

Debugging



- [Enable debugging](#)
 - [Using Twig's native functions](#)
 - [Formatted output](#)
 - [Commented Twig includes](#)
 - [Set breakpoints in Twig](#)
 - [Twig breakpoints in PhpStorm](#)
 - [Twig breakpoints in other IDEs](#)
 - [Timber Debugger](#)
 - [Using Timber Debug Bar plugin](#)
-

Enable debugging

To use debugging, the constant `WP_DEBUG` needs to be set to `true`.

wp-config.php

```
define('WP_DEBUG', true);
```

Using Twig's native functions

Twig includes a `dump` function that can output the properties of an object.

Twig

```
{{ dump(post) }}
```

Which will give you:



You can also dump *everything* sent to your template (all the contents of `$context` that were passed to the Twig file) via:

```
{{ dump() }}
```

This will give you something like:



Formatted output

For a highlighted output like you see it above, you need to have [xDebug](#) enabled in your local development environment. With some environments like MAMP, enabling it is as easy as ticking a checkbox and restarting the server. Other times, it might be more complex.

An easier solution is to use the [Timber Dump Extension](#), which will make use of the Symfony VarDumper component to generate output like this when using `{{ dump() }}` in Twig:



It also works in PHP. Instead of using `var_dump` or `print_r`, you will use `dump()` as well:

```
dump($post);
```

If you want to dump and stop code execution, you can "dump & die" using `dd`:

```
dd($post);
```

You can also make use of the [VarDumper Configurator](#) which will enhance a bit further the output by creating links on classes, objects, etc. that will open the related file in your favorite IDE/editor.

Commented Twig includes

Sometimes it's difficult to know which Twig file generated a certain output. Thankfully, there's the [Timber Commented Include](#) extension. It will generate HTML comments that indicate where a template starts and where it ends:

```
<!-- Begin output of "partials/navigation.twig" -->
<nav class="navigation">...</nav>
<!-- / End output of "partials/navigation.twig" -->).
```

The extension is only active when `WP_DEBUG` is set to `true`.

Set breakpoints in Twig

Twig breakpoints in PhpStorm

With PhpStorm, you can [set breakpoints right in your Twig files](#). To make it work, you need to enable the caching of Twig files:

functions.php

```
Timber::$twig_cache = true;
```

Then, you need to reference the path to the cached files in **Settings/Preferences → Languages & Frameworks → PHP → Debug → Templates**. If you've installed Timber through Composer, the path will be `vendor/timber/timber/cache/`.

Remember that you can set the location of the cache files through the `timber/cache/location` filter:

```
add_filter('timber/cache/location', function () {
    return '/absolute/path/to/your/cached/twig/files';
});
```

Twig breakpoints in other IDEs

Other IDEs don't allow you to set breakpoints in your PHP code. You can try out the [AjglBreakpointTwigExtension](#) extension, that allows you to set breakpoints and inspect environment and context variables.

Install it as a dev-dependency:

```
composer require ajgl/breakpoint-twig-extension --dev
```

And then add it to Timber's Twig environment:

functions.php

```
add_filter('timber/twig', function (\Twig\Environment $twig) {
    if (defined('WP_DEBUG') && WP_DEBUG
        && class_exists('Ajgl\Twig\Extension\BreakpointExtension'))
    {
        $twig->addExtension(new Ajgl\Twig\Extension\BreakpointExtension());
    }

    return $twig;
});
```

Finally, you can set a breakpoint anywhere in your Twig file:

```
<nav>
    {{ breakpoint() }}
</nav>
```

Timber Debugger

The [Timber Debugger](#) package includes all three extensions mentioned above: the [Timber Dump](#) extension, the [Timber Commented Include](#) extension and the [Twig Breakpoints](#) extension.

Using Timber Debug Bar plugin

There's a [Timber add-on](#) for the [WordPress debug bar](#). **Warning:** this currently requires PHP 5.4.

🔍 #2

🔍 🔍 🔍 2026-08-30 00:14:01 UTC

🔍 🔍 🔍 2026-08-30 00:14:01 UTC