->option_name );
?>[wrapper_class]"
                               value="<?php echo esc_attr( $s['wrapper_class'] ?? 'jepp-
oembed-wrap' ); ?>" class="regular-text">
                    </td>
                </tr>
                <tr>
                    <th scope="row"><?php esc_html_e( 'Optionen', 'oembed-manager' ); ?></th>
                    <td>
                        <fieldset>
                            <label>
                                <input type="checkbox" name="<?php echo esc_attr( $this-
>option_name ); ?>[responsive]" value="1"
                                    <?php checked( 1, $s['responsive'] ?? 1 ); ?>>
                                <?php esc_html_e( 'Responsive Container (16:9) für Videos',
'oembed-manager' ); ?>
                            </label><br>
                            <label>
                                <input type="checkbox" name="<?php echo esc_attr( $this-
>option_name ); ?>[lazy_loading]" value="1"
```

```
                                    <?php checked( 1, $s['lazy_loading'] ?? 1 ); ?>>
                                <?php esc_html_e( 'Lazy Loading für iframes aktivieren',
'oembed-manager' ); ?>
                            </label><br>
                            <label>
                                <input type="checkbox" name="<?php echo esc_attr( $this-
>option_name ); ?>[strip_inline_styles]" value="1"
                                    <?php checked( 1, $s['strip_inline_styles'] ?? 0 ); ?>>
                                <?php esc_html_e( 'Inline-Styles aus oEmbed-HTML entfernen',
'oembed-manager' ); ?>
                            </label><br>
                            <label>
                                <input type="checkbox" name="<?php echo esc_attr( $this-
>option_name ); ?>[disable_in_feed]" value="1"
                                    <?php checked( 1, $s['disable_in_feed'] ?? 0 ); ?>>
                                <?php esc_html_e( 'oEmbed im RSS-Feed deaktivieren (nur Link
anzeigen)', 'oembed-manager' ); ?>
                            </label><br>
                            <label>
                                <input type="checkbox" name="<?php echo esc_attr( $this-
>option_name ); ?>[disable_on_frontend]" value="1"
                                    <?php checked( 1, $s['disable_on_frontend'] ?? 0 ); ?>>
                                <?php esc_html_e( 'oEmbed komplett auf dem Frontend
deaktivieren', 'oembed-manager' ); ?>
                            </label>
                        </fieldset>
                    </td>
                </tr>
            </table>
        </div>
        <?php
    }

    /**
     * Tab: Eigene Provider.
     */
    private function render_tab_custom() {
        $custom = isset( $this->settings['custom_providers'] ) ? $this-
>settings['custom_providers'] : array();
        ?>
```

```
        <div class="jepp-oem-section">
            <h2><?php esc_html_e( 'Eigene oEmbed-Provider', 'oembed-manager' ); ?></h2>
            <p class="description"><?php esc_html_e( 'Hier können Sie eigene oEmbed-Provider hinzufügen, z.B. für interne Dienste.', 'oembed-manager' ); ?></p>

            <table class="widefat jepp-oem-custom-table" id="jepp-oem-custom-table">
                <thead>
                    <tr>
                        <th><?php esc_html_e( 'URL-Pattern', 'oembed-manager' ); ?></th>
                        <th><?php esc_html_e( 'Endpoint-URL', 'oembed-manager' ); ?></th>
                        <th style="width:60px;"><?php esc_html_e( 'Regex', 'oembed-manager' ); ?></th>
                        <th style="width:60px;"><?php esc_html_e( 'Aktion', 'oembed-manager' ); ?></th>
                    </tr>
                </thead>
                <tbody>
                    <?php if ( ! empty( $custom ) ) : ?>
                        <?php foreach ( $custom as $i => $cp ) : ?>
                            <tr class="jepp-oem-custom-row">
                                <td>
                                    <input type="text" class="regular-text"
                                           name="<?php echo esc_attr( $this->option_name ); ?>[custom_providers][<?php echo $i; ?>][pattern]"
                                           value="<?php echo esc_attr( $cp['pattern'] ); ?>"
                                           placeholder="https://example.com/*">
                                </td>
                                <td>
                                    <input type="url" class="regular-text"
                                           name="<?php echo esc_attr( $this->option_name ); ?>[custom_providers][<?php echo $i; ?>][endpoint]"
                                           value="<?php echo esc_attr( $cp['endpoint'] ); ?>"
                                           placeholder="https://example.com/oembed">
                                </td>
                                <td>
                                    <input type="checkbox"
                                           name="<?php echo esc_attr( $this->option_name ); ?>[custom_providers][<?php echo $i; ?>][regex]"
                                           value="1" <?php checked( 1, $cp['regex'] ?? 0 ); ?>>
```

```
                                </td>
                                <td>
                                    <button type="button" class="button jepp-oem-remove-
row">&times;</button>
                                </td>
                            </tr>
                        <?php endforeach; ?>
                    <?php endif; ?>
                </tbody>
            </table>

            <p>
                <button type="button" class="button button-secondary" id="jepp-oem-add-
provider">
                    <?php esc_html_e( '+ Provider hinzufügen', 'oembed-manager' ); ?>
                </button>
            </p>
        </div>
        <?php
    }

    /**
     * Tab: Blockliste.
     */
    private function render_tab_blocklist() {
        $blocked = isset( $this->settings['blocked_urls'] ) ? $this->settings['blocked_urls']
: '';
        ?>
        <div class="jepp-oem-section">
            <h2><?php esc_html_e( 'URL-Blockliste', 'oembed-manager' ); ?></h2>
            <p class="description">
                <?php esc_html_e( 'URLs oder URL-Muster (eine pro Zeile), die nicht
eingebettet werden sollen. Wildcards (*) sind erlaubt.', 'oembed-manager' ); ?>
            </p>
            <textarea name="<?php echo esc_attr( $this->option_name ); ?>[blocked_urls]"
                      rows="10" class="large-text code"
                      placeholder="https://example.com/private/*&#10;https://badsite.com/*"
            ><?php echo esc_textarea( $blocked ); ?></textarea>
        </div>
        <?php
```

```
    }

    /**
     * Versteckte Felder rendern, damit andere Tabs nicht verloren gehen.
     *
     * @param string $active_tab
     */
    private function render_hidden_fields( $active_tab ) {
        $s = $this->settings;

        if ( 'providers' !== $active_tab ) {
            // Deaktivierte Provider als hidden fields
            $disabled = isset( $s['disabled_providers'] ) ? $s['disabled_providers'] :
array();
            foreach ( $disabled as $d ) {
                echo '<input type="hidden" name="' . esc_attr( $this->option_name ) .
'[disabled_providers][]" value="' . esc_attr( $d ) . '">';
            }
        }

        if ( 'display' !== $active_tab ) {
            $fields = array( 'max_width', 'max_height', 'cache_duration', 'wrapper_class' );
            foreach ( $fields as $f ) {
                $val = isset( $s[ $f ] ) ? $s[ $f ] : '';
                echo '<input type="hidden" name="' . esc_attr( $this->option_name ) . '[' . $f
. ']" value="' . esc_attr( $val ) . '">';
            }
            $checkboxes = array( 'responsive', 'lazy_loading', 'strip_inline_styles',
'disable_in_feed', 'disable_on_frontend' );
            foreach ( $checkboxes as $cb ) {
                if ( ! empty( $s[ $cb ] ) ) {
                    echo '<input type="hidden" name="' . esc_attr( $this->option_name ) . '['
. $cb . ']" value="1">';
                }
            }
        }

        if ( 'custom' !== $active_tab ) {
            $custom = isset( $s['custom_providers'] ) ? $s['custom_providers'] : array();
            foreach ( $custom as $i => $cp ) {
```

```
                echo '<input type="hidden" name="' . esc_attr( $this->option_name ) .
'[custom_providers][' . $i . '][pattern]" value="' . esc_attr( $cp['pattern'] ) . '">';
                echo '<input type="hidden" name="' . esc_attr( $this->option_name ) .
'[custom_providers][' . $i . '][endpoint]" value="' . esc_attr( $cp['endpoint'] ) . '">';
                if ( ! empty( $cp['regex'] ) ) {
                    echo '<input type="hidden" name="' . esc_attr( $this->option_name ) .
'[custom_providers][' . $i . '][regex]" value="1">';
                }
            }
        }

        if ( 'blocklist' !== $active_tab ) {
            $blocked = isset( $s['blocked_urls'] ) ? $s['blocked_urls'] : '';
            echo '<input type="hidden" name="' . esc_attr( $this->option_name ) .
'[blocked_urls]" value="' . esc_attr( $blocked ) . '">';
        }
    }
}
```

Schritt 4: Frontend-Filter

Diese Klasse modifiziert die oEmbed-Ausgabe im Frontend (Wrapper, responsive, Lazy Loading etc.).

```
<?php
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

/**
 * Modifiziert die oEmbed-Ausgabe im Frontend.
 */
class Jepp_OEM_Frontend {

    /** @var array */
    private $settings;

    public function __construct() {
        $this->settings = get_option( 'jepp_oem_settings', array() );
```

```
    // oEmbed komplett deaktivieren
    if ( ! empty( $this->settings['disable_on_frontend'] ) && ! is_admin() ) {
        add_action( 'init', array( $this, 'disable_oembed_completely' ) );
        return;
    }

    // Max-Width/Height setzen
    add_filter( 'embed_defaults', array( $this, 'set_embed_defaults' ) );

    // Cache-Dauer setzen
    add_filter( 'oembed_ttl', array( $this, 'set_cache_ttl' ), 10, 4 );

    // HTML-Ausgabe modifizieren
    add_filter( 'embed_oembed_html', array( $this, 'modify_output' ), 99, 4 );

    // Im Feed deaktivieren
    if ( ! empty( $this->settings['disable_in_feed'] ) ) {
        add_filter( 'embed_oembed_html', array( $this, 'disable_in_feed' ), 100, 4 );
    }

    // Frontend-CSS laden
    add_action( 'wp_enqueue_scripts', array( $this, 'enqueue_frontend_styles' ) );
}

/**
 * oEmbed komplett deaktivieren.
 */
public function disable_oembed_completely() {
    // Auto-Discovery deaktivieren
    remove_action( 'wp_head', 'wp_oembed_add_discovery_links' );
    remove_action( 'wp_head', 'wp_oembed_add_host_js' );

    // oEmbed REST-API Endpoint deaktivieren
    remove_action( 'rest_api_init', 'wp_oembed_register_route' );

    // oEmbed-Filter entfernen
    remove_filter( 'pre_oembed_result', 'wp_filter_pre_oembed_result', 10 );

    // Auto-Embeds deaktivieren
```

```
    remove_filter( 'the_content', array( $GLOBALS['wp_embed'], 'autoembed' ), 8 );
}

/**
 * Standard-Embed-Dimensionen setzen.
 *
 * @param array $defaults
 * @return array
 */
public function set_embed_defaults( $defaults ) {
    $max_w = $this->settings['max_width'] ?? 800;
    $max_h = $this->settings['max_height'] ?? 0;

    if ( $max_w > 0 ) {
        $defaults['width'] = $max_w;
    }
    if ( $max_h > 0 ) {
        $defaults['height'] = $max_h;
    }

    return $defaults;
}

/**
 * Cache-TTL setzen.
 *
 * @param int    $ttl
 * @param string $url
 * @param array  $attr
 * @param int    $post_id
 * @return int
 */
public function set_cache_ttl( $ttl, $url, $attr, $post_id ) {
    $custom_ttl = $this->settings['cache_duration'] ?? 86400;
    return (int) $custom_ttl;
}

/**
 * oEmbed-HTML-Ausgabe modifizieren.
 *
```

```
     * @param string $html
     * @param string $url
     * @param array  $attr
     * @param int    $post_id
     * @return string
     */
    public function modify_output( $html, $url, $attr, $post_id ) {
        if ( empty( $html ) ) {
            return $html;
        }

        // Inline-Styles entfernen
        if ( ! empty( $this->settings['strip_inline_styles'] ) ) {
            $html = preg_replace( '/\s+style\s*=\s*"[^"]*"/i', '', $html );
            $html = preg_replace( "/\s+style\s*=\s*'[^']*'/i", '', $html );
        }

        // Lazy Loading für iframes
        if ( ! empty( $this->settings['lazy_loading'] ) ) {
            $html = $this->add_lazy_loading( $html );
        }

        // Wrapper-Container
        $wrapper_class = $this->settings['wrapper_class'] ?? 'jepp-oembed-wrap';
        $is_video      = $this->is_video_embed( $html, $url );
        $responsive    = ! empty( $this->settings['responsive'] ) && $is_video;

        $classes = esc_attr( $wrapper_class );
        if ( $responsive ) {
            $classes .= ' jepp-oembed-responsive';
        }

        $html = '<div class="' . $classes . '">' . $html . '</div>';

        return $html;
    }

    /**
     * Lazy Loading Attribut zu iframes hinzufügen.
     *
```

```
     * @param string $html
     * @return string
     */
    private function add_lazy_loading( $html ) {
        // Nur wenn ein iframe vorhanden ist und noch kein loading-Attribut gesetzt
        if ( stripos( $html, '<iframe' ) !== false && stripos( $html, 'loading=' ) === false )
{
            $html = str_ireplace( '<iframe', '<iframe loading="lazy"', $html );
        }
        return $html;
    }

    /**
     * Prüft ob es sich um ein Video-Embed handelt.
     *
     * @param string $html
     * @param string $url
     * @return bool
     */
    private function is_video_embed( $html, $url ) {
        $video_hosts = array(
            'youtube.com', 'youtu.be', 'vimeo.com', 'dailymotion.com',
            'dai.ly', 'tiktok.com', 'twitch.tv', 'facebook.com/watch',
            'wistia.com', 'videopress.com',
        );

        foreach ( $video_hosts as $host ) {
            if ( stripos( $url, $host ) !== false ) {
                return true;
            }
        }

        // Fallback: iframe mit typischen Video-Attributen prüfen
        if ( preg_match( '/<iframe[^>]+src=["\'][^"\']*(?:video|embed|player)/i', $html ) ) {
            return true;
        }

        return false;
    }
```

```
    /**
     * Im Feed nur den Link anzeigen.
     *
     * @param string $html
     * @param string $url
     * @param array  $attr
     * @param int    $post_id
     * @return string
     */
    public function disable_in_feed( $html, $url, $attr, $post_id ) {
        if ( is_feed() ) {
            return '<a href="' . esc_url( $url ) . '">' . esc_html( $url ) . '</a>';
        }
        return $html;
    }

    /**
     * Frontend-Styles laden.
     */
    public function enqueue_frontend_styles() {
        wp_enqueue_style(
            'jepp-oem-frontend',
            JEPP_OEM_PLUGIN_URL . 'assets/css/frontend.css',
            array(),
            JEPP_OEM_VERSION
        );
    }
}
```

Schritt 5: Admin-CSS

```
/* oEmbed Manager – Admin Styles */

.jepp-oem-wrap {
    max-width: 1100px;
}

.jepp-oem-tabs {
```

```
    margin-bottom: 20px;
}

.jepp-oem-section {
    background: #fff;
    border: 1px solid #ccd0d4;
    border-radius: 4px;
    padding: 20px 24px;
    margin-top: 15px;
}

.jepp-oem-section h2 {
    margin-top: 0;
    padding-top: 0;
    border-bottom: 1px solid #eee;
    padding-bottom: 10px;
}

/* Provider-Tabelle */
.jepp-oem-provider-table {
    margin-top: 10px;
}

.jepp-oem-provider-table td,
.jepp-oem-provider-table th {
    vertical-align: middle;
    padding: 10px 12px;
}

.jepp-oem-provider-table code {
    font-size: 12px;
    background: #f0f0f1;
    padding: 2px 6px;
    border-radius: 3px;
    word-break: break-all;
}

.jepp-oem-provider-table tbody tr:nth-child(even) {
    background: #f9f9f9;
}
```

```
.jepp-oem-provider-table tbody tr:hover {
    background: #f0f6fc;
}

/* Eigene Provider */
.jepp-oem-custom-table td {
    vertical-align: middle;
    padding: 8px 10px;
}

.jepp-oem-custom-table input[type="text"],
.jepp-oem-custom-table input[type="url"] {
    width: 100%;
}

.jepp-oem-remove-row {
    color: #a00 !important;
    border-color: #a00 !important;
    font-weight: bold;
    font-size: 16px;
    line-height: 1;
    padding: 2px 8px !important;
}

.jepp-oem-remove-row:hover {
    color: #dc3232 !important;
    border-color: #dc3232 !important;
}

/* Provider Actions */
.jepp-oem-provider-actions .button {
    margin-right: 8px;
}
```

Schritt 6: Frontend-CSS

```
/* oEmbed Manager – Frontend Styles */

.jepp-oembed-wrap {
    max-width: 100%;
    margin: 1.5em 0;
    clear: both;
}

/* Responsive Video Container (16:9) */
.jepp-oembed-responsive {
    position: relative;
    padding-bottom: 56.25%; /* 16:9 */
    height: 0;
    overflow: hidden;
}

.jepp-oembed-responsive iframe,
.jepp-oembed-responsive object,
.jepp-oembed-responsive embed,
.jepp-oembed-responsive video {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    border: 0;
}
```

Schritt 7: Admin-JavaScript

```
/**
 * oEmbed Manager – Admin JavaScript
 */
(function ($) {
    'use strict';

    $(document).ready(function () {
```

```
        // --- Provider Tab: Alle aktivieren / deaktivieren ---

        // Invertierte Logik: Checkbox AN = Provider aktiv (NICHT in disabled-Liste)
        // Beim Absenden müssen wir die NICHT angehakten als "disabled" senden.

        // "Alle aktivieren" → alle Checkboxen abhaken
        $('.jepp-oem-select-all').on('click', function () {
            $('.jepp-oem-provider-checkbox').prop('checked', true);
        });

        // "Alle deaktivieren" → alle Checkboxen abhaken entfernen
        $('.jepp-oem-deselect-all').on('click', function () {
            $('.jepp-oem-provider-checkbox').prop('checked', false);
        });

        // Invertierte Logik: Vor dem Submit die Werte umkehren
        $('form').on('submit', function () {
            $('.jepp-oem-provider-checkbox[data-inverted="1"]').each(function () {
                // Wenn NICHT gecheckt → soll als disabled gesendet werden
                // Wenn gecheckt → NICHT senden (Provider ist aktiv)
                if ($(this).is(':checked')) {
                    $(this).prop('disabled', true); // Nicht mitsenden
                } else {
                    $(this).prop('checked', true);  // Wert mitsenden
                    $(this).prop('disabled', false);
                }
            });
        });

        // --- Custom Provider Tab: Zeilen hinzufügen/entfernen ---

        var $table = $('#jepp-oem-custom-table tbody');

        $('#jepp-oem-add-provider').on('click', function () {
            var idx = $table.find('tr').length;
            var optName = $('input[name*="custom_providers"]').length > 0
                ?
$('input[name*="custom_providers"]').first().attr('name').replace(/\[custom_providers\]\[\d+\]
.*/, '[custom_providers]')
                : 'jepp_oem_settings[custom_providers]';
```

```
            var row = '<tr class="jepp-oem-custom-row">' +
                '<td><input type="text" class="regular-text" name="' + optName + '[' + idx +
'][pattern]" placeholder="https://example.com/*"></td>' +
                '<td><input type="url" class="regular-text" name="' + optName + '[' + idx +
'][endpoint]" placeholder="https://example.com/oembed"></td>' +
                '<td><input type="checkbox" name="' + optName + '[' + idx + '][regex]"
value="1"></td>' +
                '<td><button type="button" class="button jepp-oem-remove-
row">&times;</button></td>' +
                '</tr>';

            $table.append(row);
        });

        $(document).on('click', '.jepp-oem-remove-row', function () {
            $(this).closest('tr').remove();
            // Indizes neu nummerieren
            $table.find('tr').each(function (i) {
                $(this).find('input').each(function () {
                    var name = $(this).attr('name');
                    if (name) {
                        $(this).attr('name', name.replace(/\[custom_providers\]\[\d+\]/,
'[custom_providers][' + i + ']'));
                    }
                });
            });
        });
    });
})(jQuery);
```

Schritt 8: Uninstall-Routine

```
<?php
/**
 * Wird beim Löschen des Plugins ausgeführt.
 * Entfernt alle gespeicherten Optionen und den oEmbed-Cache.
 */
```

```
if ( ! defined( 'WP_UNINSTALL_PLUGIN' ) ) {
    exit;
}

// Plugin-Option entfernen
delete_option( 'jepp_oem_settings' );

// oEmbed-Cache entfernen
global $wpdb;
$wpdb->query(
    "DELETE FROM {$wpdb->postmeta} WHERE meta_key LIKE '_oembed_%'"
);
$wpdb->query(
    "DELETE FROM {$wpdb->options} WHERE option_name LIKE '_transient_oembed_%' OR option_name
LIKE '_transient_timeout_oembed_%'"
);
```

Dateistruktur

```
oEmbed Plugin/
├── oembed-manager.php              ← Haupt-Plugin-Datei
├── uninstall.php                   ← Aufräumen bei Deinstallation
├── includes/
│   ├── class-jepp-oem-settings.php ← Admin-Einstellungsseite
│   ├── class-jepp-oem-providers.php← Provider-Verwaltung
│   └── class-jepp-oem-frontend.php ← Frontend-Filter
├── assets/
│   ├── css/
│   │   ├── admin.css               ← Admin-Styles
│   │   └── frontend.css            ← Frontend-Styles
│   └── js/
│       └── admin.js                ← Admin-JavaScript
└── languages/                      ← Übersetzungsdateien
```

Funktionsübersicht

Feature	Beschreibung
Provider an/aus	Jeden WordPress-Standard-Provider einzeln deaktivieren
Eigene Provider	Neue oEmbed-Endpunkte mit URL-Pattern hinzufügen
URL-Blockliste	Bestimmte URLs/Muster von der Einbettung ausschließen
Max. Dimensionen	Maximale Breite und Höhe für Embeds festlegen
Responsive Container	Automatisch 16:9-Container für Video-Embeds
Lazy Loading	`loading="lazy"` für iframes aktivieren
Inline-Styles entfernen	Unerwünschte Style-Attribute aus dem HTML entfernen
Feed-Deaktivierung	Im RSS-Feed nur Links statt Embeds anzeigen
Komplett deaktivieren	oEmbed auf dem gesamten Frontend abschalten
Cache-Steuerung	Cache-Dauer anpassen, Cache wird bei Änderungen geleert

Das Plugin ist sofort einsatzbereit – einfach den Ordner `oEmbed Plugin` nach `wp-content/plugins/` kopieren und im WordPress-Backend aktivieren. Die Einstellungen finden sich dann unter **Einstellungen → oEmbed Manager**.

oEmbed Plugin | eigene Provider definieren

```
<iframe height="300" style="width: 100%;" scrolling="no" title="GSAP - Animation Triad Demo: Timing, Motion, Meaning" src="https://codepen.io/editor/ZcarecroW/embed/019da77a-6995-7501-9af8-4866e69b7f1c?default-tab=html%2Cresult" frameborder="no" loading="lazy" allowtransparency="true">
  See the Pen <a href="https://codepen.io/editor/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c">
  GSAP - Animation Triad Demo: Timing, Motion, Meaning</a> by ZcarecroW (<a href="https://codepen.io/ZcarecroW">@ZcarecroW</a>)
  on <a href="https://codepen.io">CodePen</a>.
</iframe>
```

CodePen als oEmbed-Provider hinzufügen

CodePen unterstützt offiziell das oEmbed-Protokoll. Du kannst es direkt über den **Tab "Eigene Provider"** in deinem Plugin hinzufügen.

Über die Admin-Oberfläche

Gehe zu **Einstellungen → oEmbed Manager → Eigene Provider** und füge folgende Werte ein:

Feld	Wert
URL-Pattern	`https://codepen.io/*/pen/*`
Endpoint-URL	`https://codepen.io/api/oembed`

Feld	Wert
Regex	☐ (nicht angehakt)

Das Problem mit deiner URL

Deine aktuelle URL hat das Format:

```
https://codepen.io/editor/ZcarecroW/embed/019da77a-6995-7501-9af8-4866e69b7f1c
```

Das ist eine **Editor/Embed-URL**, keine Standard-Pen-URL. oEmbed von CodePen funktioniert nur mit dem kanonischen Pen-Format. Du musst also in deinem WordPress-Beitrag diese URL einfügen:

```
https://codepen.io/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c
```

Dann erkennt WordPress die URL automatisch und holt über den oEmbed-Endpoint die Einbettung.

Erweiterte Variante: Auch Editor-URLs unterstützen

Falls du auch die `/editor/`-URLs und `/embed/`-URLs automatisch auflösen möchtest, brauchst du einen zweiten Provider-Eintrag mit Regex. Da das URL-Pattern `https://codepen.io/*/pen/*` nur die Standard-Pen-URLs matcht, kannst du zusätzlich einen Regex-Provider anlegen:

Feld	Wert
URL-Pattern	`#https?://codepen\.io/.+/(pen|embed|full)/.*#i`
Endpoint-URL	`https://codepen.io/api/oembed`
Regex	☑ (angehakt)

Allerdings ist der sauberste Weg einfach die **kanonische Pen-URL** zu verwenden.

Testen

Du kannst prüfen, ob der oEmbed-Endpoint funktioniert, indem du folgende URL im Browser aufrufst:

```
https://codepen.io/api/oembed?url=https://codepen.io/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c&format=json
```

Das sollte eine JSON-Antwort mit dem Embed-HTML zurückliefern.

Hinweis zum Provider-Matching

In deiner filter_providers-Methode werden Provider über `stripos` gematcht. Eigene Provider, die über register_custom_providers mit `wp_oembed_add_provider` registriert werden, sind davon nicht betroffen – sie werden direkt in WordPress registriert und funktionieren unabhängig vom Deaktivierungs-Filter.

> ich hab nun deine beiden definitionen eingetragen und auf einer wp page einen link eingefügt:
> https://codepen.io/editor/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c
> leider ist es im frontend immer noch nur ein link.

CodePen oEmbed – Problemanalyse

Das Problem liegt an mehreren Stellen:

1. Die URL ist falsch

Die URL `https://codepen.io/editor/ZcarecroW/pen/...` ist **keine gültige CodePen-Pen-URL**. Das `/editor/` gehört dort nicht hin. Die korrekte URL lautet:

```
https://codepen.io/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c
```

2. Das Provider-Matching hat ein Problem

In deiner filter_providers-Methode wird mit `md5($pattern)` als Key gearbeitet. Beim Deaktivieren von Providern wird der `md5`-Hash des Patterns als Vergleichswert gegen das Pattern und den Endpoint geprüft – das funktioniert korrekt für Standard-Provider. Aber das eigentliche Problem ist, dass **eigene Provider über `wp_oembed_add_provider` registriert werden**, was im `init`-Hook passiert – und der `oembed_providers`-Filter greift auf eine andere interne Liste zu. Das sollte eigentlich funktionieren.

3. Der eigentliche Grund: WordPress erkennt die URL nicht

WordPress macht bei oEmbed ein **Pattern-Matching** gegen die URL. Dein Pattern `https://codepen.io/*/pen/*` matcht nur URLs im Format:

```
https://codepen.io/{username}/pen/{id}
```

Aber **nicht** URLs mit `/editor/` darin.

Lösung

Schritt 1: Verwende die **richtige URL** in deinem WordPress-Beitrag. Füge in den Block-Editor einfach diese URL auf einer eigenen Zeile ein:

```
https://codepen.io/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c
```

Schritt 2: Stelle sicher, dass dein eigener Provider korrekt eingetragen ist:

Feld	Wert
URL-Pattern	`https://codepen.io/*/pen/*`
Endpoint-URL	`https://codepen.io/api/oembed`
Regex	☐ (nicht angehakt)

Schritt 3: Leere den oEmbed-Cache. Speichere einfach nochmal die Plugin-Einstellungen – das löst flush_oembed_cache aus und löscht alle gecachten oEmbed-Einträge.

Verifikation

Du kannst prüfen, ob der CodePen oEmbed-Endpoint deine URL überhaupt akzeptiert. Öffne diese URL im Browser:

```
https://codepen.io/api/oembed?url=https://codepen.io/ZcarecroW/pen/019da77a-6995-7501-9af8-4866e69b7f1c&format=json
```

Wenn du eine JSON-Antwort mit `html`-Feld bekommst, funktioniert der Endpoint. Wenn du einen Fehler bekommst, liegt das Problem bei CodePen selbst (z.B. ist der Pen möglicherweise privat).

Optional: Auch Editor-URLs unterstützen

Falls du trotzdem auch `/editor/`-URLs unterstützen willst, brauchst du einen **zweiten** Provider-Eintrag mit Regex:

Feld	Wert
URL-Pattern	`#https?://codepen\.io/(?:editor/)?([^/]+)/(?:pen|embed|full)/(.+)#i`
Endpoint-URL	`https://codepen.io/api/oembed`
Regex	☑ (angehakt)

Wichtig: Auch mit Regex-Matching sendet WordPress die **originale URL** an den oEmbed-Endpoint. Wenn CodePen die `/editor/`-URL nicht als gültige Pen-URL akzeptiert, wird es trotzdem nicht funktionieren. Daher ist die sauberste Lösung: **Verwende die kanonische Pen-URL ohne `/editor/`**.

Skill (für KIs): oEmbed Provider Configuration Generator

> Skill: oEmbed Provider Configuration Generator

Role

You are an oEmbed provider configuration specialist. Your job is to analyze any given embed code (iframe, blockquote,
script tag, URL pattern, or API documentation) and produce the exact configuration values needed to register a custom
oEmbed provider in WordPress.

Output Format

For every provider you identify, return a table with exactly these three fields:

Field	Description
URL-Pattern	The URL scheme that matches embeddable content from this provider.
Endpoint-URL	The oEmbed API endpoint that returns embed metadata (JSON/XML).
Regex	`on` if the URL-Pattern is a regular expression, `off` if it uses simple wildcards (`*`).

Rules

1. **Analyze the input thoroughly.** The user may provide:
   - A raw `<iframe>` embed code
   - A `<blockquote>` + `<script>` embed snippet
   - A plain URL or set of example URLs
   - API documentation or a link to an oEmbed endpoint
   - A description of a service in natural language
2. **Determine the oEmbed endpoint.**
   - Check if the service is listed at https://oembed.com/providers.json — if so, use the official endpoint.
   - If the user provides an iframe `src`, extract the base domain and check for a well-known oEmbed endpoint at:
     - `https://{domain}/oembed`
     - `https://{domain}/api/oembed`
     - `https://{domain}/services/oembed`
   - If the service supports oEmbed auto-discovery (`<link rel="alternate" type="application/json+oembed" ...>`), mention this and extract the endpoint from the `href`.
   - If no oEmbed endpoint exists, clearly state: **"This service does not appear to support oEmbed natively. Consider using a proxy service like Iframely or Embedly, or implement a custom handler."**
3. **Construct the URL-Pattern.**
   - The pattern must match all embeddable content URLs from this provider (not the iframe src, but the canonical/public-facing URL users would paste into WordPress).
   - Prefer simple wildcard patterns using `*` (Regex = `off`) when possible. Examples:
     - `https://example.com/video/*`
     - `https://*.example.com/content/*`
   - Use a regex pattern (Regex = `on`) only when wildcard notation is insufficient, e.g.:
     - `#https?://(www\.)?example\.com/videos?/[\w-]+#i`
4. **Validate the endpoint URL.**
   - The endpoint must accept at least a `url` query parameter: `{endpoint}?url={content_url}&format=json`
   - Include `format=json` in the endpoint if the provider requires it.
   - Example: `https://example.com/oembed?format=json`
5. **Always provide a test instruction.** After the table, give the user a curl command or browser URL they can use to verify the endpoint works:

   ```
   Test: curl
   "https://example.com/oembed?url=https://example.com/video/12345&format=json"
   ```

6. **Handle edge cases:**
   - If the iframe contains no identifiable service, ask the user for more context (e.g., the public URL where this embed appears, or the service name).

- If multiple providers are detected (e.g., a page embedding both YouTube and SoundCloud), return one table row per provider.
- If the embed uses a proprietary JavaScript SDK with no oEmbed support, explain alternatives (custom shortcode, manual iframe allowlisting).

7. **Output example for reference:**
   Input: `<iframe src="https://player.vimeo.com/video/76979871" width="640" height="360" frameborder="0"></iframe>`
   Output:

Field	Value
URL-Pattern	`https://vimeo.com/*`
Endpoint-URL	`https://vimeo.com/api/oembed.json`
Regex	off

> Note: The URL-Pattern uses the canonical Vimeo URL (`https://vimeo.com/76979871`), not the player/iframe URL. WordPress will send the canonical URL to the oEmbed endpoint.

```
Test: curl "https://vimeo.com/api/oembed.json?url=https://vimeo.com/76979871"
```

8. **Additional context the user may find helpful:**
   - Briefly explain the difference between the **canonical URL** (what users paste) and the **player/iframe URL** (what the embed code contains).
   - If the provider requires API keys or authentication for oEmbed, mention it.
   - If the provider returns `rich` type instead of `video` type, note this as it may affect responsive wrapper behavior.

Response Language

Always respond in the same language the user writes in. The configuration values themselves (URLs, patterns) remain as-is regardless of language.