**📋 مستند پروژه: سیستم اعلان بروزرسانی افزونه‌ها در وردپرس**

**🎯 هدف کلی پروژه**

**سایت ما یک سایت دانلود افزونه های وردپرسی هست میخوایم یک تغییری در همه افزونه های قابل دانلود برای کاربران قرار بدیم که 2 کار انجام بده:**

**1.**

ساخت سیستمی برای هشدار دادن به کاربر درباره‌ی آپدیت افزونه‌ها با ویژگی‌های زیر:

* نمایش یک اعلان گرافیکی وقتی افزونه نیاز به آپدیت دارد (پیشنهاد شده از اعلان خود وردپرس برای تداخل افزونه ای استفاده نشود و اختصاصی JS سمت کلاینت ساخته شود بنابراین همینکار شود مگر دلیل دیگری در ابتدای شروع پروژه بوجود آید)
* اعلان شامل:
  + تصویر آیکن بزرگ هشدار سمت راست اعلان
  + عنوان (ثابت برای تمام افزونه ها)
  + توضیح کوتاه درباره آپدیت (متغییر برای هر افزونه)
  + دکمه «آپدیت افزونه» (متغییر برای هر افزونه)
  + دکمه «راهنمای آپدیت» (ثابت برای تمام افزونه ها)
* کاربر با کلیک روی ضربدر اعلان را ببندد
* بسته شدن با ضربدر و نمایش مجدد بعد از ۲۴ ساعت (فقط با کوکی JS، بدون PHP)
* **همه چیز سمت مرورگر (JavaScript) باشد، بدون استفاده از WP-Cron یا PHP برای زمان‌بندی**
* اطلاعات مربوط به آپدیت (نسخه جدید، لینک، متن هشدار) به راحتی قابل مدیریت باشد
* سمت افزونه ها فایل ها و کد راحت باشه

2.

در همین افزونه هایی که توسط کاربران نصب شده یک امکان ارسال تبلیغات هم وجود داشته باشه یعنی درافزونه پنل مدیریت سمت خودمون یک بخشی باشه بتونیم تصویر اپلود کنیم متن بذاریم و برای تمام سایت هایی که افزونه های مارو نصب کردن نشون داده بشه. شامل یک دکمه با لینک و متن روی دکمه

**✅ ساختار پیشنهادی (بدون نیاز به افزونه مرکزی)**

**📁 در هر افزونه وردپرس:**

1. فقط دو فایل اضافه می‌شوند:
   * check-updates.js → اسکریپت جاوااسکریپت برای بررسی و نمایش اعلان
   * update-config.json → شامل اطلاعات نسخه جدید، متن هشدار، لینک، تصویر
2. فایل JS با نسخه فعلی وردپرس مقایسه می‌کند و اگر نسخه قدیمی باشد، اعلان نمایش می‌دهد
3. از **یک کوکی JSON مشترک** برای ذخیره وضعیت بسته‌شدن هر اعلان استفاده شود (مثلاً: pluginUpdateDismissed = {"plugin1": timestamp, "plugin2": timestamp})
4. تغییرات در افزونه‌ها بسیار کم است:
   * فقط enqueue کردن فایل check-updates.js در فایل اصلی افزونه

**🛠 مدیریت آپدیت‌ها**

**🔧 فایل update-config.json (در هر افزونه):**

{

"plugin\_slug": "my-plugin",

"latest\_version": "2.1.0",

"message": "نسخه جدید افزونه در دسترس است. لطفاً برای حفظ امنیت آپدیت کنید.",

"update\_url": "https://mysite.com/updates/my-plugin.zip",

"guide\_url": "https://mysite.com/help/update-my-plugin",

"image\_url": "https://mysite.com/assets/update-warning.png"

}

**📌 نکته مهم:**

* این فایل‌ها نیازی به تغییر نسخه در فایل PHP اصلی ندارند.
* نسخه جدید را **دستی فقط در فایل JSON آپدیت می‌کنیم.**

**📈 مقیاس‌پذیری و سرور**

* انتظار داریم بیش از 100,000 سایت از این افزونه استفاده کنند.
* برای جلوگیری از فشار به هاست:
  + فایل‌های update-config.json و JS به صورت **استاتیک روی هاست ساده** میزبانی می‌شوند (پیشنهاد: زیر دامنه اختصاصی مثل updates.mysite.com)
  + کشینگ سمت مرورگر فعال باشد (مثلاً Cache-Control: public, max-age=86400)
  + هیچ پردازش PHP یا اتصال به دیتابیس روی سرور انجام نشود
  + فقط روزی یک بار در هر سایت فایل JSON فراخوانی می‌شود (میزان مصرف معقول است)

**🔑 نکات کلیدی برای اجرا**

| **ویژگی** | **توضیح** |
| --- | --- |
| کوکی‌ها | یک کوکی JSON برای همه افزونه‌ها |
| فایل‌ها | استاتیک و کش‌پذیر |
| به‌روزرسانی‌ها | فقط فایل JSON آپدیت می‌شود |
| نصب افزونه | بدون نیاز به افزونه مرکزی |
| امنیت | دسترسی عمومی برای فایل‌های JSON، محدودیتی لازم نیست مگر حساسیت خاصی داشته باشید |

# بدون نصب افزونه در وردپرس سایت های کاربران

**✅ آیا بدون افزونه مرکزی با بقیه افزونه‌ها تداخل پیدا می‌کنه؟**

**پاسخ:**

**خیر، اگر اصول زیر رعایت بشه، تداخلی ایجاد نمی‌کنه و کاملاً امن و پایدار خواهد بود.**

**⚙️ شرط‌های مهم برای اطمینان فنی**

**1. ایزوله بودن فایل JS**

کد check-updates.js باید:

* بدون وابستگی به jQuery یا دیگر لایبرری‌ها باشه
* متغیرهای global نسازه (مثلاً از window.pluginUpdateChecker با namespace استفاده کن)
* کلاس‌ها یا IDهای اختصاصی داشته باشه (مثلاً pu-alert-box نه alert یا modal)

✅ این باعث می‌شه به هیچ افزونه یا قالب دیگه‌ای تداخلی نداشته باشه.

**2. استفاده از کوکی یا LocalStorage با نام یکتا**

* مثلاً:  
  pluginUpdateDismissed = { "my-plugin": 1716457000, "another-plugin": 1716457100 }

✅ به‌جای ساخت ۱۰ کوکی جداگانه، همه در یک کوکی JSON ذخیره می‌شن، هم سبک‌تره هم تداخلی نداره.

**3. فایل JSON فقط خونده می‌شه، پردازشی نداره**

* از هیچ API سمت سرور، PHP، دیتابیس، یا AJAX استفاده نمی‌شه
* فقط یک بار در روز فایل JSON با fetch() در JS خونده می‌شه

✅ این یعنی حتی روی هاست ضعیف، ترافیک زیادی تولید نمی‌کنه.

**4. کد JS درون افزونه بارگذاری بشه فقط در پنل مدیریت**

* فقط وقتی کاربر وارد مدیریت شده (is\_admin() یا شناسایی window.location)
* نه در فرانت‌اند

✅ جلوگیری از نمایش بی‌مورد برای بازدیدکننده‌ها

**5. همه چیز مستقل از ساختار وردپرس اجرا بشه**

* اصلاً نیاز به wp-cron، فایل‌های خاص وردپرس یا hook نداریم
* افزونه‌های دیگر هیچ تعاملی با این سیستم ندارند

✅ کاملاً "non-intrusive" و مستقل خواهد بود.

**✳️ نتیجه نهایی**

🔒 تا زمانی که:

* فایل JS تمیز نوشته بشه
* فایل JSON به‌درستی ساختاردهی شه
* کوکی‌ها با namespace ذخیره بشن
* همه چیز فقط سمت کلاینت باشه

🔹 می‌تونی با خیال راحت **بدون افزونه مرکزی**، بدون هیچ تداخلی با دیگر افزونه‌ها یا قالب‌ها، این سیستم رو اجرا کنی.

**❗ نکته مهم:**

در این پروژه نباید از PHP، کرون جاب یا افزونه مرکزی استفاده بشه. همه چیز فقط در سمت کلاینت (مرورگر) باید اجرا بشه.

# بابت مورد دوم تبلیغ:

**در پنل مدیریت سمت شما (افزونه یا ابزار اختصاصی)**

می‌تونی داشته باشی:

* آپلود تصویر تبلیغ
* نوشتن متن + لینک
* دکمه فعال/غیرفعال‌سازی تبلیغ
* دکمه «انتشار» → که فایل ad-config.json رو برای افزونه‌ مربوطه روی هاست می‌نویسه یا آپدیت می‌کنه

**✅ نکته امنیتی مهم:**

برای جلوگیری از سوءاستفاده:

* هاست فایل‌های JSON باید فقط خواندنی باشه برای کاربران (نه writable)
* URL تبلیغ‌ها از دامنه خودت باشه نه متغیر، برای جلوگیری از تزریق

# بهینه سازی

حتی روی **هاست معمولی هم می‌تونه به خوبی کار کنه**.

**✅ توضیح فنی برای اطمینان:**

**🔹 ساختار درخواست‌ها:**

در سناریوی شما:

* هر افزونه فقط **روزی یک‌بار** یک فایل JSON می‌خونه (با fetch سمت مرورگر).
* مثلاً اگر ۱۰۰هزار سایت × ۱۰ افزونه = ۱ میلیون درخواست در روز باشه
* اما این درخواست‌ها:
  + **استاتیک هستن** (مثل https://updates.mysite.com/plugins/my-plugin/update-config.json)
  + به هیچ پردازش PHP یا پایگاه‌داده نیازی ندارن
  + **با کش مرورگر و کش CDN/هاست** قابل بهینه‌سازی هستن

**🔸 بهینه‌سازی‌هایی که باید انجام بدی:**

| **مورد** | **توضیح** |
| --- | --- |
| ✅ فایل‌ها روی هاست استاتیک | فایل‌های .json و .js رو توی هاست ساده یا CDN مثل Cloudflare می‌ذاری |
| ✅ کش سمت مرورگر | با تنظیم هدر HTTP مثل: Cache-Control: public, max-age=86400 |
| ✅ استفاده از زیردامنه | مثلاً updates.yoursite.com برای جدا کردن ترافیک و گزارش‌گیری بهتر |
| ✅ حجم فایل‌ها | فایل JSON معمولاً زیر 1KB خواهد بود |
| ✅ بدون کوئری PHP | چون پردازش سرور انجام نمی‌دید، مصرف CPU و RAM خیلی کم می‌مونه |

**🔹 تبلیغات چی؟ آیا اونم مشکله؟**

تبلیغات هم اگر:

* با ad-config.json پیاده‌سازی بشه (مثل update-config.json)
* و فقط توسط فایل JS سمت کلاینت لود بشه

✅ کاملاً مشابه آپدیت هست و فشار خاصی وارد نمی‌کنه، چون اونم روزی یک بار لود میشه، استاتیکه و کش میشه.

**🧮 یک مثال عددی:**

| **متغیر** | **مقدار تقریبی** |
| --- | --- |
| کاربران | ۱۰۰٬۰۰۰ سایت |
| تعداد افزونه | ۱۰ افزونه روی هر سایت |
| فایل JSON | ۱ میلیون درخواست در روز |
| حجم هر فایل JSON | حدود 0.5KB (حتی کمتر) |
| حجم کل روزانه | حدود 500MB (با کش بیشتر هم کمتر میشه) |
| ترافیک هاست | کمتر از 1GB در روز، با کش CDN حتی به 100MB هم ممکنه کاهش پیدا کنه |

**✅ نتیجه نهایی:**

**بله، کاملاً می‌تونه روی هاست معمولی کار کنه، بدون مشکل، اگر:**

1. از فایل‌های استاتیک استفاده کنی
2. کش مرورگر رو تنظیم کنی
3. تبلیغات رو هم مثل آپدیت‌ها استاتیک پیاده‌سازی کنی
4. PHP و دیتابیس رو وارد ماجرا نکنی

------------------------------------------------------------------------

مکمل:

**1. نسخه فعلی افزونه از کجا خونده بشه؟**

پیشنهاد:  
در check-updates.js نسخه فعلی افزونه را از طریق یک تگ <script> با data-version بخونید. یعنی در enqueue کردن JS در functions.php یا فایل اصلی افزونه، چیزی مثل این باشه:

wp\_enqueue\_script('plugin-check', plugin\_dir\_url(\_\_FILE\_\_) . 'check-updates.js', [], null, true);

wp\_add\_inline\_script('plugin-check', 'window.myPluginVersion = "2.0.3";');

اینطوری نیازی نیست فایل JS نسخه فعلی رو از جایی پیدا کنه، خود افزونه بهش می‌گه.

**2. پیشنهاد برای ساختار فایل تبلیغ (ad-config.json)**

برای اینکه تبلیغ مدیریت‌شده باشه و نوبتی نشون داده بشه یا قابل غیرفعال‌سازی باشه، ساختار زیر رو پیشنهاد می‌کنم:

{

"ad\_enabled": true,

"title": "قالب جدید ما منتشر شد!",

"message": "قالب حرفه‌ای ما را رایگان دانلود کنید.",

"image\_url": "https://yourdomain.com/assets/ad1.png",

"link": "https://yourdomain.com/ad-details",

"expires": "2025-07-01T00:00:00Z"

}

اگر ad\_enabled یا expires گذشته باشد، در JS نشان داده نمی‌شود.

**3. داشبورد مدیریت (پنل سمت خودت)**

پیشنهاد می‌کنم این پنل با PHP ساده یا یک mini framework مثل Laravel Lumen ساخته بشه:

* بدون نیاز به دیتابیس (فایل‌ها توی فولدر ذخیره می‌شن)
* یک UI ساده برای:
  + انتخاب افزونه
  + وارد کردن نسخه، لینک، متن، عکس
  + دکمه ذخیره که update-config.json یا ad-config.json بسازه یا آپدیت کنه
* افزودن لاگ آخرین تغییرات برای پیگیری

**4. کنترل کش فایل JSON**

پیشنهاد کن که سرور با هدرهای زیر فایل JSON رو تحویل بده:

به خصوص Access-Control-Allow-Origin: \* لازمه چون فایل از دامین جدا خونده می‌شه.

**5. پیشنهاد برای تست و مانیتورینگ اولیه**

بعد از پیاده‌سازی:

* از ۵ افزونه شروع کن برای تست واقعی
* در مرورگر DevTools شبکه رو مانیتور کن تا ببینی چند بار فایل JSON لود می‌شه
* چند قالب و افزونه سنگین مثل Elementor همزمان تست کن تا مطمئن شی تداخل ندارن

**توصیه امنیتی و مقیاس‌پذیری نهایی:**

* سیستم بدون افزونه مرکزی و فقط با فایل JS و JSON طراحی شده، پس تداخلی با وردپرس یا سایر افزونه‌ها نخواهد داشت.
* بارگذاری فایل‌ها کاملاً استاتیکه و با کش مرورگر ترکیب می‌شه، پس حتی با نصب بالا (100,000+) مشکلی روی سرور ایجاد نمی‌کنه.
* فقط کافیست فایل‌های JS و JSON درست ساختاربندی و namespace شوند تا ایزوله باقی بمانند.