Block Stylesheets

- Why use block stylesheets?
- Creating block stylesheets
  - Organizing and naming block stylesheets
  - Adding CSS to a block stylesheet
  - Registering a block stylesheet

When styling blocks, you should always do so via the `theme.json` styles property if possible. This ensures that your styles have the best compatibility across the system, working alongside the default WordPress styles, those added by plugins, and user customizations.

But there are times when you simply need to step outside of what's easily achievable via `theme.json`. For those cases, you should use WordPress' built-in block stylesheets system.

In this article, you will learn how to register per-block stylesheets, but remember that `theme.json` should be your first choice for styling in most cases.

Why use block stylesheets?

The primary use case for block stylesheets is when you have too much CSS to add to `styles.blocks.{blockname}.css` in `theme.json`. This property allows you to add custom CSS, but it's only ideal when it's just a small bit of code. This is because you lose out on syntax highlighting and must place everything in a single line (JSON doesn't support line breaks).

You may also be tempted to put all your custom CSS into your theme's primary `style.css` file. That may be OK for some use cases, but the block stylesheets system often offers better performance by only loading the block's CSS if the block is in use on a page. On the front end, it will also inline this code within the `<head>` area.

Creating separate stylesheets for individual blocks is also beneficial for larger and more complex projects that have a lot of custom CSS for many different blocks. The separation of the files makes it easier to organize and manage your code.

Creating block stylesheets

To create custom block stylesheets, there are three steps you must take:

1. Decide on an organizational and naming scheme.
2. Write your custom CSS.
3. Register your custom block stylesheet(s).

Organizing and naming block stylesheets

Before registering a block stylesheet, you first need to know what folder you will store your custom block stylesheets in. This can be anywhere you choose (there is no standard location), and the code below will assume you are putting block stylesheets in an `/assets/blocks` folder in your theme.

You should also decide on how you will name your CSS files. Again, there is no standard naming convention, but a good option is to use the block namespace and slug like so: `{namespace}-{slug}.css`. With this naming convention, a stylesheet for the `core/group` block would become `core-group.css`.

Here is an example structure of what this could look like with CSS files for a few core blocks:

- `assets/`
  - `blocks/`
    - `core-group.css`
    - `core-image.css`
    - `core-media-text.css`

Adding CSS to a block stylesheet

To style a core block, the most important thing you need to know is its CSS class. This is automatically generated according to the block's namespace and slug in the form of `.wp-block-{namespace}-{slug}`.

Here is an example of styling a block with the namespace and slug of `super/duper` would look like:

```
.wp-block-super-duper {
    /* custom CSS goes here. */
}
```

Core WordPress blocks are an exception to this naming rule. Their namespace is `core`, but this is not included in any of the core blocks' CSS classes. Instead, they use the `.wp-block-{slug}` format.

It’s possible for third-party block developers to change the CSS class that gets output, so this general guide may not always be true for third-party blocks. In those cases, you will want to locate the block’s CSS class in the source code.

Suppose that you wanted to add some custom styling for the core Image block, which has the namespace and slug of `core/image`. You would need to target the `.wp-block-image` class.

Let’s try creating a gradient background, which essentially acts as a *faux* border for the `<img>` element within the Image block. The goal is to create a style that looks like this:

> WordPress post editor showing an image of palm trees with an orange-to-red gradient border.

First, create an `/assets/blocks/core-image.css` file in your theme. Then, add this CSS code to it:

```
.wp-block-image img {
    padding: 1rem;
    background: linear-gradient(-60deg,#ff5858,#f09819);
}
```

Because this stylesheet isn’t registered, your custom styles won’t show in the editor or on the front end yet.

Registering a block stylesheet

To register your block stylesheet, you will use the `wp_enqueue_block_style()` function. When registering block stylesheets, you should also execute the code on the `init` hook.

The `wp_enqueue_block_style()` function accepts two parameters:

- **`$block_name`**: The block name, including both the namespace and slug (e.g., `core/image`).
- **`$args`**: An array of arguments that is passed to `wp_register_style()`:
  - **`handle`**: A unique handle for your stylesheet.
  - **`src`**: The source URL for the stylesheet.
  - **`path`**: The directory path for the stylesheet (needed to inline the CSS in `<head>`).
  - **`deps`**: An array of registered stylesheet handles this stylesheet depends on.
  - **`ver`**: A custom stylesheet version number.
  - **`media`**: The media for which the stylesheet has been defined.

To register your custom stylesheet for the Image block, add this code to your `functions.php` file:

```
add_action( 'init', 'themeslug_enqueue_block_styles' );

function themeslug_enqueue_block_styles() {
    wp_enqueue_block_style( 'core/image', array(
```

```
        'handle' => 'themeslug-block-image',
        'src'    => get_theme_file_uri( "assets/blocks/core-image.css" ),
        'path'   => get_theme_file_path( "assets/blocks/core-image.css" )
    ) );
}
```

You can also configure additional arguments for your call to `wp_enqueue_block_style()`, but the above is the minimum needed for WordPress to inline your CSS code in the `<head>` area of the site.

For a deeper dive into block stylesheets, check out Leveraging theme.json and per-block styles for more performant themes on the WordPress Developer Blog.

□□□ #2
□ □□□ □□□ 2026-10-31 18:40:20 UTC
□ □□□ □□□ 2026-10-31 18:40:22 UTC