Disable Editor functionality

- Restrict block options
- Curate heading levels
- Disable the Pattern Directory
- Disable block variations
- Disable block styles
- Disable access to the Code Editor
- Disable responsive editing
- Disable block states editing
- Disable formatting options for RichText blocks

This page is dedicated to the many ways you can disable specific functionality in the Post Editor and Site Editor that are not covered in other areas of the curation documentation.

Restrict block options

There might be times when you don't want access to a block at all to be available for users. To control what's available in the inserter, you can take two approaches: an allow list that disables all blocks except those on the list or a deny list that unregisters specific blocks.

Curate heading levels

Core WordPress blocks with a heading level dropdown include support for the `levelOptions` attribute. This applies to the Heading, Site Title, Site Tagline, Query Title, Post Title, and Comments Title blocks. The `levelOptions` attribute accepts an array of numbers corresponding to heading levels, where `1` represents H1, `2` represents H2, and so on.

This attribute allows you to specify which heading levels should appear in the dropdown UI, providing a lightweight curation method that does not require block deprecations. Any existing heading levels are preserved in the markup, while `levelOptions` only affects the UI display.

You can apply this attribute directly in the block markup, a technique that will be commonly used in block templates, template parts, and patterns. For example, the following markup disables H1, H2, and H6 in the Heading block by setting `"levelOptions":[3,4,5]`.

```
<!-- wp:heading {"level":3,"levelOptions":[3,4,5],"className":"wp-block-heading"} -->
<h3 class="wp-block-heading">Markup example</h3>
<!-- /wp:heading -->
```

You can also use block filters to set the default value of this attribute globally or for specific blocks. The example below disables H1, H2, and H6 for all Heading blocks. You can further customize this by restricting certain heading levels based on conditions like user capabilities.

```
function example_modify_heading_levels_globally( $args, $block_type ) {

    if ( 'core/heading' !== $block_type ) {
        return $args;
    }

    // Remove H1, H2, and H6.
    $args['attributes']['levelOptions']['default'] = [ 3, 4, 5 ];

    return $args;
}
add_filter( 'register_block_type_args', 'example_modify_heading_levels_globally', 10, 2 );
```

Disable the Pattern Directory

To fully remove patterns bundled with WordPress core from being accessed in the Inserter, the following can be added to your `functions.php` file:

```
function example_theme_support() {
    remove_theme_support( 'core-block-patterns' );
}
add_action( 'after_setup_theme', 'example_theme_support' );
```

Disable block variations

Some Core blocks are actually block variations. A great example is the Row and Stack blocks, which are actually variations of the Group block. If you want to disable these “blocks”, you actually need to disable the respective variations.

Block variations are registered using JavaScript and need to be disabled with JavaScript. The code below will disable the Row variation.

```
wp.domReady( () => {
    wp.blocks.unregisterBlockVariation( 'core/group', 'group-row' );
});
```

Assuming the code was placed in a `disable-variations.js` file located in the root of your theme folder, you can enqueue this file in the theme’s `functions.php` using the code below.

```
function example_disable_variations_script() {
    wp_enqueue_script(
        'example-disable-variations-script',
        get_template_directory_uri() . '/disable-variations.js',
        array( 'wp-dom-ready' ),
        wp_get_theme()->get( 'Version' ),
        true
    );
}
add_action( 'enqueue_block_editor_assets', 'example_disable_variations_script' );
```

Disable block styles

There are a few Core blocks that include their own block styles. An example is the Image block, which includes a block style for rounded images called “Rounded”. You many not want your users to round images, or you might prefer to use the border-radius control instead of the block style. Either way, it’s easy to disable any unwanted block styles.

Unlike block variations, you can register styles in either JavaScript or PHP. If a style was registered in JavaScript, it must be disabled with JavaScript. If registered using PHP, the style can be disabled with either. All Core block styles are registered in JavaScript.

So, you would use the following code to disable the “Rounded” block style for the Image block.

```
wp.domReady( () => {
    wp.blocks.unregisterBlockStyle( 'core/image', 'rounded' );
```

```
});
```

This JavaScript should be enqueued much like the block variation example above. Refer to the block styles documentation for how to register and unregister styles using PHP.

Disable access to the Code Editor

The Code Editor allows you to view the underlying block markup for a page or post. While this view is handy for experienced users, you can inadvertently break block markup by editing content. Add the following to your `functions.php` file to restrict access.

```
function example_restrict_code_editor_access( $settings ) {
    $settings[ 'codeEditingEnabled' ] = false;

    return $settings;
}
add_filter( 'block_editor_settings_all', 'example_restrict_code_editor_access' );
```

This code prevents all users from accessing the Code Editor. You could also add capability checks to disable access for specific users.

Disable responsive editing

The “Responsive styles” option in the View menu lets users apply style changes to a single viewport only. If you want style edits to always apply to every viewport, add the following to your `functions.php` file.

```
function example_disable_responsive_editing( $settings ) {
    $settings[ 'responsiveEditingEnabled' ] = false;

    return $settings;
}
add_filter( 'block_editor_settings_all', 'example_disable_responsive_editing' );
```

Responsive styles already defined in `theme.json` or in Global Styles continue to be applied on the front end.

Disable block states editing

The state controls for blocks in the block inspector and Global Styles let users apply block styles to states. To hide those controls, add the following to your `functions.php` file.

```
function example_disable_block_states_editing( $settings ) {
    $settings[ 'blockStatesEditingEnabled' ] = false;

    return $settings;
}
add_filter( 'block_editor_settings_all', 'example_disable_block_states_editing' );
```

State styles already saved in `theme.json`, in Global Styles, or in a block's `style` attribute continue to be applied in the editor and on the front end. This setting does not affect viewport state controls, which are controlled by `responsiveEditingEnabled`.

Disable formatting options for RichText blocks

Blocks that support RichText come with the default formatting options provided by WordPress.

Formatting options need to be disabled with JavaScript using `unregisterFormatType`. The code below will globally disable the Inline Image, Language, Keyboard Input, Subscript, and Superscript options.

```
wp.domReady( () => {
    wp.richText.unregisterFormatType( 'core/image' );
    wp.richText.unregisterFormatType( 'core/language' );
    wp.richText.unregisterFormatType( 'core/keyboard' );
    wp.richText.unregisterFormatType( 'core/subscript' );
    wp.richText.unregisterFormatType( 'core/superscript' );
});
```

This JavaScript should be enqueued much like the block variation example above.

□□□ #2
□ □□□ □□□ 2026-10-31 18:28:11 UTC
□ □□□ □□□ 2026-10-31 18:28:12 UTC