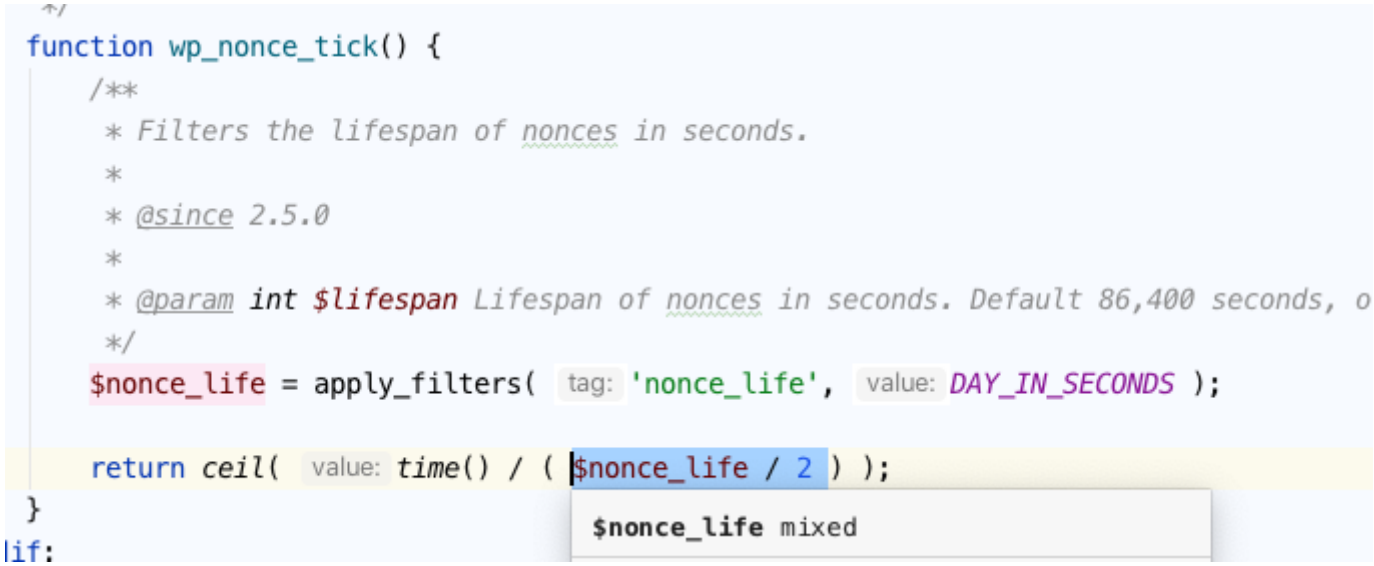
Knowledge

Knowledge is a familiarity, awareness, or understanding of someone or something, such as facts, information, descriptions, or skills, which is acquired through experience or education by perceiving, discovering, or learning.

- Nonces
- Move a site to a new domain
- Change root site
- Blocks in Simple English
- Filtering Block based content
- Auto linking project toolchains
- Performance Strategy
- Performance
- WP-CLI

Nonces

- WordPress nonces are not [cryptographic nonces](), as the latter are used only once, and the former are not:



> Nonces are regenerated every 12h, but are valid for 24h, hence that code. (12h = 1 tick, and they're valid for two ticks)

Move a site to a new domain

Move a site workflow:

1. Delete existing domain mappings for the site.
2. `wp search replace $old_url $new_url --all-tables; wp cache flush`, avoiding trailing slashes.
3. `wp rewrite flush --url=$new_url; wp rewrite flush;`
4. Login to the network site, edit the site, press Save Changes. (not sure why)
5. Verify site loads and login works.
6. re-add domain.
7. restart browsers because of cached redirects.

Change root site

To change the root site in a multisite network:

```
define( 'BLOG_ID_CURRENT_SITE', 1 );
```

> Plugins might not be compatbile with this change.

Blocks in Simple English

The following block types exist and this is what they do. I'm always getting confused by the terminology, as I'm not a native English speaker and the terminology does not cleanly map on other programming paradigms.

Block Type	**What They Say**	**What I say**
Synced patterns Enduser	*Previously Reusable Blocks* *... you will be able to arrange blocks in unlimited ways and save them as patterns for use throughout your site, directly within the editing experience. You can also specify whether to sync your patterns, so that one change applies to all parts of your site, or to keep them unsynced, so you can customize each instance.*	Patterns as symlinks: use one pattern in many places. Can be created by editors Can be converted to regular patterns.
Block Variations Developer	*Block Variations is the API that allows a block to have similar versions of it, but all these versions share some common functionality. Each block variation is differentiated from the others by setting some initial attributes or inner blocks. Then at the time when a block is inserted these attributes and/or inner blocks are applied.* *A great way to understand this API better is by using the embed block as an example.*	Blocks as a function: pass in parameters and the block acts differently.
Block Patterns Enduser	*Block Patterns are a collection of predefined blocks that you can insert into posts and pages and then customize with your own content. Using a Block Pattern can reduce the time required to create content on your site, as well as being a great way to learn how different blocks can be combined to produce interesting effects.*	Blocks group snapshot: insert a combination of blocks in one go. Good for templating layout sections, not content (use innerblocks instead)
Dynamic Blocks Developer	*Dynamic blocks are blocks that build their structure and content on the fly when the block is rendered on the front end.*	Live data blocks.

Block Styles Enduser	*Block Styles allow alternative styles to be applied to existing blocks. They work by adding a className to the block's wrapper. This className can be used to provide an alternative styling for the block if the block style is selected.*	Block Styles!

Filtering Block based content

Name	Type	Usage
`parse_blocks( string $content )`	Function	if you want to take a bunch of block attributes and store them in meta on save / generate stuff. More info
`pre_render_block`	Filter hook	More info
`render_block`	Filter hook	More info
`render_block_data`	Filter hook	More info

More resources

- A Crash Course in WordPress Block Filters

Auto linking project toolchains

If you use direnv you can automatically setup your project toolchain requirements. In the case of WordPress projects this typically includes composer, PHP, node, npm.

Simply add the required php and composer version to the project's `.envrc` file:

```
#example project requirements
use php 7.4
use composer 2

# Set node version
nvmrc=~/.nvm/nvm.sh
if [ -e $nvmrc ]; then
  source $nvmrc
  nvm use
fi
PATH_add node_modules/.bin
```

To make this happen, you must add support for the switching by adding the following to `~/.direnvrc`:

```
# Usage: use php <version>
#
# Loads the specified php version into the environent
#
use_php() {
  php --version | grep -q "PHP $1" || (brew unlink php@$1 &&  brew link php@$1 --force)
}

# Usage: use composer <version>
#
# Loads the specified composer version into the environent
#
```

```
use_composer() {
  composer --version | grep -q "version $1" || composer self-update --$1
}
```

Performance Strategy

- CDN for assets
- Full page cache such as Batcache
- Fragment caching for menus
- Longcache for lower traffic sites.

Performance

When migrating content, **suspend cache invalidation and flush the cache afterwards**.

With that in mind, the following optimisation reduces the number of database queries to 1:

```
-□□$post_data = [
-□□□'ID' => $post_id,
-□□□'post_content' => $content,
-□□];
-
-□□$updated = wp_update_post( $post_data, true );
+□□$updated = $wpdb->update(
+□□□$wpdb->posts,
+□□□[
+□□□□'post_content' => $content,
+□□□],
+□□□[
+□□□□'ID' => $post_id,
+□□□]
+□□);
```

WP-CLI

To download all attachment files from a remote site: from your local uploads directory:

```
wp post list --post_type=attachment --field=_wp_attached_file | xargs -I {} wget -x -nH --cut-dirs=2 "https://$DOMAIN/wp-content/uploads/{}"
```