Examples

n. One that is representative of a group as a whole.
n. One serving as a pattern of a specific kind.
n. A similar case that constitutes a model or precedent.

- Minimal Cron
- Run WP CLI On All Sites Within A Network
- Hooks for assets
- Updating post data on status transition
- Filter one-liners
- WP-CLI create post with custom tax
- Hiding blocks from the block inserter

Minimal Cron

Change the following:

- Prefix: svd_

```
function svd_deactivate() {
	wp_clear_scheduled_hook( 'svd_cron' );
}

add_action('init', function() {
	add_filter( 'cron_schedules', 'svd_schedule_cron' );
	add_action( 'svd_cron', 'svd_run_cron' );
	register_deactivation_hook( __FILE__, 'svd_deactivate' );

	if (! wp_next_scheduled ( 'svd_cron' )) {
		wp_schedule_event( time(), 'minute', 'svd_cron' );
	}
});

function svd_schedule_cron( $schedules ) {
	$schedules[ 'minute' ] = array( 'interval' => 1 * MINUTE_IN_SECONDS, 'display' => __( 'Every 1
minute.', 'svd' ) );
	return $schedules;
}

function svd_run_cron( ) {
	// Do your stuff!
}
```

Run WP CLI On All Sites Within A Network

```
# update an option.
wp site list --field=url | xargs -I % wp --url=% option update [option name] [option value]
```

Hooks for assets

Quick reference for where to hook in styles and scripts:

Where	*Action*
Admin	admin_enqueue_scripts
Frontend	wp_enqueue_scripts
Block editor (admin)	enqueue_block_editor_assets
Blocks (front and admin)	enqueue_block_assets

Don't register or enqueue scripts and styles in init, this will break WordPress updates and unexpected things might happen.

Updating post data on status transition

You might be tempted to set the WP_Post properties (but this is an action), or use `wp_update_post()` but there is some hard-coded behaviour in `wp_insert_post()` that might affect your data.

So it's better to emulate `wp_publish_post()`.

```
function action_on_future_to_publish( $post ) {
	global $wpdb;

	if ( ! wp_is_post_revision( $post ) ) {
		// Update via the database to bypass wp_insert_post resetting the dates.
		// @see wp_publish_post()
		$data = [
			'post_modified'     => $post->post_date,
			'post_modified_gmt' => $post->post_date_gmt,
		];
		$wpdb->update( $wpdb->posts, $data, [ 'ID' => $post->ID ] );

		// After bypassing the WP caches, refresh the cache.
		clean_post_cache( $post->ID );
	}
}
add_action( 'future_to_publish', 'action_on_future_to_publish', 10, 1 );
```

Filter one-liners

Render links in content as HTML A elements:

```
add_filter( 'the_content', 'make_clickable' );
```

WP-CLI create post with custom tax

`wp mycpt create --tags='test1,test2'`

```
$tags = null;
if ( isset( $assoc_args['tags'] ) ) {
	$tags = wp_parse_list( $assoc_args['tags'] );
	unset( $assoc_args['tags'] );
}

...

$command = "post create --porcelain " . assoc_args_to_str( $assoc_args );
$id = WP_CLI::runcommand( $command, [ 'return' => true ] );
if ( ! (bool) $id ) {
	WP_CLI::error( 'Post not saved!' );
}
wp_set_object_terms( $id, $tags, TAX_MYCPT_TAG );
WP_CLI::success( sprintf( 'Created post %s.', $id ) );
```

Hiding blocks from the block inserter

Hiding blocks allow the blocks to be used for existing content, but not for new content.

```
function get_plugin_settings() {
	$disabled_blocks = [];
    	$disabled_blocks[] = 'my/blockname';

	return [
		'disabledBlocks' => $disabled_blocks,
	];
}

# enqueue scripts hook:
	wp_localize_script(
		'myblocks-editor',
		'myBlocksSettings',
		get_plugin_settings()
	);
```

Javascript:

```
import { addFilter } from '@wordpress/hooks';

// Define disabled blocks.
// If you need to rely on logic available only in PHP,
// pass this data using a global variable instead.
const DISABLED_BLOCKS = window.myBlocksSettings.disabledBlocks;

/**
 * Conditionally enable/disable insertion of blocks.
 *
 * Ensure sure this function runs BEFORE you register your blocks.
 *
 * @param {object} settings block type definition
```

```
 * @param {string} name name of the block type
 * @returns {object} block type settings
 */
function filterBlockRegistration( settings, name ) {
	if ( ! DISABLED_BLOCKS.includes( name ) ) {
		return settings;
	}

	// Ensure there is a supports section
	if ( undefined === settings.supports ) {
		settings.supports = {};
	}

	// Disable the UI to add this block .
	// If the block is added in another way,
	// e.g. legacy, programmatically, copy-paste, it will still work.
	settings.supports.inserter = false;

	return settings;
}

addFilter(
	'blocks.registerBlockType',
	'myNamespace',
	filterBlockRegistration
);
```