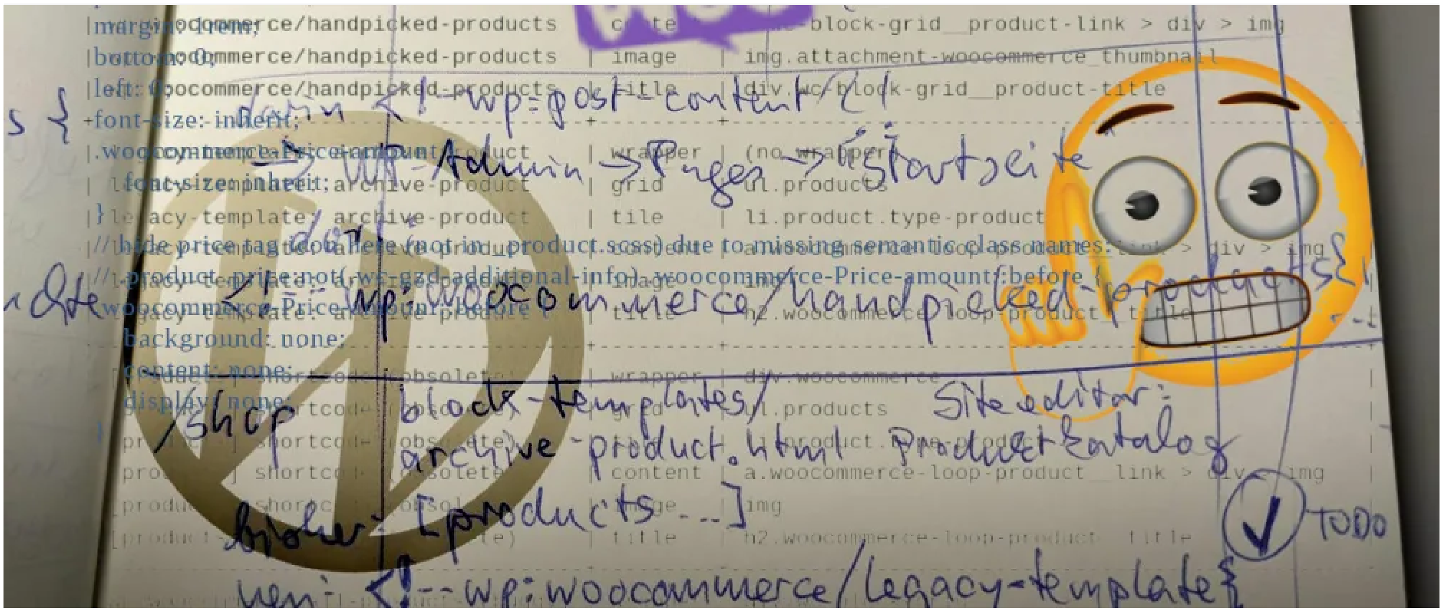


Edit Manage Stats

Ingo Steinke
Posted on Jan 13 • Updated on Mar 1

9

WooCommerce Blocks naming inconsistencies and solutions

#wordpress #woocommerce #webdev #devrant

Using WordPress as a Developer (10 Part Series)

I want to share some more experience about WordPress, WooCommerce and, more specifically, using WooCommerce blocks in a full site editing enabled theme. Don't get me wrong, I have been using WordPress from the beginning and I will continue to do so. But recently, I had a lot of trouble figuring out how to use the latest WordPress features correctly as a web developer and it seems that I am not the only one.

WooCommerce Rants and Resources 😠🤓

In this post, you will find some facts that might spare you, as a fellow developer, some gotchas, hair-pulling, and teeth-grinding, including specific error messages and questions that Google claimed to have not one single bit of information about on the whole internet. As if! Now there are some more helfpul links.

Inconsistency of Naming

What's still most irritating to me is the inconsistency of naming, especially when different technologies and plugins are involved, and even more so when some have been translated into German and others haven't.

The WooCommerce Shop page (PLP) == Product Catalog Site Template == WordPress Product Archive

So the default product listing page containing a product grid of all products in the shop is usually called "the shop" in the backend (WP-admin) main menu with a matching default URL slug " `/shop` ". But it can also be called a "product archive", as products are, technically, just a custom post type, and any WordPress page showing an overview of all posts of a certain type is also called an archive page, as we can also see when inspecting the page body's CSS class names.

```
▼<body class="archive post-type-archive post-type-archive-product logged-in admin
 -bar wp-embed-responsive theme-wordpress_theme_foodtogether woocommerce-shop woo
 commerce woocommerce-page woocommerce-js woocommerce-active customize-support"
```

But, adding another moniker, the main shop page can also be called the "product catalog" when editing page templates using the (full) site editor ("edit website").

Blocks, Legacy Blocks, and Legacy Shortcodes

Likewise, what would we call the blocks that actually produce a product grid? Well, it depends! There is a so-called "legacy products block" which is technically called `<!-- wp:woocommerce/legacy-template {"template":"archive-product"} /-->` (here's the "archive" again), also known as the WooCommerce classic template block, while another, maybe even more legacy legacy version is not a block, but rather a classic shortcode, which produces a visually and functionally similar output using the classic WooCommerce PHP code, simply written as `[products]`. Both versions can have added parameters, some of which are documented or available in the block editor's UI and some aren't.

Legacy Products, handpicked Products, all Products?

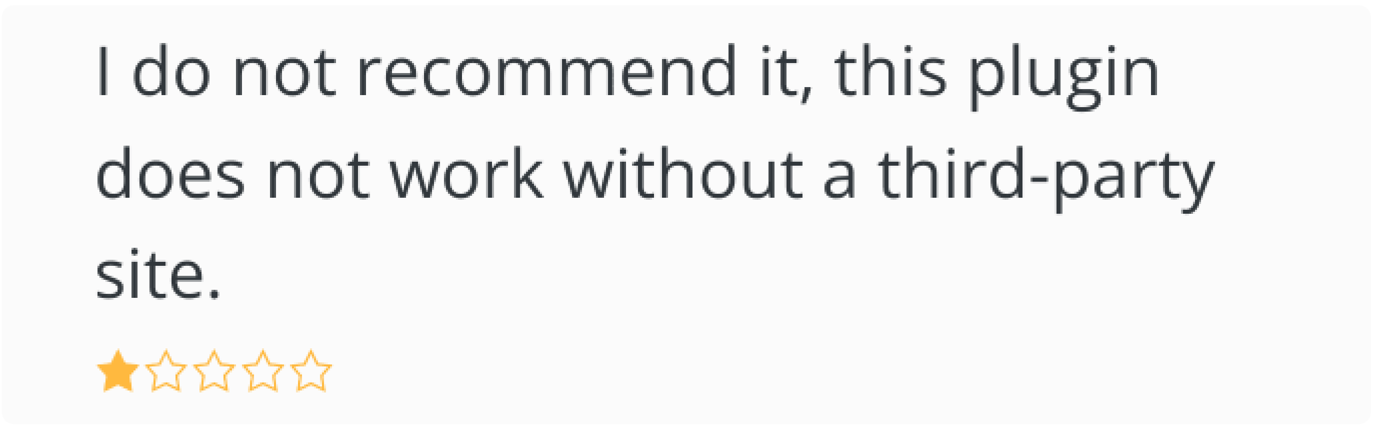
If we dare to use some more advanced block-editing, single-page-application (SPA) style Java-Script-over-PHP, and probably still beta, alternatives provided by the WooCommerce Blocks plugin, we have some more choices, like `<!-- wp:woocommerce/all-products -->` or `<!-- wp:woocommerce/handpicked-products -->`.

Any choice comes with a disadvantage. While many third-party plugins are still incompatible with modern WooCommerce blocks at the beginning of 2023, the classic or legacy template does not offer any configuration options, so we can't get rid of pagination to display all products on one page.

Even Gustavo Roganti's workaround in `functions.php` does not help here. Please comment below this post, or on GitHub, if you know a solution!

(In)Compatibility Idiosyncrasies, Markup Variations and Class Names 🙃

If you need any guidance about possible features, bugfixes, workarounds, and compatibility status, you should know which one exactly you are using. Not every one of those behaves the same, some are fully compatible with some third-party plugins, some aren't, and some have their very own limitations, bugs, or variation of generated markups and CSS class names. We discuss more details about WooCommerce Blocks markup and classes later.


I do not recommend it, this plugin does not work without a third-party site.

★☆☆☆☆

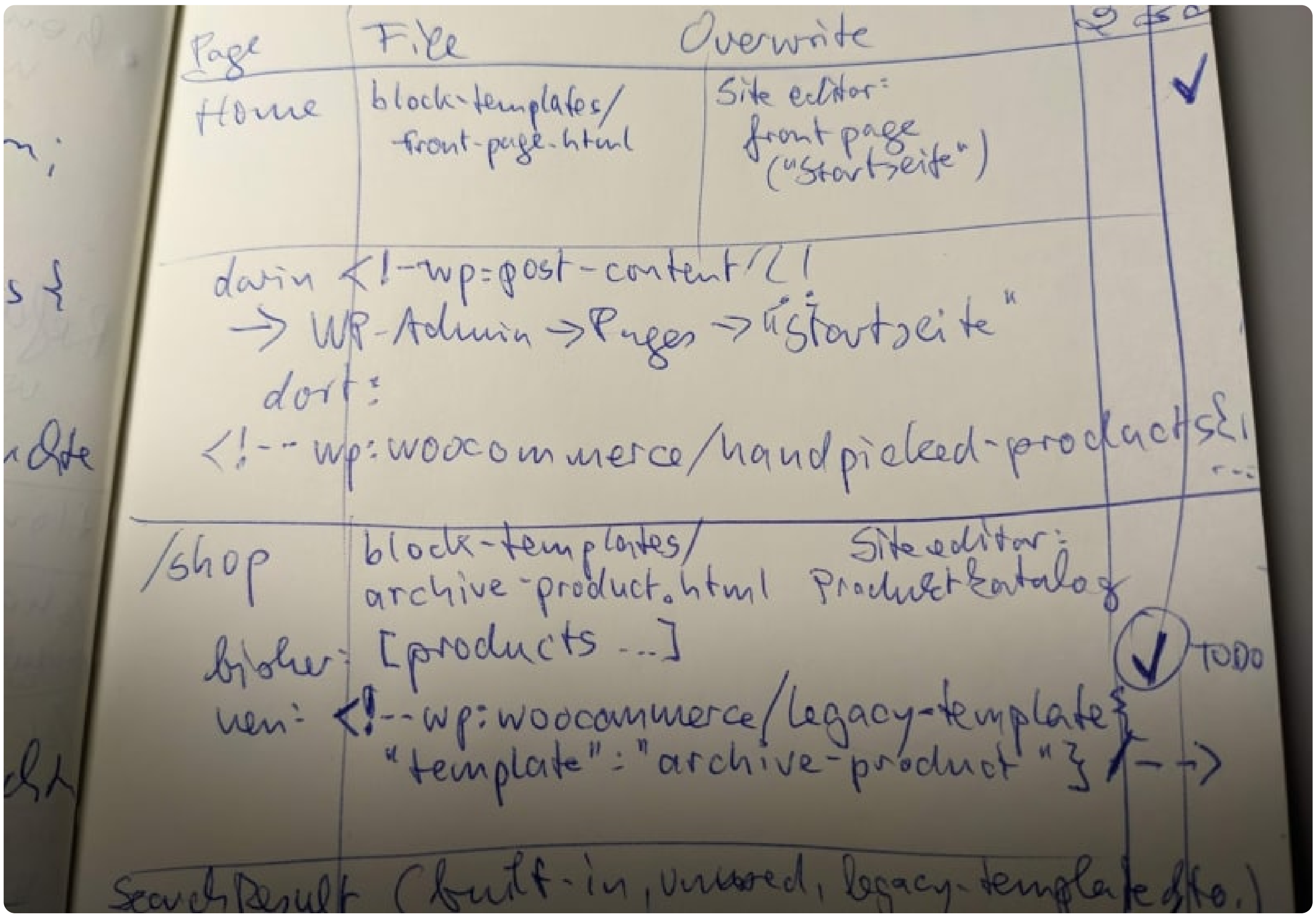
No Support when using Third-Party Plugins?

And beware, if you have written your own custom theme code or just in case you might be using some other plugins besides the ones officially recommended by WooCommerce, you can guess the answer to your support issue.

Unintentional Obfuscation

Another very annoying and error-prone aspect of full-site editable themes, at least this legacy block theme built on early 2022 or even 2021 templates, is the indirect chaining, and thus, obfuscation, of the actual content source. Editing the front page template, we might find a code snippet `<!-- wp:post-content {"layout":{"inherit":true}} /-->` which loads the actual content of a post or page currently set as home page, where in turn we might find a template part snippet like `<!-- wp:template-part {"slug":"section-newsletter","theme":"my_legacy_block_theme"} /-->` which opens a file called `block-template-parts/section-newsletter.html` only to find another code snippet `<!-- wp:pattern {"slug":"my_legacy_block_theme/section-newsletter"} /-->` which links to another file, `block-patterns/newsletter.php`, containing some PHP boilerplate code that returns more WordPress block editor code, like `<!-- wp:group {"backgroundColor":"my-theme-color-23"} --><!-- wp:mailpoet/subscription-form-block {"formId":1} /--><!-- /wp:group -->`, referencing a named color defined in `theme.json` and calling another WordPress block, provided by another plugin (MailPoet).

Photograph of my notebook, a detective journal trying to trace where to edit and verify the actual code that produces the final front-end markup in our legacy WooCommerce block theme.



Page	File	Overwrite		
Home	block-templates/ front-page.html	Site editor: front page ("Startseite")		✓
darin <!--wp:post-content/<! → WP-Admin → Pages → "Startseite" dort: <!-- wp:woocommerce/handpicked-products{ ...				
/shop	block-templates/ archive-product.html	Site editor: Produktkatalog		
bisher: [products ...] neu: <!--wp:woocommerce/legacy-template{ "template":"archive-product"}/-->			✓	TODO
Search Result (built-in, unused, legacy-template etc.)				

Round and round we go, wasting file I/O and CPU cycles to produce some simple HTML markup that can hopefully be cached as a static asset by just another plugin which attempts to contain the mosaic (not to call it a mess) that our site has been built upon.

Reviewing and refactoring our code, many of those template chains can, of course, be simplified to reduce the time it takes for the server to load and for a developer to understand what is going on and where to find the right place to edit some code, if needed.

Trying to Trace the Root Cause 🕵️

So after spending hours and days trying to find the relevant lines of code hidden inside the obfuscating boilerplate stuff and redirection chains, we can finally start to isolate and reproduce our actual issues and finally file some meaningful reproducible minimal examples for questions and bug reports.

Support Cases, GitHub Issues, StackOverflow Questions

Expect more to come, as I have several support cases, GitHub issues, StackOverflow questions, and code snippets from trying to maintain a full-site-editing enabled

WooCommerce block theme, that has already become a piece of legacy code despite having been conceived about one year ago. Here are some excerpts of my recent bookmarks:

- PayPal Payments Not Working With New WooCommerce Checkout Blox
- WooCommerce PayPal Payments doesn't support the new Checkout block
- WooCommerce-Blocks: Allow more than 6 rows in product grid blocks
- WooCommerce: Change number of products displayed per page
- How to display Woocommerce Category image?
- WooCommerce: Sending multiple AJAX requests to add products to cart failing
- WOOF (now HUSKY) product filter pagination always returns 0
- WOOF: database error: table wordpress.wp_woof_sd doesn’t exist
- MailPoet: Opt-in causes WooCommerce Checkout fatal error 400 invalid parameter extensions

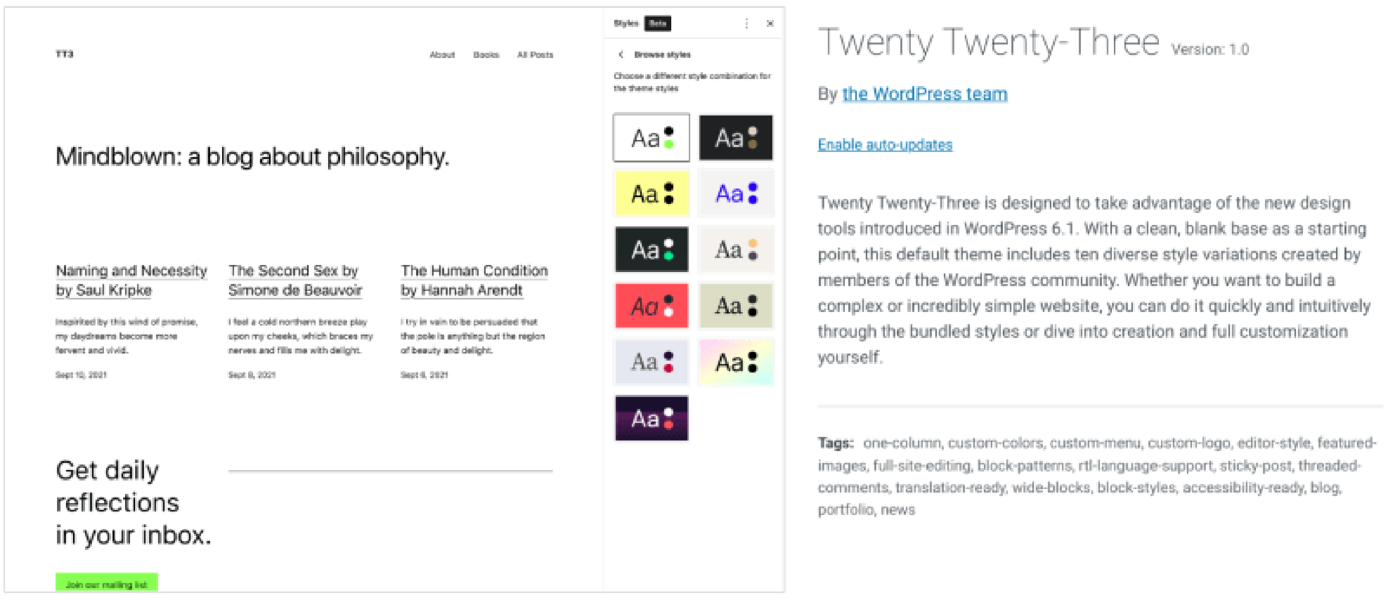
Don't be an Early Adopter!

What I already learned and stated in the previous posts about WooCommerce and WordPress, and what might have seemed obvious looking back now: don't be an early adopter of new technology if you plan to develop something for a production system! Especially when you are dealing with a mosaic of third-party code and user-editable fragments like it's the case for WordPress.

Twenty-Twenty-Three to escape our Dilemma?

I tried and started to rebuild the same shop functionality using a new, standard Twenty-Twenty-Three theme instead of our own, now outdated, early adopter legacy code block theme.

This fixed most of the bugs, but it eliminated the features and most of the styles as well. So let's take the best out of both worlds to fix our existing theme and make it align to the current standard or best practices again.



Mindblown: a blog about philosophy.

Naming and Necessity by Saul Kripke

Inspirited by this wind of promise, my daydreams become more fervent and vivid.

Sept 10, 2021

The Second Sex by Simone de Beauvoir

I feel a cold northern breeze play upon my cheeks, which braces my nerves and fills me with delight.

Sept 8, 2021

The Human Condition by Hannah Arendt

I try in vain to be persuaded that the pole is anything but the region of beauty and delight.

Sept 6, 2021

Get daily reflections in your inbox.

Choose a different style combination for the theme styles

Twenty Twenty-Three Version: 1.0

By the WordPress team

Enable auto-updates

Twenty Twenty-Three is designed to take advantage of the new design tools introduced in WordPress 6.1. With a clean, blank base as a starting point, this default theme includes ten diverse style variations created by members of the WordPress community. Whether you want to build a complex or incredibly simple website, you can do it quickly and intuitively through the bundled styles or dive into creation and full customization yourself.

Tags: one-column, custom-colors, custom-menu, custom-logo, editor-style, featured-images, full-site-editing, block-patterns, rtl-language-support, sticky-post, threaded-comments, translation-ready, wide-blocks, block-styles, accessibility-ready, blog, portfolio, news

Mysteriously Missing Products using wc-ajax

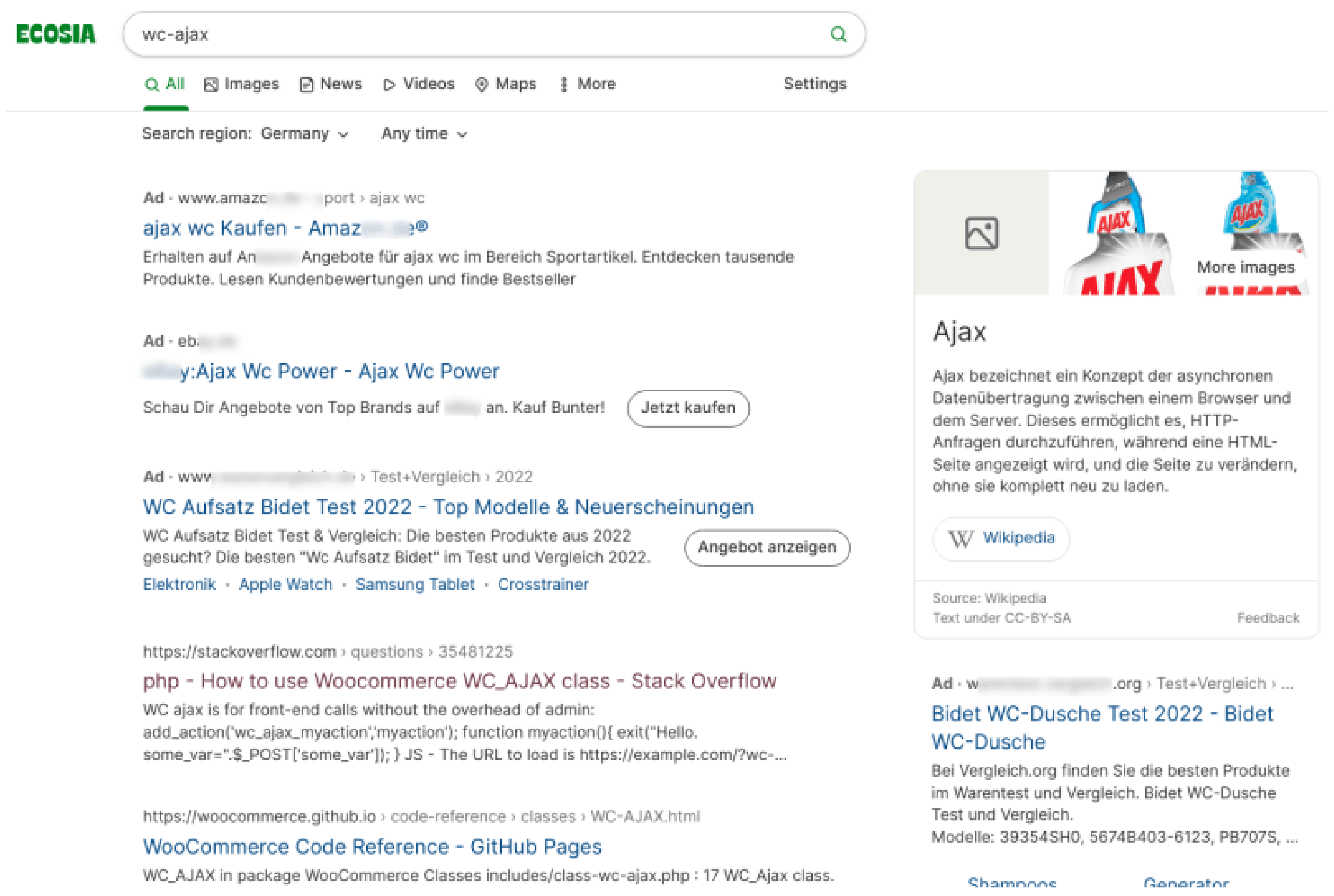
One bug that did not magically disappear though are the mysteriously missing products when customers use the product tiles' "add to cart" (AJAX) buttons to quickly, especially when the server does not respond quickly.

I started to inspect the issue, found several similar issues on StackOverflow, but no actual error in my developer tools. There is a `POST wc-ajax` call sending the products to add, receiving an updated mini-cart DOM fragment, and a matching number of subsequent cart update requests all with complete response JSON containing an updated cart content.

Only problem: some of the POST requests seem to be ignored in an erratic fashion, everything initiated and handled using built-in technology, and no apparent warnings, timeouts, or error messages.

The "real meaning" of "wc-ajax"?! 🤪

Funny thing, when searching for the problem's details and the URLs involved, I remembered, that "WC AJAX" is a popular toilet cleaning detergent in Europe, and only web developers will probably think about any other meaning when typing "wc-ajax" into their favorite search engine.



Ad · www.amazo[redacted]port › ajax wc
ajax wc Kaufen - Amaz[redacted]e®
Erhalten auf Am[redacted] Angebote für ajax wc im Bereich Sportartikel. Entdecken tausende Produkte. Lesen Kundenbewertungen und finde Bestseller

Ad · eb[redacted]
[redacted]y:Ajax Wc Power - Ajax Wc Power
Schau Dir Angebote von Top Brands auf [redacted] an. Kauf Bunter! Jetzt kaufen

Ad · www[redacted] › Test+Vergleich › 2022
WC Aufsatz Bidet Test 2022 - Top Modelle & Neuerscheinungen
WC Aufsatz Bidet Test & Vergleich: Die besten Produkte aus 2022 gesucht? Die besten "Wc Aufsatz Bidet" im Test und Vergleich 2022. Angebot anzeigen
Elektronik · Apple Watch · Samsung Tablet · Crosstrainer

https://stackoverflow.com › questions › 35481225
php - How to use Woocommerce WC_AJAX class - Stack Overflow
WC ajax is for front-end calls without the overhead of admin: add_action('wc_ajax_myaction','myaction'); function myaction(){ exit("Hello. some_var=".$_POST['some_var']); } JS - The URL to load is https://example.com/?wc-...

https://woocommerce.github.io › code-reference › classes › WC-AJAX.html
WooCommerce Code Reference - GitHub Pages
WC_AJAX in package WooCommerce Classes includes/class-wc-ajax.php : 17 WC_Ajax class.

More images

Ajax
Ajax bezeichnet ein Konzept der asynchronen Datenübertragung zwischen einem Browser und dem Server. Dieses ermöglicht es, HTTP-Anfragen durchzuführen, während eine HTML-Seite angezeigt wird, und die Seite zu verändern, ohne sie komplett neu zu laden.
Wikipedia
Source: Wikipedia
Text under CC-BY-SA
Feedback

Ad · w[redacted].org › Test+Vergleich › ...
Bidet WC-Dusche Test 2022 - Bidet WC-Dusche
Bei Vergleich.org finden Sie die besten Produkte im Warentest und Vergleich. Bidet WC-Dusche Test und Vergleich.
Modelle: 39354SH0, 5674B403-6123, PB707S, ...
Shampoos Generator

At least, there have been results, and some even cover the subject that I intended to search for.

So what could be the next steps to fix our problems? As I noted when documenting the state of my research, we could use *default* legacy code to replace our *custom* legacy code.

Stepping back to Best Practice Legacy Blocks

I found that the default legacy "archive product" block (`<!-- wp:woocommerce/legacy-template { "template": "archive-product"`) seems to be the most compatible and bug-free at the time of writing in January 2023. While it provides legacy functionality, including support for many popular third-party WooCommerce extensions, it is still a block that can safely be integrated in the block editor.

WooCommerce blocks are not bad per se, but "wp:woocommerce/all-products" seemed to be a problematic one, while "wp:woocommerce/handpicked-products" work fine for what its name describes: a small number of hand-picked products, as cross-selling recommendations on single product detail pages (PDP), partner listing etc.

Recuce Complexity!

As we already identified the variety and inconsistency as a crucial cause for problems and errors making it hard to maintain a custom WooCommerce block theme, we should try to reduce complexity and follow the principle of least surprise.

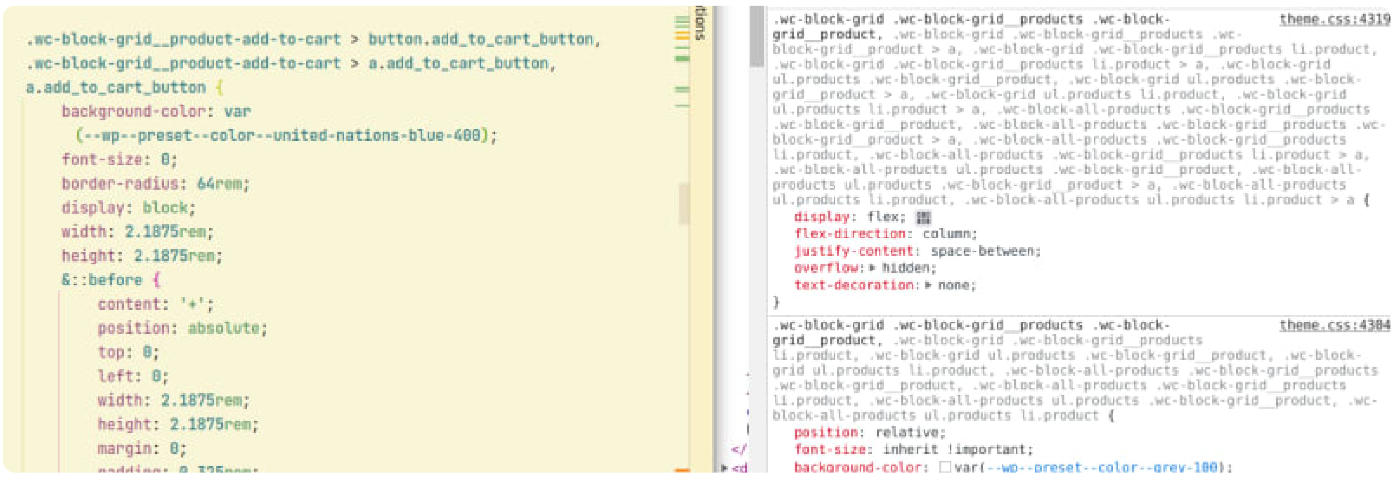
So let's restrict our code to maximum two different product grids (default legacy `archive-products`, plus `handpicked-products`, if needed) and if we did use more or different product grid / tile blocks before, it's time to clean up our style sheets and remove any unused selectors.

How to target different variations of WooCommerce Product Grids in CSS

If we do have full control over our template markup, we might want to add a custom class name to the product grid or its wrapper, but that contradicts my preference for default blocks with default parameter. Custom attributes could be destroyed by our customers making edits to their site content or adding a custom template using default blocks offfered by the editor library.

What they will proably not do (or at least they shouldn't) is using legacy shortcodes, so we only need to focus on the legacy template and the new blocks. So we have only a limited number of possible wrappers and element class name combinations, and we can remove obsolete legacy-legacy variations from our existing CSS to keep the code more compact and readable. Or at least that was what I had hoped to do, but it turned out that the legacy template block is quite similar to the legacy shortcode version, apart from the missing wrapper element. But at least this means that there are not that many combinations and edge cases left to consider.

And the actual answer how to target the different variations in CSS, you can use comma-separated lists inside selectors. But beware, if you combine that with (S)CSS nesting, this might soon lead to an unmaintainable bloated mess like this.



```
.wc-block-grid__product-add-to-cart > button.add_to_cart_button,
.wc-block-grid__product-add-to-cart > a.add_to_cart_button,
a.add_to_cart_button {
    background-color: var
      (--wp--preset--color--united-nations-blue-400);
    font-size: 0;
    border-radius: 64rem;
    display: block;
    width: 2.1875rem;
    height: 2.1875rem;
    &::before {
        content: '+';
        position: absolute;
        top: 0;
        left: 0;
        width: 2.1875rem;
        height: 2.1875rem;
        margin: 0;
```

```
.wc-block-grid .wc-block-grid__products .wc-block-                theme.css:4319
grid__product, .wc-block-grid .wc-block-grid__products .wc-
block-grid__product > a, .wc-block-grid .wc-block-grid__products li.product,
.wc-block-grid .wc-block-grid__products li.product > a, .wc-block-grid
ul.products .wc-block-grid__product, .wc-block-grid ul.products .wc-block-
grid__product > a, .wc-block-grid ul.products li.product, .wc-block-grid
ul.products li.product > a, .wc-block-all-products .wc-block-grid__products
.wc-block-grid__product, .wc-block-all-products .wc-block-grid__products .wc-
block-grid__product > a, .wc-block-all-products .wc-block-grid__products
li.product, .wc-block-all-products .wc-block-grid__products li.product > a,
.wc-block-all-products ul.products .wc-block-grid__product, .wc-block-all-
products ul.products .wc-block-grid__product > a, .wc-block-all-products
ul.products li.product, .wc-block-all-products ul.products li.product > a {
   display: flex;
   flex-direction: column;
   justify-content: space-between;
   overflow: ▸ hidden;
   text-decoration: ▸ none;
}
.wc-block-grid .wc-block-grid__products .wc-block-                theme.css:4304
grid__product, .wc-block-grid .wc-block-grid__products
li.product, .wc-block-grid ul.products .wc-block-grid__product, .wc-block-
grid ul.products li.product, .wc-block-all-products .wc-block-grid__products
.wc-block-grid__product, .wc-block-all-products .wc-block-grid__products
li.product, .wc-block-all-products ul.products .wc-block-grid__product, .wc-
block-all-products ul.products li.product {
   position: relative;
   font-size: inherit !important;
   background-color: ☐var(--wp--preset--color--grey-100);
```

So we will have to weed out the unused variations to keep our code clean and readable!

I have written down the different variation of elements and class names in a table so that I wouldn't have to inspect the frontend to prevent getting confused over and over again.

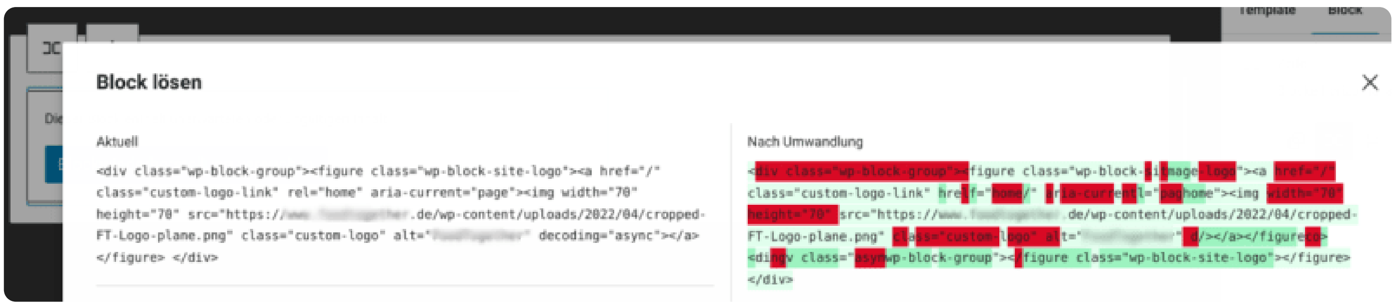
While the new product blocks come with several BEM-style class names, wrapped in a div with unsurprising class names, the legacy archive product template is quite minimalist, without any default wrapper, and a sparing `<ul class="products">`. While that might be enough to target the outer parts of our product grid, beware that the tile contents might be nested in a different order depending on the block or template used and possible third-party plugins like TP image flipper, which might also add additional image elements for galleries or hover effects. Apart from `wp:woocommerce/all-products` nesting `div > a > img`, all other variations covered in this post seem to use `a > div > img` consistently.

```
+-------------------------------------+---------+------------------------------------------------+
| block / template / shortcode        | element | markup / class names                           |
+-------------------------------------+---------+------------------------------------------------+
| wp:woocommerce/handpicked-products  | wrapper | .wc-block-grid.wp|wc-block-handpicked-products |
| wp:woocommerce/handpicked-products  | grid    | ul.wc-block-grid__products                     |
| wp:woocommerce/handpicked-products  | tile    | li.wc-block-grid__product                      |
| wp:woocommerce/handpicked-products  | content | a.wc-block-grid__product-link > div > img      |
| wp:woocommerce/handpicked-products  | image   | img.attachment-woocommerce_thumbnail           |
| wp:woocommerce/handpicked-products  | title   | div.wc-block-grid__product-title               |
+-------------------------------------+---------+------------------------------------------------+
| legacy-template: archive-product    | wrapper | (no wrapper)                                   |
| legacy-template: archive-product    | grid    | ul.products                                    |
| legacy-template: archive-product    | tile    | li.product.type-product                        |
| legacy-template: archive-product`   | content | a.woocommerce-loop-product__link > div > img   |
| legacy-template: archive-product`   | image   | img                                            |
| legacy-template: archive-product`   | title   | h2.woocommerce-loop-product__title             |
+-------------------------------------+---------+------------------------------------------------+
| [products] shortcode (obsolete)     | wrapper | div.woocommerce                                |
| [products] shortcode (obsolete)     | grid    | ul.products                                    |
| [products] shortcode (obsolete)     | tile    | li.product.type-product                        |
| [products] shortcode (obsolete)     | content | a.woocommerce-loop-product__link > div > img   |
| [products] shortcode (obsolete)     | image   | img                                            |
| [products] shortcode (obsolete)     | title   | h2.woocommerce-loop-product__title             |
+-------------------------------------+---------+------------------------------------------------+
| wp:woocommerce/all-products (buggy) | wrapper | div.wc-block-grid                              |
| wp:woocommerce/all-products (buggy) | grid    | ul.wc-block-grid__products                     |
| wp:woocommerce/all-products (buggy) | tile    | li.wc-block-grid__product.wc-block-layout      |
| wp:woocommerce/all-products (buggy) | content | div.wc-block-grid__product-image > a > img     |
| wp:woocommerce/all-products (buggy) | image   | img.product-image                              |
| wp:woocommerce/all-products (buggy) | title   | h2.wc-block-grid__product-title                |
+-------------------------------------+---------+------------------------------------------------+
```

TODO: maybe I should add a specific example of SCSS to target product tiles only for the markup variations generated by archive-products and handpicked-products.

How to avoid seemingly broken Custom Blocks

Avoid using the code editor and avoid maintaining custom block editor syntax, as the same code that used to be valid once, might cause a warning about a seemingly broken block that breaks the page preview and will make your customers or their content editors go crazy!



Aktuell

```
<div class="wp-block-group"><figure class="wp-block-site-logo"><a href="/"
class="custom-logo-link" rel="home" aria-current="page"><img width="70"
height="70" src="https://[illegible].de/wp-content/uploads/2022/04/cropped-
FT-Logo-plane.png" class="custom-logo" alt="[illegible]" decoding="async"></a>
</figure> </div>
```

Nach Umwandlung

```
<div class="wp-block-group"><figure class="wp-block-siimage-logo"><a href="/"
class="custom-logo-link" hreilf="home/" aria-currentl="paghome"><img width="70"
height="70" src="https://[illegible].de/wp-content/uploads/2022/04/cropped-
FT-Logo-plane.png" class="custom-logo" alt="[illegible]" d/></a></figureco>
<dingv class="asynwp-block-group"></figure class="wp-block-site-logo"></figure>
</div>
```

Instead, delete any unnecessary customization and replace it with the current version of a default block, provided by WordPress, WooCommerce, WooCommerce, or one of their top recommended partner plugins, nothing else!

There are ways to customize content, besides custom blocks, there are template parts, there are shortcodes, and mostly everything can be inserted using a "custom HTML" block to encapsulate our customizations without surprising the block editor or its users.

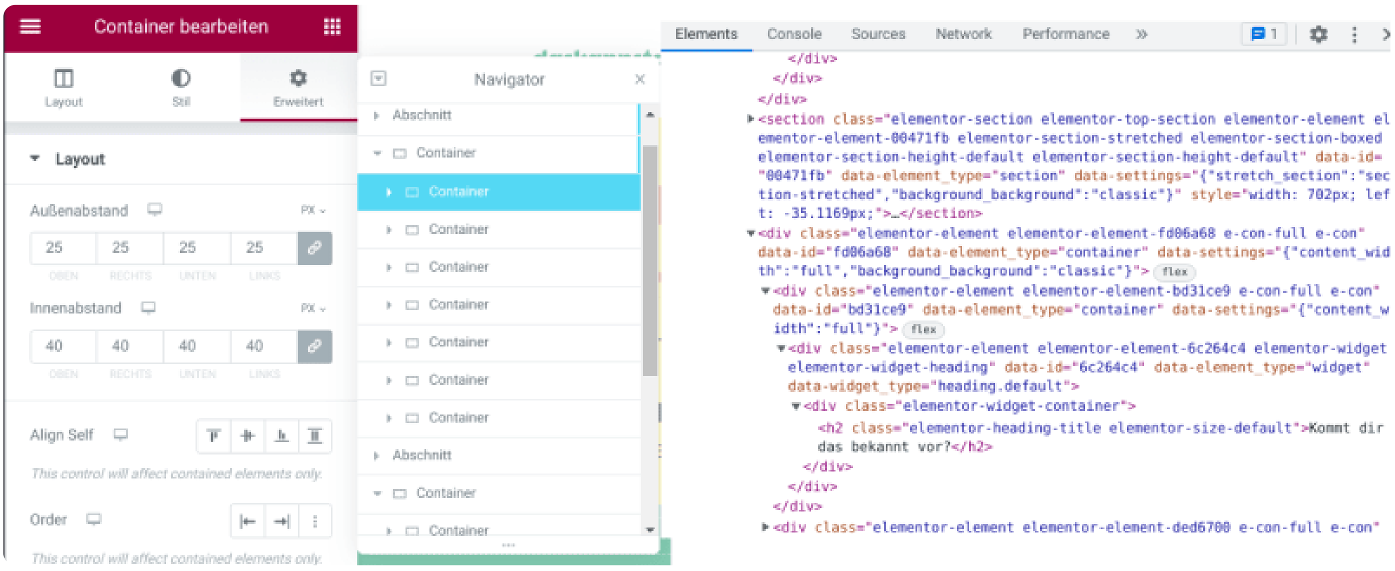
And we still have `functions.php` with its `add_action`, `add_filter`, `wp_enqueue_script` etc.

Just when I thought Gutenberg sucks

... I had the pleasure to work with Elementor.

And if you want to edit it, you have a tab "layout" with a section "container", where you can also apply flexbox styles that apply to the child elements. Other styles are edited in the "style" tab, while the "advanced" tab has another "layout" section where you can edit layout styles.

The generated markup looks just like that. At least in this use case, every element is an `.elementor-element`, and every container is also either a "full-width" (`.e-con-full`) or a "boxed" one (`.e-con-boxed`). Good luck if you want to apply global CSS fixes only to certain elements or constallations. Hard or impossible even with parent selectors. You have to think modular an add the same patches as element-based custom CSS instead, as every single element also has an individual ID-ish class like `.elementor-element-96197c0`, but then how would customers reuse that?



Außenabstand: 25 25 25 25 (OBEN, RECHTS, UNTEN, LINKS)

Innenabstand: 40 40 40 40 (OBEN, RECHTS, UNTEN, LINKS)

This control will affect contained elements only.

This control will affect contained elements only.

```
      </div>
    </div>
  </div>
 ▸<section class="elementor-section elementor-top-section elementor-element el
  ementor-element-00471fb elementor-section-stretched elementor-section-boxed
  elementor-section-height-default elementor-section-height-default" data-id=
  "00471fb" data-element_type="section" data-settings="{"stretch_section":"sec
  tion-stretched","background_background":"classic"}" style="width: 702px; lef
  t: -35.1169px;">…</section>
 ▾<div class="elementor-element elementor-element-fd06a68 e-con-full e-con"
  data-id="fd06a68" data-element_type="container" data-settings="{"content_wid
  th":"full","background_background":"classic"}"> flex
   ▾<div class="elementor-element elementor-element-bd31ce9 e-con-full e-con"
    data-id="bd31ce9" data-element_type="container" data-settings="{"content_w
    idth":"full"}"> flex
     ▾<div class="elementor-element elementor-element-6c264c4 elementor-widget
      elementor-widget-heading" data-id="6c264c4" data-element_type="widget"
      data-widget_type="heading.default">
       ▾<div class="elementor-widget-container">
          <h2 class="elementor-heading-title elementor-size-default">Kommt dir
          das bekannt vor?</h2>
        </div>
      </div>
    </div>
   ▸<div class="elementor-element elementor-element-ded6700 e-con-full e-con"
```

So let's hope that the WordPress core team fixes Gutenberg's bugs and conceptual shortcomings soon, otherwise more customers might insist we have to use Elementor instead, which is none the better.

In Elementor, everything is an "element", while in the WYSIWYG UI navigator, mostly everything is a "container".

Using WordPress without losing our Heads as Developers

In most cases, let's just use default blocks (elements / components), focus on adding CSS styles and some custom JavaScript, and if our customers want anything more custom, offer them to make an estimation for a new project using a proper e-commerce system like Shopware, or anything "headless" using the WordPress and WooCommerce APIs.

From a developer's perspective, I would even stick to classic themes (which can still use the block editor, probably restricted to certain fields or post types), provide block patterns / templates on request, and ignore theme.json and full site editing as along as possible — or at least, until it is stable, documented, and code validation provides helpful output like line numbers or findings instead of the ominous "unexpected or invalid content" message.

Coming up next: Using WordPress without losing our Heads

Using WordPress as a Developer (10 Part Series)