

WordPress Meetup Activity

Debug Like a Pro: WordPress Developer Debugging Tools

60 minutes · Intermediate-Advanced · Browser + Local CLI (optional)

What You Will Be Able to Do

1. Configure **WP_DEBUG** constants to capture PHP errors, warnings, and notices
2. Diagnose slow queries and hook misfires with Query Monitor
3. Read JavaScript console errors and REST API calls in DevTools
4. Set a PHP breakpoint with Xdebug and step through execution
5. Choose the right tool for each category of problem
6. Reproduce issues safely in Playground before touching production

What You'll Need

1. A modern browser -- Chrome recommended (DevTools and CDP support)
2. An internet connection -- Playground downloads WordPress in the browser
3. Node.js + npm (Part 4 only) -- For the Xdebug demo, facilitator machine only
4. That's it. No local server, database, or WordPress install required for Parts 1-3

WP_DEBUG — Surface PHP Errors

1. playground.wordpress.net -- open the File Browser from the sidebar.
2. Navigate to **wp-config.php** in the site root. Locate the DEBUG section.
3. Edit the three constants (shown below). The File Browser auto-saves.

```
define( 'WP_DEBUG',      true );  
define( 'WP_DEBUG_LOG',  true );  
define( 'WP_DEBUG_DISPLAY', false );
```
4. Go to **Tools > Plugin File Editor**. Select any active plugin. Add: **echo \$undefined_variable;** on a new line. Update File.
5. Visit the site front end. Check **wp-content/debug.log** in File Browser for the warning.

WP_DEBUG Constants at a Glance

Constant	Values	Effect
WP_DEBUG	true / false	Master switch. Enables PHP error reporting and triggers the other constants.
WP_DEBUG_LOG	true / false	Writes errors to wp-content/debug.log. Works independently of DISPLAY.
WP_DEBUG_DISPLAY	true / false	Shows errors in the browser. Set false on any site where users could see it.
SCRIPT_DEBUG	true / false	Forces WordPress to load unminified .dev.js and .dev.css files for JS debugging.

Note: Recommended development config: WP_DEBUG true, WP_DEBUG_LOG true, WP_DEBUG_DISPLAY false -- errors captured silently, site looks normal to visitors.

Query Monitor — Queries, Hooks, Errors

1. playground.wordpress.net/?plugin=query-monitor -- loads Playground with Query Monitor pre-installed.
2. In the admin bar, click the performance stats numbers (timing, memory, query count -- e.g. 0.17s 32MB 45Q). There is no Query Monitor text label. The panel opens at the bottom.
3. **Database Queries** tab: note the count, total time, and any slow queries highlighted in red.
4. **Hooks and Actions** tab: use the Filter dropdown to search for a hook (e.g. init). Callbacks are shown inline -- no expand button.
5. **PHP Errors** tab: only appears when WordPress has detected PHP errors. If the tab is absent, the current page has no PHP errors.
6. **HTTP API Calls** tab: visit the site front end. Observe outbound HTTP requests made during the page load.

Query Monitor -- Panel Overview

Panel	What it shows
Database Queries	All DB queries: SQL text, calling function, execution time. Filter by type or component.
Hooks and Actions	Every hook fired on the current request. Search to find what runs when.
PHP Errors	Notices, warnings, and fatal errors -- captured even when WP_DEBUG_DISPLAY is off.
HTTP API Calls	Outbound wp_remote_get/post calls: URL, response code, timing. Flag slow integrations.
Blocks / Admin Screen	Context-sensitive: 'Blocks' tab on front end, 'Admin Screen' tab in admin. Only accessible when the admin bar is visible (exit block editor fullscreen first).
Environment	PHP version, WordPress version, active theme, must-use plugins, memory limit.

Browser DevTools -- Console and Network

1. In Playground: open the File Browser sidebar. Create a new file at **wp-content/plugins/bad-js/bad-js.php**.
2. Paste this plugin code (shown below). The File Browser auto-saves.

```
<?php
/* Plugin Name: Bad JS Demo */
add_action( 'wp_footer', function() {
    echo '<script>undeclaredFunction();</script>';
} );
```
3. Go to **Plugins** in the admin and activate **Bad JS Demo**.
4. Open Console (F12 > Console tab). Visit the site front end. Spot the **ReferenceError: undeclaredFunction is not defined**.
5. Click the filename link in the Console to jump to the Sources tab and see the offending line.
6. Open the Network tab. Reload the page. Filter by **"XHR"** or **"Fetch"** to see REST API calls (e.g. /wp-json/wp/v2/...).

Xdebug -- Step Through PHP Execution

1. This part runs on the **facilitator laptop** -- participants observe and ask questions.
2. Start Playground with Xdebug enabled via the CLI:

```
npx @wp-playground/cli@latest server \
  --xdebug --experimental-devtools --auto-mount
```
3. In Chrome, open **chrome://inspect** and click '**Open dedicated DevTools for Node**'.
4. In DevTools, open the **Sources** tab. Navigate to a plugin PHP file and click the gutter to set a Breakpoint.
5. Trigger the code path in the browser (e.g. load the front page).
6. Execution pauses at the breakpoint. Inspect variables in the right panel. Click **Step Over (F10)** to advance line by line.

Note: Xdebug requires Node.js + the Playground CLI. It does not run in the browser-only playground.wordpress.net. A post-session setup guide is included in the facilitator notes.

Debrief -- 5 Minutes

- 1 Which tool would you reach for first when a page starts loading slowly? Why?
- 2 How would you isolate whether a bug is PHP-side or JavaScript-side?
- 3 Where in your current workflow could Playground replace a local dev environment?

Tool Reference -- When to Use What

Tool	Best for	Not for
WP_DEBUG	PHP notices, warnings, fatal errors	JS errors, slow queries, hook order
Query Monitor	DB performance, hook inspection, HTTP API tracing	Step-through debugging, JS runtime errors
Browser DevTools	JS errors, REST API calls, CSS/layout issues	PHP-side logic, server performance
Xdebug	Complex PHP logic, variable state, call stack	Production use -- development only

Resources + Next Steps

Debugging in WordPress -- Developer reference: WP_DEBUG, SCRIPT_DEBUG, SAVEQUERIES and more	https://developer.wordpress.org/advanced-administration/debug/debug-wordpress/
Query Monitor -- Plugin page: free, open source, authored by John Blackbourn	https://wordpress.org/plugins/query-monitor/
Chrome DevTools docs -- Official guide: Console, Network, Sources, and Breakpoints	https://developer.chrome.com/docs/devtools/
Xdebug -- WordPress Playground: Playground CLI reference for --xdebug and CDP integration	https://developer.wordpress.org/playground/
Xdebug Official Docs -- Step debugging, profiling, and code coverage	https://xdebug.org/docs/
Common WordPress Errors -- Advanced Administration Handbook: all major error types	https://developer.wordpress.org/advanced-administration/wordpress/common-errors/
WordPress Playground -- Browser-based sandbox for safe testing and development	https://playground.wordpress.net
Learn WordPress -- Free courses and lessons on debugging and development	https://learn.wordpress.org

Feedback



Scan this QR code to provide
feedback on this activity