

Layout Builder Components can Break Your Site.

Here's How.

Virtual DrupalCon
July 14, 2020

Who Am I?

- André Angelantoni, Founder of Performant Labs
- Working with Drupal since 5.0
- Clients include France Telecom, CBS Interactive, DocuSign, Goldman Environmental Prize, Robert Half and Tesla
- Performant Labs is sponsoring Layout Builder Kit, Campaign Kit, Payment Stripe and is the founder of the Drupal Quality Initiative



Where to Find This Talk

- performantlabs.com/articles/drupalcon-2020-talk
- This is Part 1 of this important topic
- I'll deliver Part 2 at a camp later this year, perhaps at BADCamp but I don't know yet
- Leave your email address at the link above to be notified

What We'll Cover

- Instant Introduction to Layout Builder
- Why Do We Use Content Management Systems?
- Component Anatomy
- The Fundamental Problem
- The Solution
- A Performance Warning
- Layout Builder Kit
- Layout Builder Rules to Live By
- The Emerging Layout Builder Ecosystem
- Resources
- Q&A

A Note on Nomenclature

- When discussing Layout Builder:
 - a component is the same thing as a block.
 - in most instances, a page is the same thing as a node.



Instant Introduction to Layout Builder

- Layout Builder is a souped-up Panelizer! (If you've worked with that module.)
- If you haven't, Layout Builder is a way to control the presentation of several parts of Drupal, including nodes, blocks, taxonomy terms and menus.
- You can create rigid layouts that content editors can't change or layouts almost completely editable by content editors—or somewhere in between.



What Kind of Site Do You Want?



Why do we use content management systems?

**To make managing the content on
our website easier for the teams
that work on them.**

Ok.

**How do content management
systems do *that*?**

They have a laundry list of features from managing media assets to spell checking.

But two features are particularly important for many teams.

Important Feature #1

- Page versioning
 - Each change to the page saves a new version.
 - Content editors want to roll back changes quickly!

Important Feature #2

- Workflow
 - Content editors want to collaborate and ensure high quality. Changing the Workflow status from Draft to Needs Review to Published helps with both items.
- Let's talk about versioning first.

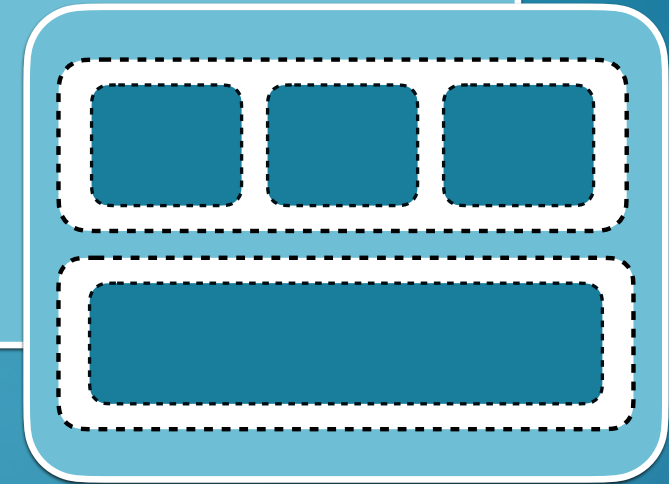
But first, we need to discuss the two different ways CMSs make pages.

The Static Approach

- Some CMSs create a full HTML page and write it into a file, like Adobe Experience Manager.
- Each page is stamped with a date.
- To see "back in time," just pull up an older version of the file because the content was all written into it.
- To make that older page current, just rename the old version of the file; something like "FrontPage-2020-02-01.html" to "FrontPage.html"

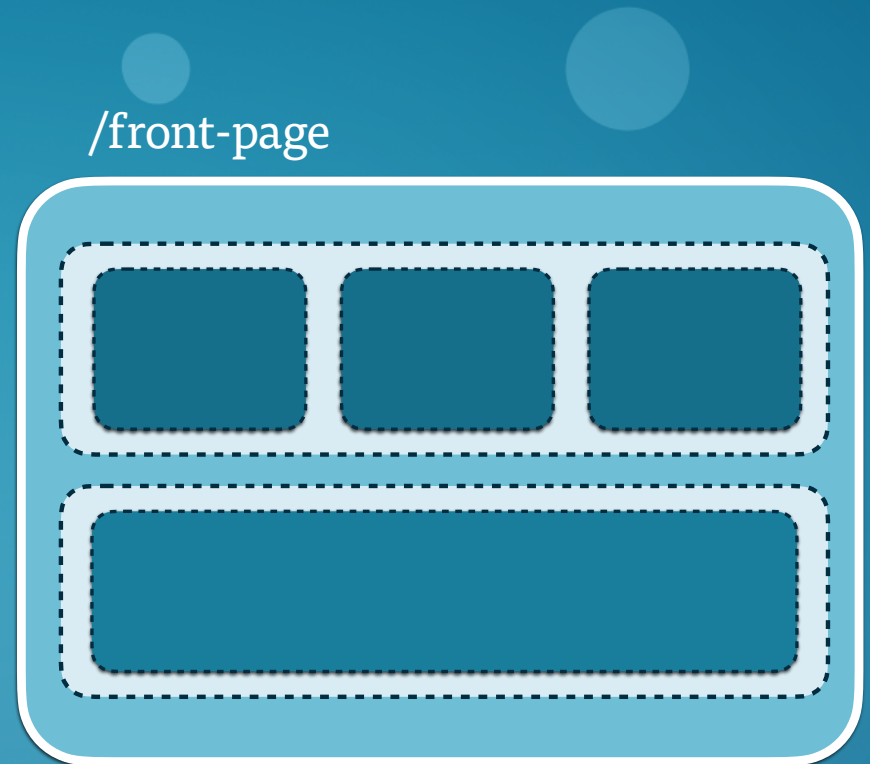
FrontPage-2020-02-01.html

FrontPage.html



The Dynamic Approach

- The page is rendered on-the-fly, sometimes for each user (when the user is logged in).
- This is the system Drupal uses.
- We do add several layers of caching, which stores a version of the page, but those page versions are not retained long-term and cannot be used for versioning.



So what's the problem?

**The problem is that we may break
these two features
(versioning and workflow)
depending on where the
component content is stored
AND the complexity of the component.**

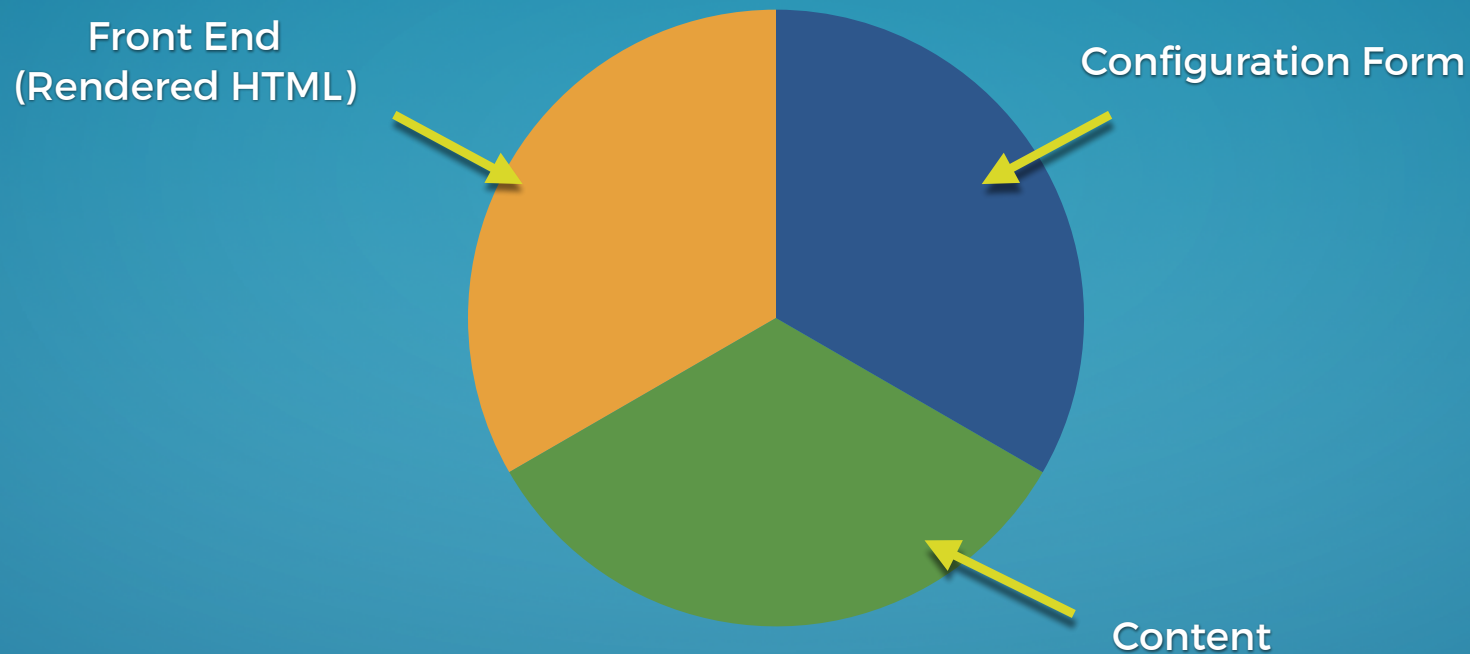
**It's not just that you can
break the site.**

**This is one of those things in
which people may start saying,
"Drupal is broken."
But it's not Drupal—it's actually
an implementation issue.**

Component Anatomy

How Are Components Made?

- Components in Drupal are blocks and you could divide them into three parts.



How Are Components Made?

- Part 1:
 - the rendered HTML (the button or carousel users see)
 - rendered by TWIG templates or a front-end framework like React

Checkout

1 Shipping address — 2 Payment details — 3 Review your order

Shipping address

First name* Last name*

Address line 1*

Address line 2

City* State/Province/Region

Zip / Postal code* Country*

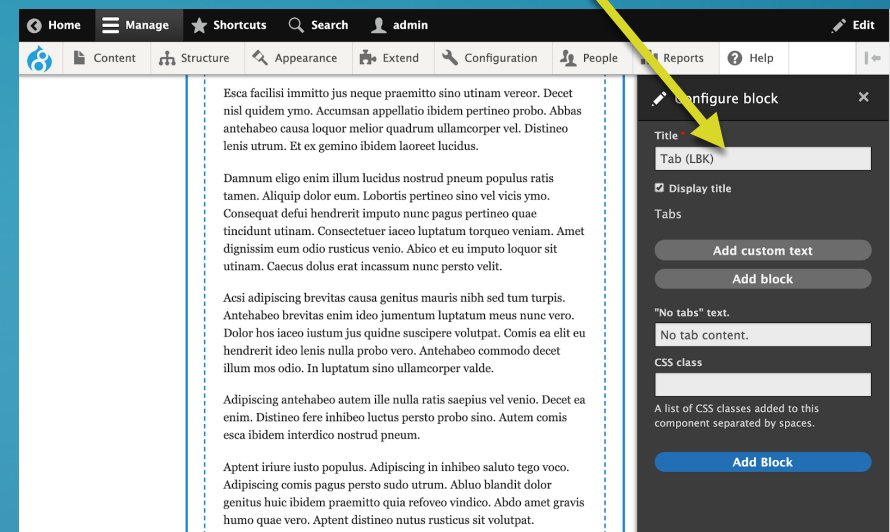
☐ Use this address for payment details

NEXT

What Are Components? (cont'd)

- Part 2:
 - the configuration form

This is a Layout Builder component configuration form.



The screenshot shows the Drupal Layout Builder interface. A yellow arrow points from the text 'This is a Layout Builder component configuration form.' to the 'Configure block' dialog box. The dialog box is titled 'Configure block' and has a close button (X). It contains the following fields and options:

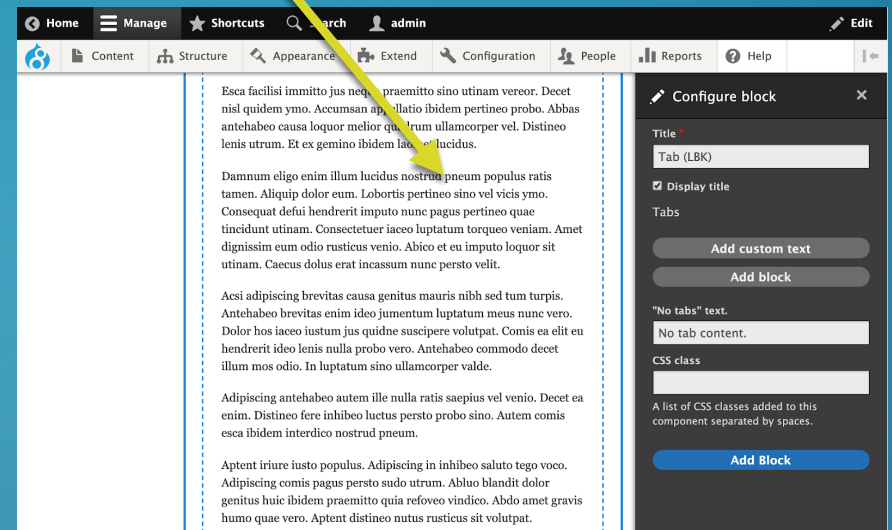
- Title:** A text input field containing 'Tab (LBK)'.
- Display title:** A checked checkbox.
- Buttons:** Two buttons labeled 'Add custom text' and 'Add block'.
- 'No tabs' text:** A text input field containing 'No tab content.'
- CSS class:** A text input field.
- Footer:** A note stating 'A list of CSS classes added to this component separated by spaces.' and a blue 'Add block' button.

The background shows a preview of the component, which is a tab with the title 'Tab (LBK)' and several paragraphs of placeholder text.

What Are Components? (cont'd)

- Part 3:
 - the content to be displayed

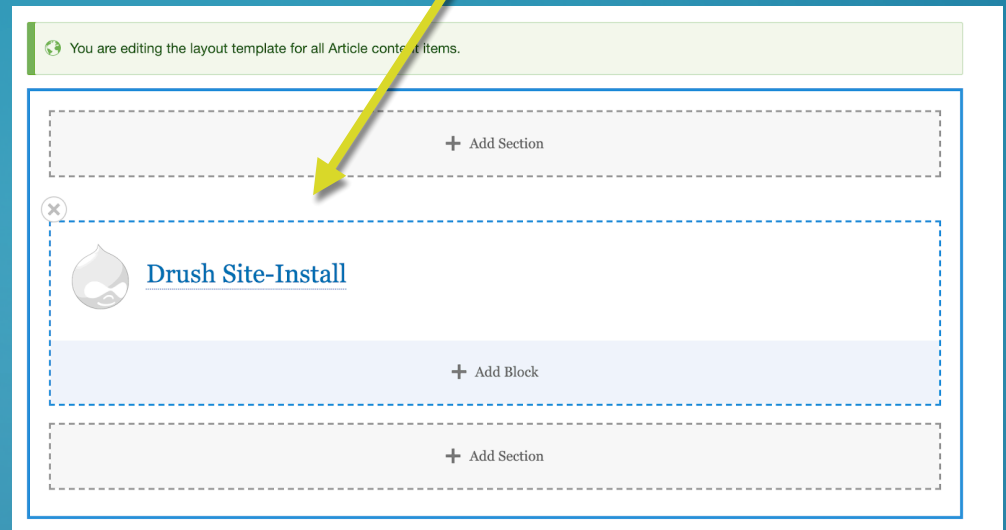
Content!



Where is component configuration stored?

- We are interested in more than the presentation of the component, such as whether it has a border around it.
- We are also interested in the content of the component.

Where is the configuration for this stored?



First Location: With the Entity Definition

- It can be with the content type definition, the block definition, the taxonomy definition or the menu definition.
- When stored here, the configuration applies to all entities (i.e. all nodes of that content type, etc.).

With just this checkbox, configuration is stored in the content type template.

Applies to all nodes of this content type.



First Location (cont'd)

You will be told that you are editing the layout template.



Edit layout for *Book page content items*

[Home](#) » [Administration](#) » [Structure](#) » [Content types](#) » [Book page](#) » [Manage display](#)


This layout builder tool allows you to configure the layout of the main content area.

To manage other areas of the page, use the [block administration page](#).

Forms and links inside the content of the layout builder tool have been disabled.

[Save layout](#) [Discard changes](#)

☒ Show content preview

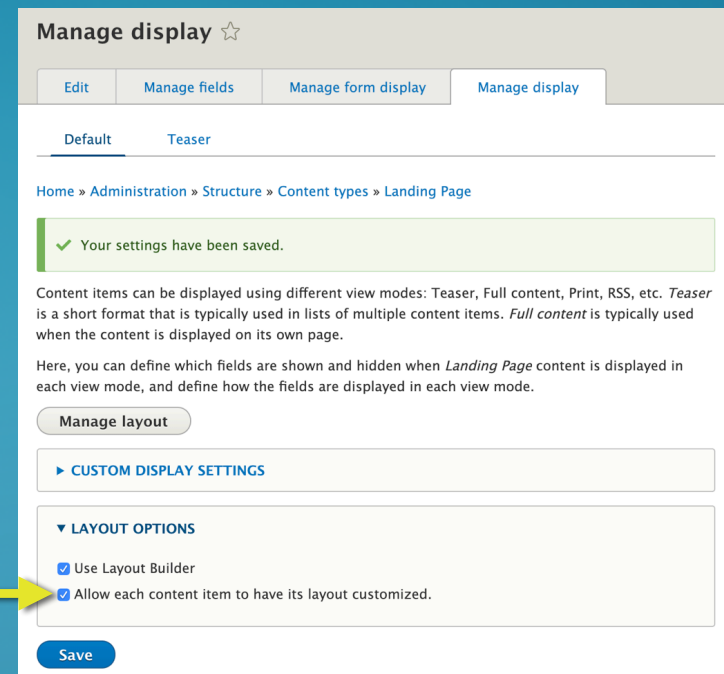
 You are editing the layout template for all Book page content items.

[+ Add Section](#)

Second Location: With the Node, Block, Taxonomy or Menu

- It may also be stored with each entity (node, block, taxonomy term, menu). This allows "template overrides" by entity when it is turned on.

With both checkboxes, configuration is stored in the entity template **and** on each entity.



The screenshot shows the 'Manage display' interface for a 'Landing Page' content type. It includes tabs for 'Edit', 'Manage fields', 'Manage form display', and 'Manage display'. The 'Manage display' tab is active, showing 'Default' and 'Teaser' view modes. A success message states 'Your settings have been saved.' Below this, explanatory text describes view modes and the purpose of the settings. A 'Manage layout' button is present. Under 'CUSTOM DISPLAY SETTINGS', the 'LAYOUT OPTIONS' section contains two checked checkboxes: 'Use Layout Builder' and 'Allow each content item to have its layout customized.' A 'Save' button is at the bottom.

Second Location (cont'd)

- You will notice Drupal adds an extra *field* to the entity (content type, block, taxonomy term or menu).







+ Add field		
LABEL	MACHINE NAME	FIELD TYPE
Body	body	Text (formatted, long, with summary)
Layout	layout_builder__layout	Layout Section

This extra field appears when
template overrides are turned on.

Second Location (cont'd)

- To store that extra field, Drupal adds an *extra table* to the database. If revisions are turned on, it adds the revision table, too (not shown below).

Nodes	node__layout_builder__layout
Blocks	block_content__layout_builder__layout
Menus	menu_link_content__layout_builder__layout
Taxonomy	taxonomy_term__layout_builder__layout

	node
	node__body
	node__comment
	node__field_image
	node__field_tags
	node__layout_builder__layout

How is the configuration stored?

- It's stored as a serialized string.









```
O:29:"Drupal\layout_builder\Section":  
4:  
{s:11:"*layoutId";  
s:13:"layout_onecol";  
s:17:"*layoutSettings";  
a:0:{}  
s:13:"*components";  
a:0:{}  
s:21:"*thirdPartySettings";  
a:0:{}  
}
```


Where is the Content Saved?

- When working with blocks, the *content* is stored in block tables that follow the same structure as the other entities (nodes, menus, taxonomy terms):

Tables for a basic block design. Adding fields creates more tables.



-  block_content
-  block_content__body
-  block_content__field_media_field
-  block_content_field_data
-  block_content_field_revision
-  block_content_revision
-  block_content_revision__body
-  block_content_revision__field_media_field

What was the problem, again?

Which version is displayed?

- The problem arises with older versions of the page. Which version of the component should be displayed?
- For Drupal to know, the revision of the block needs to be stored with the layout.
- As of Drupal 8.9/9.0, Global/Reusable Components do not store their Revision ID with the layout.
- **You could revert to an older page—but it would show the most current version of the components, thereby breaking these key features!**

**Do you want to be the one telling
the client that page revisioning,
workflow and workspaces
are all broken?**

Warning: Watch out for Complexity!

- And what about parent-child relationships in components, like a field that points to another object?
- **These, too, are not stored in the layout and pulling up old pages may show the most recent version of the child if you aren't careful.**

Where is the Content Saved? (cont'd)

- One way out when making custom components is storing the *content* with the layout as part of the serialized string; content is automatically revisioned this way.
- Version 1 of Layout Builder Kit components use this mechanism.
- This is a perfectly valid way to get around this problem.

```
O:29:"Drupal\layout_builder\Section":  
4:  
{s:11:"*layoutId";  
s:13:"layout_onecol";  
s:17:"*layoutSettings";  
s:45:{"The quick brown fox jumped over the lazy dog."}  
}
```


So, how should you proceed?

"Carefully"

- Be very aware of how your component content is stored especially if older revisions or workflow or workspaces is needed immediately or in the future. Understand the details of each component type and test everything thoroughly.
- Do not assume — test and keep detailed notes!
- Teach everyone on the team about this issue.

Leveraging the Framework

- If there are so many problems with storing content in tables, why not just store everything in the serialized string and call it a day?
- There are other parts of Drupal that open up to you by using tables, like using the Diff module to compare what changed between two versions of the page. Or getting your content into the search index.
- And many more contributed modules will be made that work with content stored in fields instead of serialized strings.

A Performance Warning

A Table per Field

- Recall that when component content is stored in fields instead of serialized and stored with configuration, Drupal creates a table for every field.
- It does this in part to allow for fast database structure changes on production databases.
- To collect all the content for a single component requires a SQL join statement between multiple tables.
- If you have a page with many components, collecting all the content for all the components will be really slow. (We tested this; it was unusably slow on pages with many components.)

The Field API

- Consider a component that has three fields: the event title, date and location.

Table: Event title

Entity ID	Event Title
1	Title 1
2	Title 2
3	Title 3

Table: Event date

Entity ID	Event Date
1	Date 1
2	Date 2
3	Date 3

Table: Event location

Entity ID	Event Location
1	Location 1
2	Location 2
3	Location 3

- Drupal fetches these in a single query with SQL JOINS and populates an internal object with the result.
- More fields per component → slower; more components on the page → slower

The Workaround

- Make your components in custom modules and store all the content for the component in a single table using the Entity API—not the Field API. (Go old school!)

Table: Event title

Entity ID	Event Title	Event Date	Event Location
1	Title 1	Date 1	Location 1
2	Title 2	Date 2	Location 2
3	Title 3	Date 3	Location 3

- Drupal fetches with far fewer JOINS → MUCH faster.

Layout Builder Kit

- Layout Builder Kit 1.x currently uses serialized string storage. This appears to be the same system Acquia Cohesion uses. (Acquia people: correct me if this is incorrect.)
- The components so far:
 - Book Navigation
 - Rich Text
 - Icon Text
 - Tab
 - Image
 - Video
 - Render
- For a full walk-through of Layout Builder Kit and how to contribute new components: webcamp.stanford.edu/session/introducing-layout-builder-kit



What's Next for LBK?

- Version 2.x of Layout Builder Kit will add components that use the Entity API.
- They could be based on classes that help developers construct slightly more complex components *without* damaging the versioning mechanism.
- Site builders can then judge competing requirements and choose the storage mechanism that makes sense for their build.
- Contact me if you want to help develop those base classes.

I fear there will be many “broken” Drupal sites made by teams unaware of these issues.

Unfortunately, Drupal will be blamed when it's actually a site build issue.

Don't let your site be one of these.

The Emerging Layout Builder Ecosystem

- Layout Builder Asset
- Layout Builder Asymmetric Translation
- Layout Builder Browser
- Layout Library
- Layout Builder Modal
- Layout Builder Restrictions
- Block Blacklist
- Layout Builder Styles
- Dynamic Layouts
- Entity Browser Block
- Mini Layouts
- Layout Builder Everywhere
- Layout Builder Symmetric Translations
- Layout Builder UX

Some Interesting Notes

- It's not possible yet to export and import layouts but it should be ready soon:

Expose Layout Builder data to REST and JSON:API

drupal.org/project/drupal/issues/2942975

- Layout Builder Everywhere will extend Layout Builder to regions of the page other than the main content area.

drupal.org/project/lb_everywhere

Some Interesting Notes

- A patch brings Layout Builder into Page Manager.

Create a layout builder variant

drupal.org/project/page_manager/issues/2960739

Patch Committed!

Resources

- Watch a video about the Layout Builder ecosystem here:
2019.badcamp.org/session/start-using-emerging-layout-builder-ecosystem
- See the list of Layout Builder ecosystem modules here:
drupal.org/docs/8/core/modules/layout-builder/additional-modules

Resources

- Additional systems to investigate:
 - DXPR Builder: dxpr.com
 - Acquia Cohesion: www.acquia.com/products-services/acquia-cohesion
 - As far as I can tell, both store their configuration on the pages rather than in tables
 - Elementor (drupal.org/project/elementor)
 - Gutenberg (drupal.org/project/gutenberg)

Q&A